

UNIVERSITÉ DE LILLE

THÈSE DE DOCTORAT

pour obtenir le grade de :
DOCTEUR DE L'UNIVERSITÉ DE LILLE
dans la spécialité INFORMATIQUE ET APPLICATIONS
par

Marc Damie

Privacy-Preserving Computations on Sparse Data

Calculs respectueux de la vie privée
sur données éparses

Thèse soutenue le 11 décembre 2025 devant le jury composé de :

Président du jury:

M. Romain Rouvoy	Professeur	Université de Lille (FR)
-------------------------	------------	--------------------------

Rapporteurs:

M. Geoffroy Couteau	Chargé de recherche	IRIF, UMR 8243, Université Paris Cité (FR)
----------------------------	---------------------	--

M ^{me} Katerina Mitrokotsa	Professeure	University of St. Gallen (CH)
--	-------------	-------------------------------

Examineurs:

M ^{me} Lisa Kohl	Chargée de recherche	CWI Amsterdam (NL)
----------------------------------	----------------------	--------------------

M. Thijs Veugen	Professeur	TNO & University of Twente (NL)
------------------------	------------	---------------------------------

Directeurs de thèse:

M. Jan Ramon	Directeur de recherche	Inria Lille (FR)
---------------------	------------------------	------------------

M. Andreas Peter	Professeur	Carl von Ossietzky Universität Oldenburg (DE)
-------------------------	------------	---

Co-encadrant:

M. Florian Hahn	Professeur Assistant	University of Twente (NL)
------------------------	----------------------	---------------------------

Acknowledgements

While the advancement of Science should be the only motivation to read a thesis, my short scientific career has taught me one thing: people who open theses are only interested in the acknowledgements – usually to know whether they are mentioned. Thus, I spent four years working on this manuscript, but all the pressure now resides in this (purely non-scientific!) section.

To the jury **Prof. Mitrokotsa**, thank you for agreeing to write a review report on my thesis. As my work covers a wide range of topics in privacy-enhancing technologies, I am glad to have you in the jury, as your broad expertise in computer security and cryptography will be essential to evaluate the entirety of my work. **Dr. Couteau**, thank you also for agreeing to write a review report on my thesis. Several core contributions of my thesis are related to Function Secret Sharing, so I am glad to have you; one of the researchers who contributed to key works in the HSS/FSS literature. **Dr. Kohl**, thank you for being part of this jury. Like Dr. Couteau, you have contributed (sometimes with him) to central papers in the FSS/HSS literature, so I am looking forward to discussing my own FSS contributions with you. **Prof. Veugen**, thank you for being part of this jury. As you know, I would like to move toward applied research, so I found it essential to have in my jury an MPC expert at the intersection of academia and industry. Finally, **Prof. Rouvoy**, thank you for accepting to chair this jury. Your expertise in distributed systems and “green” ICT is essential to complete this jury and to evaluate my latest contributions.

On that note, I wish “*bon courage*”¹ to the jury for reading this manuscript.

To my supervisors Throughout these four years, you gave me a huge independence to explore all possible directions and build my own Ph.D. project, and I thank all three of you (**Jan**, **Andreas**, **Florian**) for that.

Andreas and **Florian**, you have been there since the very beginning of my academic career, starting from my research internship. Even though I was an ML student from a small French university, you still agreed to supervise me on cryptographic topics; despite my complete lack of cryptography knowledge at that time. My transition from ML to cryptography and privacy was an audacious bet for me, and it was most likely also a bet for you as supervisors. I am truly thankful that you made this bet with me.

Andreas, I have always been impressed by your ability to analyze research works fairly and precisely. I remember you apologizing in advance for (barely negative) feedback on my work; an apology that was never needed, as your comments were

¹Commonly translated as “good luck,” but this translation is unsatisfying as it loses the reference to courage... and courage is definitely more useful than luck when reading a thesis.

always fair, accurate, and actionable. Moreover, your impressive diplomatic skills made every discussion constructive, so I have always appreciated your feedback, whether positive or negative.

Florian, you were the one person with whom I could discuss (and complain) about reviewer B. You have always been positive and understanding about any idea I came up with. This positivity was contagious and helped me keep moving forward with my research.

To my co-authors **Edwige**, our paper is the one I am most proud of. It is not the most complex or the most impressive. It is simply the most meaningful: it delivers a clear message. **Jean-Benoist**, our collaboration led to a highly unusual but much-needed cryptography paper. **Federico**, it has been a pleasure to collaborate with you recently; our cryptographic expertises complemented each other perfectly. **Chakshu**, supervising a student with you has been a great opportunity to explore a new area of cryptography. Finally, I thank all the Master's and Bachelor's students I have supervised, whose work is published or in the process of being published: **Marco**, **Bram**, **Merijn**, **Mihai**, and **Nilufer**.

To my Inria colleagues As a proud member of the Ph.D. room, I had the opportunity to meet many current and former students, but two of them marked my time in Lille: **Edwige**, thank you for all our Monday discussions, and **Paul M.**, thank you for the conversations and the beers we shared. I want to also thank all colleagues with whom I may have shared a discussion, an idea, a meal, or a drink²: Mikaela, Antoine, Gaurav, Sophie, Bastien, Dimitri, Joseph, Pascal, Aurélien, Cesar, Aleksei, Michael, Rémi, Nathalie, Kamal, Nathan, Batiste, Paul A., Arijus, Quentin, Baptiste, Damien, Tudor. A nice team would be nothing without a great lead, so thank you, **Marc**, for everything you do for the team. Finally, I thank the team assistants, **Aurore** and **Nathalie**, for making research life so much easier.

To my Twente colleagues In the middle of my Ph.D., I moved to a new country but more importantly, to a new Ph.D. room. This led to new connections, notably with my fellow crypto Ph.D. students: **Vasco** and **Federico**. I extend this acknowledgement to all the colleagues I had the chance to meet in Twente³: Jorik, Valeriu, Amina, Matteo, Aditya, Stefano, Maarten, Bob, Luca, Thijs, Mattia, Claudio, Tiago, Roland, Isadora, Gianpietro, Philipp, Willem, Constantin, Patricia, Zsolt, Herson, Chakshu, Jerre. I also want to thank the management of our group – **Andrea**, **Giancarlo**, and **Florian** – especially as my unusual Ph.D. organization caused some administrative headaches. Even if the Dutch bureaucracy is not as legendary as the French, the help of our team assistants, **Bertine** and **Laura**, has been essential to make my experience much smoother.

²As Magnet is an ML team, I obviously asked ChatGPT to shuffle the names.

³As SCS values computer security, I used a cryptographically secure PRG to shuffle this list.

Finally, a special thank you to all members of the **Twente Hacking Squad** for all these ~~uns~~successful CTF challenges.

To some inspiring researchers As a young Ph.D. student, I entered academia with all the ideals we commonly associate with it. This first research experience was also an opportunity to discover academia beyond those ideals. Unfortunately, the “publish or perish” culture has made it drift away from them. As I now reach the end of my Ph.D., I feel a certain bitterness about the societal impact of academic research. Indeed, I often wonder about the true goal of modern research: at times, it seemed that publishing complex papers conforming to rigid standards matters more than designing systems that are genuinely useful to people.

However, from time to time, some researchers have given me hope that research can still improve people’s lives. **Phil Rogaway**, thank you for your remarkable paper on the moral character of cryptographic work [Rog15]. I came to cryptography to fight for online privacy, and it was deeply reassuring to read that such political motivations can still have a place in cryptographic research. **Seny Kamara**, thank you for reminding the community that we should start designing *cryptography for the people*. That is exactly what I want to do for the rest of my career, and I am grateful that such a respected cryptographer is spreading this message.

Finally, **Franck Ghitalla**, thank you for showing me that a research career does not need to be standard to be impactful; it can even be chaotic! You were not a computer scientist, yet you taught me more about this discipline than nine years of formal education ever did.

To organizations fighting for online privacy Early in my studies, I joined organizations promoting open-source culture and online privacy. In fact, these activist activities were a key factor motivating my switch to cryptography. Indeed, many non-profit organizations have long promoted cryptographic tools as essential instruments in the fight against online surveillance. At some point, I felt the need to develop this expertise to support such movements.

Thus, I would like to thank all non-profit organizations defending open-source software, online privacy, and alternative digital services. More specifically, I thank **Framasoft**, **Picasoft**, **Signal Foundation**, **Linux Foundation**, **Mastodon**, and all organizations (and individuals) developing the alternative digital public spaces that make up the Fediverse and the ATmosphere.

Now that I have developed expertise in cryptography and privacy, it is time to give back. I look forward to collaborating with any non-profit organization that could benefit from a pro bono engineer or researcher.

As with any good document or presentation prepared by a French person, these acknowledgements need to randomly switch to French in order to reach their full potential. Thus, I apologize to any English-speaker still reading these lines. If I learned anything from meeting international researchers, it is that an unreasonable proportion of them learned French in high-school *and then* have forgotten everything about it except *croiiiissaaaaant*. The following paragraphs will then be your occasion to reconnect with your inner French speaker.

Ma soutenance marque la fin d’une étape importante de ma vie, à la fois professionnelle et personnelle. À l’approche de cet événement, je dois avouer avec un certain plaisir que je suis toujours bien entouré. La thèse de doctorat est un exercice solitaire, qui peut parfois isoler. L’une de mes fiertés durant ces années a été d’entretenir ces amitiés.

Le travail de chercheur nous immerge dans un microcosme avec ses propres règles du jeu. On se prend vite au jeu, en enchaînant les articles scientifiques et les frustrations liées au douloureux processus de publication. Conserver des amitiés fut donc essentiel pour garder les pieds sur terre, pour se rappeler qu’un article scientifique (où qu’il soit publié) ne changera pas la face du monde, pour se rappeler qu’il existe une vie et un monde au-delà de notre recherche.

À mes ami-es “UTCéen·nes” Claire, Tom et Margaux, nos voyages réguliers ont été des respirations bienvenues. Comme à chaque fois, j’attends avec impatience le prochain! À tous les autres UTCéens (François, Morgane, Lisa, Clément, Marianne, Guillaume), les occasions sont rares, mais c’est toujours un plaisir de tou·tes se retrouver pour un anniversaire, une crémaillère ou quelque autre événement important de nos vies respectives.

À mes ami-es “fénélonien·nes” Cédric, Erwan, Léo et Nihel, vous avez souvent été étonnés que je vienne des Pays-Bas à la moindre soirée que vous proposiez. J’espère que mes épuisants trajets en Flixbus témoignent de l’intérêt intact que je porte pour nos retrouvailles.

À mes “vieux” amis Thomas, j’ai eu le plaisir de récemment reprendre contact afin de travailler avec toi, Alix et Yann sur un projet audiovisuel à propos d’Internet et des GAFAM. Ces échanges comptent parmi les discussions les plus intéressantes que j’ai eues au cours de ces quatre dernières années... elles m’ont rappelé pourquoi je suis devenu informaticien: pour construire un monde numérique meilleur.

Enfin, Antoine, je ne m’explique toujours pas la longévité de notre amitié. Nous sommes amis depuis toujours et le temps comme la distance ne semblent avoir aucun effet sur cela. Je ne saurais mettre le doigt sur la recette de notre longue amitié, mais je te remercie d’être toujours là.

À Circé  Il est très difficile de quantifier en quoi un chat qui dort peut contribuer sensiblement à l'écriture d'un article. Il n'en reste pas moins que tu as été la personne la plus présente durant la rédaction de mes nombreux papiers; même si cette assistance fut très passive. Ainsi, il convient de te remercier dans ta propre langue: *miaou miaou miaouuu miaou !*

À ma famille Je tiens naturellement à remercier ma famille qui a contribué à la personne que je suis devenue: parents, frère, soeur, grand-parents, tantes, oncles et belle famille. **Papy Alphonse**, à chaque diplôme que nous obtenions, tu étais étonné qu'on ait pu le faire alors que tu avais à peine le certificat d'études. Malheureusement tu n'es plus là pour apprécier mon dernier diplôme. **Mamy Françoise**, tu étais là dans les premières étapes de mon parcours scolaire en venant me chercher à l'école maternelle et c'est une immense joie que tu sois toujours là pour cette dernière étape qu'est le doctorat. **Nicolas**, je te remercie d'avoir été la personne ayant amené ma passion pour l'informatique; tu m'as accompagné depuis ma découverte des jeux-vidéos jusqu'au montage de mon premier ordinateur. **Flora**, mon doctorat ne changera pas le fait que tu es l'enfant préférée, mais je te remercie quand même d'être la première source de divertissement de la famille. **Papa et Maman**, vous avez fourni le meilleur terreau pour ma réussite académique: une immense bienveillance et un soutien inconditionnel.

À Ségolène Ta présence à mes côtés a rendu cette étape de ma vie bien plus simple. Tu as fait de ce doctorat une étape de *notre* vie commune en déménageant avec moi aux Pays-Bas, acceptant toutes les incertitudes d'une telle expatriation. Je pourrais te remercier pour de nombreuses autres choses, mais ce déménagement en est un symbole fort. Tu as accepté l'instabilité inhérente à la vie d'un doctorant, incapable de se projeter plus loin que sa soutenance de thèse. Maintenant que cette thèse arrive à son terme, je suis pressé de me projeter sur la suite avec toi.



Abstract

Data breaches and privacy violations have raised global concerns about the protection of personal information. To address these concerns, cryptographic protocols known as Multi-Party Computations (MPC) have been developed to enable multiple parties to jointly compute on their private inputs without revealing them. These protocols have found applications notably in tax fraud detection, healthcare, and machine learning. However, existing MPC protocols remain insufficient for many real-world scenarios, particularly when dealing with high-dimensional or structured data.

This thesis focuses on sparse data; datasets containing mostly zero values, which naturally arise in applications such as recommender systems and healthcare. While plaintext algorithms for sparse data are well established, few cryptographic protocols are optimized for this setting, limiting the practicality of MPC in domains where sparsity is the norm.

The core contributions of this thesis introduce new cryptographic protocols tailored to sparse computations. We propose MPC protocols for secure sparse matrix multiplication, enabling performance gains that make previously impractical applications feasible. We also design new Function Secret Sharing (FSS) schemes able to efficiently aggregate sparse data.

Beyond these protocols, the thesis makes several orthogonal contributions that question key assumptions and practical aspects of privacy-enhancing technologies. We study the role of leakage in sparse computations by analyzing access-pattern leakage in searchable encryption, providing new attacks and statistical insights into its risks. We present Fedivertex, a new graph dataset based on decentralized social media, to benchmark decentralized machine learning. Finally, we evaluate the energy consumption of several privacy-enhancing technologies.

Taken together, these contributions both advance the design of cryptographic protocols for sparse data and provide a broader perspective on the challenges of deploying privacy-preserving computations in practice.



Résumé

Les fuites de données et atteintes à la vie privée soulèvent des inquiétudes croissantes relatives à la protection des informations personnelles. Pour répondre à ces préoccupations, des protocoles cryptographiques appelés calculs multipartites, ou en anglais, Multi-Party Computations (MPC) ont été développés afin de permettre à plusieurs entités d'effectuer conjointement des calculs sur leurs données privées sans les révéler. Ces protocoles ont trouvé des applications notamment dans la lutte contre la fraude fiscale, la santé et l'apprentissage automatique. Cependant, les protocoles MPC existants demeurent insuffisants pour de nombreux scénarios réalistes, en particulier lorsqu'il s'agit de traiter des données de grande dimension ou présentant une structure particulière.

Cette thèse se concentre sur les données éparses; des jeux de données contenant majoritairement des valeurs nulles, qui apparaissent naturellement dans des applications telles que les systèmes de recommandation, la santé ou la génomique. Alors que les algorithmes pour traiter les données éparses en clair sont bien établis, très peu de protocoles cryptographiques sont optimisés pour ce cadre, ce qui limite l'utilisation de MPC dans les domaines où les données éparses sont la norme.

Les contributions centrales de cette thèse introduisent de nouveaux protocoles cryptographiques adaptés à ces données. Nous proposons des protocoles pour la multiplication sécurisée de données éparses, permettant des gains de performance qui rendent possibles des applications jusque-là irréalisables. Nous concevons également de nouveaux algorithmes de Function Secret Sharing (FSS) capables d'agréger efficacement des données éparses.

Au-delà de ces protocoles, la thèse apporte plusieurs contributions transversales qui interrogent des hypothèses fondamentales et des aspects pratiques des technologies de protection de la vie privée. Nous étudions le rôle des fuites d'information dans les calculs sur données éparses en analysant les fuites dans le chiffrement cherchable, en proposant de nouvelles attaques et en apportant un point de vue statistique sur leurs risques. Nous présentons Fedivertex, un nouveau jeu de données de graphes basé sur les réseaux sociaux décentralisés, conçu pour l'évaluation de l'apprentissage automatique décentralisé. Enfin, nous évaluons la consommation énergétique de plusieurs technologies de protection de la vie privée.

Prises dans leur ensemble, ces contributions font progresser la conception de protocoles cryptographiques optimisés pour les données éparses et offrent une perspective plus large sur les défis liés au déploiement pratique des calculs préservant la vie privée.

Contents

Acknowledgements	ii
Abstract	vii
Résumé	viii
Contents	ix
Introduction	xii

Part I: Background

1 Privacy	2
1.1 Threat modelling	2
1.2 Privacy-preserving computations	4
1.3 Privacy preservation	7
2 Multi-Party Computations (MPC)	10
2.1 Secure computations on shared data	10
2.2 Function Secret Sharing (FSS)	14
2.3 Other cryptographic primitives	20
3 Sparse data in Privacy-Preserving Computations	22
3.1 Sparse data	22
3.2 Secure operations on sparse data	24
3.3 Applications to statistics and machine learning	26
4 Leakage in sparse computations: a sense of déjà vu	30
4.1 Necessary leakage in secure sparse operations	30
4.2 Relation to encrypted search	32
4.3 Volume pattern leakage, an acceptable leakage?	38
5 How to securely shuffle?	41
5.1 Introduction	41
5.2 Secure shuffler definition	43
5.3 Comparison of the existing secure shufflers	51
5.4 Applications to privacy-preserving computations	62
5.5 How to choose a secure shuffler?	65
5.6 Perspectives on the literature	70

Part II: Risks of Access Pattern Leakage

6	Leakage-abuse attack against Searchable Encryption	77
6.1	Introduction	77
6.2	Threat model	79
6.3	Score attack	81
6.4	Experimental results	83
6.5	Refined score attack	86
6.6	Attack mitigation	92
7	Revisiting the attacker’s knowledge in SSE attacks	95
7.1	Introduction	95
7.2	Revisiting the notion of similar data	97
7.3	Attack analysis from a data-similarity point of view	102
7.4	Making the “similar-data” assumption unlikely	107
7.5	Do data breaches provide “sufficiently similar” data?	111

Part III: MPC protocols for sparse matrix multiplications

8	Secure sparse multiplications	115
8.1	Introduction	115
8.2	Secure sparse multiplications	116
8.3	Experimental analysis	123
8.4	Mitigating the leakage	127

Part IV: Improving Function Secret Sharing schemes for sparse aggregation

9	Eliminating exponential key growth in PRG-based DPF	133
9.1	Introduction	133
9.2	Honest-majority PRG-based scheme	134
9.3	Key size benchmark	138
10	DDH-based schemes for multi-party Function Secret Sharing	140
10.1	Introduction	140
10.2	DDH-based Distributed Point Function	141
10.3	Key size benchmark	147
10.4	DDH-based Distributed Comparison Function	150
10.5	DDH-based secret encodings and their applications	155
11	Noisy Function Secret Sharing	158
11.1	Introduction	158
11.2	Formalizing the desirable share properties	160
11.3	Noisy Function Secret Sharing	164

11.4 Noisy DPF scheme	171
11.5 Leveraging FSS noise in DP computations.	181

Part V: Towards Real-World Deployments

12 Fedivertex: the Fediverse graph dataset	185
12.1 Introduction.	185
12.2 Dataset presentation	187
12.3 Dataset analysis	192
12.4 Applications.	194
13 Energy Consumption of TLS, Searchable Encryption, and FHE	199
13.1 Introduction.	199
13.2 Methodology	201
13.3 Results.	203
13.4 Orthogonal discussions.	211
13.5 Conclusion.	212
14 Conclusion	214
14.1 Summary of the Contributions	214
14.2 Promising Future Works	215
14.3 Broader Perspectives	216
Acronyms	218
LLM Usage Disclosure	219
References	220

I Introduction

Context

Privacy is about being able to keep certain intimate things to yourself.
 [...] Privacy protects us from unwanted pressures and abuses of power.
 We need it to be autonomous individuals, and for democracies to function
 well we need citizens to be autonomous.

— Carissa Véliz in her book *Privacy is Power* [Vél121]

Privacy is a value intuitively recognized by most people. We all have a general sense of what it means, yet few can provide a precise definition. The attempt to define privacy and its importance has been the subject of long-standing debates, from ancient Greek philosophers [Swa94] to contemporary scholars [Moo08, NW18].

Despite these ontological debates, societies have already taken concrete measures to preserve privacy. As early as 1890, Warren and Brandeis [WB90] wrote a foundational text establishing the *right to privacy*. Their work has since inspired much of the legislations on privacy. In the 1970s, European countries such as France, Germany, and Sweden adopted the first laws on data privacy (e.g., the “Loi Informatique et Libertés” in France). These regulations were enacted years before the invention of the World Wide Web and decades before the rise of “Big Data.” At that time, computer systems were only beginning to spread in administrations and companies, but concerns about the risks of automation had already emerged.

With the development of the Web and its associated digital services, potential privacy risks grew. Companies and governments gained the ability to collect data on millions of individuals, with or without consent. As these surveillance capabilities were deployed, several global scandals followed, including Snowden’s disclosures [Lyo14] and the Facebook-Cambridge Analytica scandal [Wy119]. These events marked the beginning of a new era, in which personal data could be exploited by both companies and governments to serve political or economic interests. Zuboff [Zub19] named it *surveillance capitalism*.

To address the major concerns raised by these scandals, some governments updated their regulations to control the new risks that emerged during the last decades. For example, the European Parliament adopted the General Data Protection Regulation (GDPR) [GDPR16], and the State of California passed the California Consumer Privacy Act (CCPA) [Buk19]. These regulations introduced new rights and obligations; for instance, the GDPR requires data processors to minimize the amount of data collected.

While such regulations represent major progress, a weak link remains in the privacy protection chain: enforcement. For example, although the principle of data minimization is clear, how can we determine the minimal amount of

information necessary for a specific operation? This question has since inspired a vast body of research and engineering projects aimed at building Privacy-Enhancing Technologies (PETs); technologies designed to minimize the information revealed by their users. The goal of this thesis is to contribute to this field by introducing new features to existing technologies and by improving their efficiency.

Scope

Since the 1980s, cryptography has become an essential tool to fight any form of surveillance [Rog15]. It has been first applied to secure communications via encrypted emails or more recently via secure messengers like Signal. However, interpersonal communication is not the only context in which sensitive information may be revealed. With the growing reliance on cloud services, more and more computations are outsourced to third-party servers. These servers often process massive amounts of data. While such computations can provide valuable benefits to society (e.g., detecting tax fraud or improving rare disease diagnosis), they also raise significant privacy concerns.

Unfortunately, “classic” cryptographic schemes used for securing communications are insufficient to protect privacy during outsourced computations. To address this, the cryptography community developed new primitives capable of performing secure computations directly on encrypted data. These primitives are generally grouped under the term *Multi-Party Computation* (MPC). MPC protocols enable multiple parties to jointly compute a function f over secret inputs $\{x_1, \dots, x_n\}$, revealing only the result $f(x_1, \dots, x_n)$ while keeping the inputs private.

Depending on the definition, MPC encompasses a wide variety of primitives, including secret sharing, homomorphic encryption, and zero-knowledge proofs. This thesis focuses in particular on secret-sharing-based protocols. In such schemes, a secret value s is divided into p shares $\{\llbracket s \rrbracket_1, \dots, \llbracket s \rrbracket_p\}$, each distributed to a different party. Individually, the shares reveal no information about s . Over the years, cryptographers have developed many protocols that allow computations on shared secrets [EKR18]. For example, it is possible to sum or sort a list of shared values. During these protocols, the parties exchange randomized messages that, by design, reveal no more information than the final output.

MPC protocols have found applications in diverse domains such as fraud detection [SvHA⁺19], machine learning (ML) [MZ17], healthcare [vDTT⁺24], and aggregate statistics [CB17]. In these applications, input data is highly sensitive, but the outputs are valuable to improve public or private services. MPC thus preserves the utility of the computation output while ensuring the privacy of the inputs. However, existing protocols remain impractical for many real-world applications, particularly when the data involved is *sparse*; i.e., containing a large majority of zero values.

Sparse data arises naturally in many applications. For example, consider a recommender system for a video streaming platform such as YouTube, Netflix,

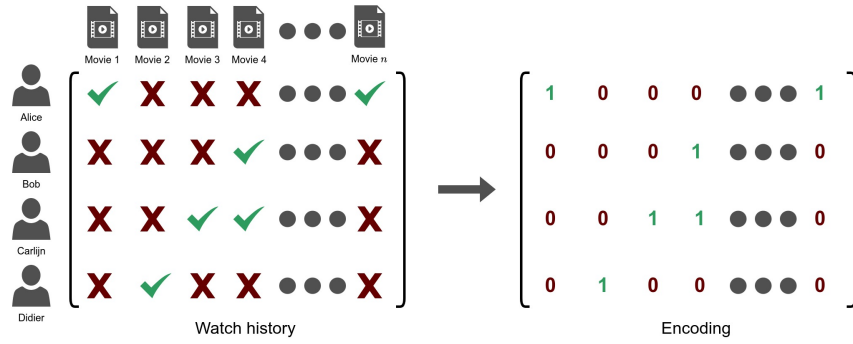


Figure 1 / Simplified example of sparse data in a movie recommender system

or PeerTube. Each user watches only a small fraction of the entire catalog. We can represent a user's watch history as a binary vector, where the i -th entry is 1 if the video has been watched and 0 otherwise. As shown in Fig. 1, this results in very long vectors with mostly zeros. In real-world datasets, sparsity can be very high [LLW17]: around 99% zeros in the Netflix dataset (containing user ratings about 18K movies), 99.9% in Tox21 (containing binary chemical information including the toxicity of various chemical compounds), 99.99% in Bookcrossing (containing user ratings about 271K books), and 99.999% in Flickr (representing user-photo relations on the social network Flickr). With such levels of sparsity, specialized algorithms are essential. Without sparsity-aware techniques, computations become inefficient, or even infeasible if the dataset cannot fit in memory using dense storage.

Storing sparse datasets in a dense format (one value per cell) requires excessive memory and quickly becomes prohibitive in practice. As a result, computations may be infeasible due to performance or memory constraints. Over the past 50 years, many efficient algorithms for plaintext sparse data have been developed [BGS11, Duf77], integrated into widely used software libraries [DHP02], and become indispensable for fields such as ML. In contrast, very few MPC protocols have been designed to exploit sparsity, leaving a major gap in the literature. This gap makes MPC impractical for many valuable applications.

The goal of this thesis is therefore to design new MPC protocols, or extend existing ones, to enable the efficient and privacy-preserving processing of sparse data.

Contributions

This thesis addresses the challenge of processing sparse data in privacy-preserving computations from complementary angles. Together, these contributions provide a broad perspective on the topic, which we categorize into four main themes.

Risks of Access Pattern Leakage Before designing any protocol, it is essential to establish what information can be safely revealed. Some MPC protocols can

trivially support sparse data if data owners reveal the positions of non-zero entries [YCF⁺25]. This information *leakage* corresponds to the access pattern leakage well known in the encrypted search literature.

While prior work on sparse matrix multiplications has suggested avoiding such leakage [YCF⁺25], its concrete risks in sparse computations have not been studied. To address this gap, we examine access pattern leakage in searchable encryption, which (as explained in Chapter 4) can be interpreted as a form of sparse data computation under encryption since the search index can be represented as a sparse matrix. Because many searchable encryption schemes [BHJP14, CGK006] intentionally leak access patterns for efficiency, a large body of work has investigated risk associated to this leakage through leakage-abuse attacks [CGPR15, IKK12, PW16, OK21].

We conceive a new leakage-abuse attack against searchable encryption and revisit existing ones using statistical methods to better assess their practical impact. These results demonstrate the real risks of leakage and highlight the dangers of similar leakage in sparse computations more broadly (i.e., not only in searchable encryption). This motivates the need for MPC protocols that minimize leakage.

MPC protocols for sparse matrix multiplications Second, we design MPC protocols for sparse matrix multiplication that avoid the access pattern leakage whose risks were demonstrated in the previous part. We focus on matrix multiplication because it is a central operation in many ML algorithms, making it crucial for PPML applications on sparse data. For example, sparse matrix multiplications are essential to implement efficient recommender systems. Our contribution builds on prior work in secure sparse vector aggregation [AHI⁺22, BGG⁺22]. Specifically, we extend their sorting-based approach to perform sparse matrix multiplication efficiently over secret-shared sparse data.

Improving Function Secret Sharing schemes for sparse aggregation Following our matrix multiplication protocols, we also address gaps in sparse data aggregation through new Function Secret Sharing (FSS) schemes. FSS generalizes secret sharing to functions: instead of splitting a secret value, a party splits a function into multiple keys. Much of the literature has focused on point functions (i.e., functions that are zero everywhere except at one point) because of their many applications, including private information retrieval, anonymous communication, and PPML. Most relevant to this thesis, recent work [AFK⁺25, BGH⁺25] has highlighted their potential for sparse data aggregation.

While existing two- and three-party FSS schemes are highly efficient, extending them to an arbitrary number of parties is challenging. Some schemes, for example, require keys whose size grows exponentially with the number of parties. To support applications in sparse computations, we design new multi-party schemes that overcome such bottlenecks. Our contributions focus on constraints that directly affect sparse data applications, such as the ability to aggregate shared secrets.

Towards Real-World Deployments Finally, we explore two auxiliary contributions that address practical aspects of real-world deployment. These contributions concern privacy-enhancing technologies in general; not only those operating on sparse data.

On the one hand, we curated a new graph dataset called *Fedivertex*, which models interactions between servers in decentralized social media platforms. This dataset is particularly relevant for benchmarking decentralized PPML algorithms, which until now have mostly been tested on graphs unrepresentative of real-world PPML settings. Because decentralized social media is naturally suited for decentralized PPML, our dataset challenges existing assumptions and enables new applications in this domain. This contribution has an indirect connection to sparse data, since graphs are commonly stored as (usually sparse) adjacency matrices. Moreover, decentralized PPML algorithms can involve sparse data or models. However, our main intention here is not to study sparsity but to highlight new application domains for decentralized PPML.

On the other hand, we analyze the energy consumption of three cryptographic PETs: TLS, searchable encryption, and fully homomorphic encryption. While privacy is often seen as a value to protect at all costs, cryptographic techniques can incur significant performance overheads. As societies try to reduce their carbon footprint, replacing standard technologies with PET-based alternatives could be counterproductive, as it increases energy consumption. We assess the additional energy consumption induced by three PETs, and discuss orthogonal questions to outline promising directions toward sustainable PETs. This contribution concerns PETs in general, with no direct focus on sparse data. Some of the evaluated technologies (e.g., searchable encryption) happen to involve sparse data, but our analysis is broader. Our goal is to initiate a discussion about the environmental costs of PETs as a whole, which naturally includes (but is not limited to) PETs involving sparse data.

Outline of the manuscript

Part I: Background

- Chapter 1 introduces definitions of privacy.
- Chapter 2 presents the MPC primitives used in this thesis.
- Chapter 3 presents the concept of sparse data, existing MPC protocols supporting sparse data, and their applications to ML and statistics.
- Chapter 4 examines the need for leakage in MPC protocols optimized for sparse data, drawing parallels with related cryptographic primitives (notably searchable encryption and Oblivious RAM).
- Chapter 5 formalizes the concept of secure shufflers, surveys the existing cryptographic implementations and their applications to privacy-preserving

computations, and provides guidelines to choose adequate shufflers. This chapter is based on [DHPR25c].

Part II: Risks of Access Pattern Leakage

- Chapter 6 presents a new attack based on access pattern leakage against searchable encryption. While existing attacks all required a partial knowledge of the encrypted data, our attack is the first to achieve high attack accuracy based on similar (but different) attacker's knowledge. This chapter is based on [DHP21].
- Chapter 7 revisits leakage-abuse attacks through a statistical analysis, clarifying the conditions on the attacker's knowledge under which attack accuracy degrades in real-world deployments. This chapter is based on [DLHP25].

Part III: MPC protocols for sparse matrix multiplications

- Chapter 8 designs MPC protocols for secret-shared sparse matrix multiplication using oblivious sorting. These protocols provide performance gains of up to $\times 1000$ and enable applications previously impractical with classic protocols. This chapter is based on [DHPR25d].

Part IV: Improving Function Secret Sharing schemes for sparse aggregation

- Chapter 9 modifies the PRG-based multi-party scheme of [BGI15]. Thanks to the honest-majority assumption, we replace the exponential factor q^p of their key size with a factor $\binom{p}{m+1}$. This chapter is based on [DHPR25b].
- Chapter 10 introduces DDH-based multi-party schemes with $O(\sqrt[3]{N})$ key size, improving over existing $O(\sqrt{N})$ schemes. In practice, our construction yields keys up to $10\times$ smaller. This chapter is based on [DHPR25a].
- Chapter 11 introduces noisy FSS, a relaxation preserving the standard privacy guarantees but allowing noise in the output. We build a first noisy scheme adapted from [CBM15] and present how to leverage the noise in differentially private computations. This chapter is based on [DMH⁺25].

Part V: Towards Real-World Deployments

- Chapter 12 presents Fedivertex, a new graph dataset modeling interactions between servers on decentralized social media. This dataset is designed to benchmark decentralized ML. This chapter is based on [DC25b].
- Chapter 13 assesses the energy consumption overhead of three cryptographic PETs. This chapter is based on [DPP25].
- Chapter 14 concludes the thesis with a summary of contributions and perspectives for future work.

Publications

During my Ph.D., I had the chance to collaborate with different researchers, including my supervisors, and other fellow Ph.D. students. These collaborations resulted in 9 publications (or preprints) that form the basis of this thesis:

- (In Proceedings) [DHP21] Marc Damie, Florian Hahn, and Andreas Peter. A Highly Accurate Query-Recovery Attack against Searchable Encryption using Non-Indexed Documents. IN: *30th USENIX Security Symposium (USENIX Security 21)*, 2021. ISBN: 978-1-939133-24-3.

Contribution: I am the main author.

- (In Proceedings) [DLHP25] Marc Damie, Jean-Benoist Leger, Florian Hahn, and Andreas Peter. Revisiting the Attacker's Knowledge in Inference Attacks Against Searchable Symmetric Encryption. IN: *Applied Cryptography and Network Security*, 2025. ISBN: 978-3-031-95764-2.

DOI: [10.1007/978-3-031-95764-2_15](https://doi.org/10.1007/978-3-031-95764-2_15)

Contribution: I am the main author. Jean-Benoist Léger contributed his statistical expertise to refine the approach and improve several proofs.

- (Preprint) [DHPR25c] Marc Damie, Florian Hahn, Andreas Peter, and Jan Ramon. How to Securely Shuffle? A survey about Secure Shufflers for privacy-preserving computations, 2025.

DOI: [10.48550/arXiv.2507.01487](https://doi.org/10.48550/arXiv.2507.01487)

Contribution: I am the main author.

- (Preprint) [DHPR25d] Marc Damie, Florian Hahn, Andreas Peter, and Jan Ramon. Secure Sparse Matrix Multiplications and their Applications to Privacy-Preserving Machine Learning, 2025.

Contribution: I am the main author.

- (In Proceedings) [DHPR25b] Marc Damie, Florian Hahn, Andreas Peter, and Jan Ramon. Eliminating exponential key growth in PRG-based Distributed Point Functions. IN: *European Symposium on Research in Computer Security Workshops*, 2025.

Contribution: I am the main author.

- (In Proceedings) [DHPR25a] Marc Damie, Florian Hahn, Andreas Peter, and Jan Ramon. DDH-based schemes for multi-party Function Secret Sharing. IN: *Nordic Conference on Secure IT Systems (NordSec)*, 2025.

Contribution: I am the main author.

- (Preprint) [DMH⁺25] Marc Damie et al. Noisy Function Secret Sharing and its applications to Differential Privacy, 2025.

Contribution: I am the main author. I developed the idea and designed the schemes. Federico Mazzone contributed by improving or/and writing the proofs and definitions.

- (Preprint) [DC25b] Marc Damie and Edwige Cyffers. Fedivertex: a Graph Dataset based on Decentralized Social Networks for Trustworthy Machine Learning, 2025.

DOI: 10.48550/arXiv.2505.20882

Contribution: I initiated the project, developed the crawlers, managed all legal procedures prior deployment, administered the crawlers, and implemented several analysis scripts. Edwige Cyffers worked with me from the start to transform the idea into a contribution for the ML community. She refined the scope, suggested improvements for the crawler, implemented some data analysis, and wrote most of the dataset paper.

- (Preprint) [DPP25] Marc Damie, Mihai Pop, and Merijn Posthuma. Energy Consumption of TLS, Searchable Encryption and Fully Homomorphic Encryption, 2025.

DOI: 10.48550/arXiv.2508.04583

Contribution: I am the main author. I proposed the project, designed the methodology, implemented half of the experiments, and wrote the paper. Mihai Pop and Merijn Posthuma contributed as part of their Bachelor's theses [Pop25,Pos25] by implementing the experiments on SSE and HTTPS.

Contributions unrelated to the thesis In addition to the papers included in this thesis, I also contributed to other projects:

- [DDHP22] Marco Dijkslag, Marc Damie, Florian Hahn, and Andreas Peter. Passive Query-Recovery Attack Against Secure Conjunctive Keyword Search Schemes. IN: *Applied Cryptography and Network Security*, 2022. ISBN: 978-3-031-09234-3.

DOI: 10.1007/978-3-031-09234-3_7

Contribution: I co-supervised Dijkslag's Master's thesis [Dij21].

- [vDDH25] Bram van Dartel, Marc Damie, and Florian Hahn. Evaluating Membership Inference Attacks in heterogeneous-data setups. IN: *Security in Machine Learning Applications (co-located with ACNS)*, 2025.

Contribution: I co-supervised Bram van Dartel for his Master's thesis [vDar24] and adapted his thesis for publication.

- [GGD25] Nilufer Guldali, Chakshu Gupta, and Marc Damie. IoTLS 2.0: How far has IoT industry come in securing communications with TLS?, 2025.

Contribution: I co-supervised Nilufer Guldali for her Bachelor's thesis [Gu125] and adapted her thesis for publication.

Part I

Background

This first part provides wide background knowledge that is necessary to understand the thesis. We start with general knowledge about privacy and cryptography and gradually dive into more specific concepts, notably existing cryptographic works involving sparse data. We conclude this part with a survey on secure shufflers, a cryptographic protocol that has found many applications in privacy-preserving computations including on sparse data.

1 Privacy

Discussions about privacy date back to Ancient Greece [Swa94] and have long been part of political and philosophical debates. In 1890, [WB90] introduced the *right to privacy*, a key idea that shaped many modern privacy laws.

Recently, The development of large-scale computer systems has changed the privacy debate. Companies and governments can now collect vast amounts of personal data. This uncontrolled collection and use of data has raised concerns worldwide and led to regulations such as the California Consumer Privacy Act (CCPA) [Buk19] and the European Union’s General Data Protection Regulation (GDPR) [GDPR16]. This thesis supports these regulations by developing methods that allow useful computations without revealing private information.

Even though privacy has been discussed for centuries, there is still a lot of research to build a single holistic definition [Moo08,NW18]. As reference, we remind the definition from Parker [Par73]:

Definition 1.1 (Privacy [Par73]). Privacy is control over when and by whom the various parts of us can be sensed by others.

This thesis does not aim to settle philosophical questions about privacy; it “simply” designs cryptographic tools that enhance privacy in digital services. In computer security, privacy is often defined in simpler, more practical terms so that it can be measured and analyzed. Our work focuses on privacy for computations on personal data. The rest of this chapter explains what privacy means in this context.

1.1 Threat modelling

Definition 1.1 gives a constructive definition of privacy: control over who *can* sense something. In contrast, computer security research often takes a defensive approach, focusing on who *cannot* access data. Here, privacy is defined in terms of the threats a protocol must resist, ensuring that data remains private or confidential against those threats. To compare protocols, the research community has established a standard terminology for threat modeling.

Threat models characterize agents or behaviors that threaten a desirable security property of a protocol (e.g., correctness or privacy). When defining a secure protocol, it is necessary to specify a threat model against which the security properties are guaranteed. In other words, the desirable property of a protocol holds as long as no adversary is more powerful (i.e., has more knowledge or computational power) than what is defined in the threat model. This subsection reminds the terminology commonly used in security literature [EKR18].

Agent behavior Threat modelling distinguishes three types of agents: *honest agents* (following the protocol), *semi-honest agents* (who follow the protocol but try to infer some private information passively), and *malicious agents* (who do not follow the protocol and can perform actions to disrupt the protocol and gain additional information). A malicious agent can notably be a corrupted device or server. While semi-honest agents may compromise only the privacy of a computation, malicious agents also threaten the correctness of the computation.

Majority behavior Secure protocols typically guarantee data privacy against a proportion of dishonest agents (i.e., malicious or semi-honest depending on the protocols). For instance, Clarion [EB22] is secure as long as one out of p servers (involved in the protocol) is honest. On the other hand, RPM [LK23] is secure as long as there is a majority of honest servers involved in the protocol.

More formally, secure protocols are characterized by the tuple (m, p) , with p the number of parties involved and m the maximum number of parties corrupted by the adversary. The security of an (m, p) -secure protocol is guaranteed only if the number of dishonest parties is below than or equal to m . The literature creates two main sub-categories of protocols: honest-majority protocols ($m < p/2$) and dishonest-majority protocols ($m \geq p/2$).

Collusion Two agents sharing their information (outside the protocol) are referred to as “colluding” agents. Threat models usually consider a unique adversary composed of all the corrupted parties (to be as conservative as possible in the security analysis).

Computational power Threat models also characterize the adversary computational power. Assuming a computationally unbounded adversary provides maximum security, referred to as information-theoretic security. Even an adversary with infinite computational power cannot break the security of information-theoretic schemes.

Unfortunately, it is not always possible to build *efficient* information-theoretic protocols (e.g., secure communication channels [Mau99]), so cryptographic protocols commonly rely on computational hardness assumptions. In such *computationally secure* protocols, an attacker would be able to break the scheme if and only if they manage to solve a computational problem assumed to be hard. This thesis will notably use the PRG, DDH, and LWE assumptions [BBC⁺13].

Due to the recent advances in quantum computing, the cryptography community has developed an interest in “quantum-safe” hardness assumptions (like LWE) that also hold against adversaries with quantum computational power. On the contrary, problems like DDH or RSA can be efficiently solved by quantum computers. The focus of this thesis is not specifically post-quantum cryptography, so we will not discuss this topic further.

Global adversary While most secure protocols restrict their threat model to the agents actively involved in the protocol, some works also include in their threat model a “global adversary” able to observe the network traffic of the agents. This concept is motivated by anonymous communication protocols that are sensitive to traffic analysis attacks [MD05]. In such protocols, a global adversary is typically a state or an internet service provider.

1.2 Privacy-preserving computations

Defining privacy-preserving computations requires more than just specifying a threat model. While threat models describe the adversary’s capabilities, they do not capture precisely what information must remain hidden. To address this, researchers rely on formal privacy definitions that specify what an adversary is prevented from learning. Two main approaches have emerged, originating from different research communities: cryptography and differential privacy.

1.2.1 Cryptography

In cryptography, privacy definitions often rely on the ideal-real paradigm. Consider a protocol π that computes a function $f(x)$ on input x . The protocol π is said to be private if it does not reveal any more information about x than what would be revealed by a trusted curator who computes $f(x)$ and only outputs the result. Here, the execution of π corresponds to the real world, while the trusted curator represents the ideal world. In other words, the protocol is private if no adversary can learn more from interacting with π than what can be inferred from the output $f(x)$ alone.

We can formalize this intuition as follows:

Definition 1.2 (Privacy via Ideal-Real Paradigm). Let $f : \mathcal{X}_1 \times \dots \times \mathcal{X}_d \rightarrow \mathcal{Y}$ be a function and π a protocol computing $f(x_1, \dots, x_d)$ for any $x_i \in \mathcal{X}_i$.

The protocol π is *private* if, for any adversary interacting with the real-world execution of π , there exists a simulator interacting only with the output $f(x)$ in the ideal world such that the adversary’s view (i.e., everything the adversary knows and observes) in the real world is computationally indistinguishable from the simulator’s view in the ideal world.

Definitions based on simulators [Lin17] are widely used in cryptography. By proving the existence of a simulator that can reproduce an adversary’s view using only the protocol’s output, one shows that the protocol leaks no additional information. Our contributions build upon such definitions to prove the security guarantees of the protocols we introduce in this thesis.

1.2.2 Differential Privacy (DP)

While cryptographic definitions focus on limiting what an adversary can learn during protocol execution, information can still be leaked through the output itself. For example, if n voters participate in an election and the result is unanimous, the output reveals every voter's choice with certainty.

To address such information leakage from public outputs, differential privacy (DP) offers a complementary framework. Introduced by Dwork [Dwo06], DP quantifies the privacy loss induced by any computation with public outputs:

Definition 1.3. (ϵ -Differential Privacy) A randomized algorithm $\mathcal{M} : \mathbb{X}^N \rightarrow \mathbb{Z}$ is ϵ -differentially private if for any datasets D and D' that differ on a single element and for all subsets S of $\mathbb{Z}$:

$$\Pr[\mathcal{M}(D) \in S] \leq e^\epsilon \cdot \Pr[\mathcal{M}(D') \in S]$$

The space $\mathbb{X}$ corresponds to the secret value space (i.e., each data owner P_i owns a private value $x_i \in \mathbb{X}$). The parameter $\epsilon \in [0, \infty)$ controls the privacy loss: smaller ϵ implies more privacy.

Definition 1.3 focuses on datasets X and X' differing on a single element [DR14]. This concept has link with a highly important concept in DP: sensitivity. Sensitivity bounds the influence of a single input on the algorithm output. Sensitivity can be expressed using different metrics, notably the l_2 norm:

Definition 1.4. For any function $f : \mathcal{X}^n \rightarrow \mathcal{Y}$, the l_2 -sensitivity Δ_2 is:

$$\Delta_2 = \max_{\substack{X, X' \in \mathcal{X}^n \\ X, X' \text{ differ on one element}}} \|f(X) - f(X')\|$$

Definition 1.3 is referred to as “pure” or “exact” DP; it corresponds to the historical DP definition. Pure DP is sensitive to highly unlikely events [SU21] that can complicate the analysis. Instead of pure DP, recent DP works commonly rely on approximate DP; a variant allowing failure for highly unlikely events. In this variant, the failure rate is defined by a second parameter δ :

Definition 1.5 ((ϵ, δ) -Differential Privacy). A randomized algorithm $\mathcal{M} : \mathbb{X}^N \rightarrow \mathbb{Z}$ is (ϵ, δ) -differentially private if for any datasets D and D' that differ on a single element and for all subsets S of $\mathbb{Z}$:

$$\Pr[\mathcal{M}(D) \in S] \leq e^\epsilon \cdot \Pr[\mathcal{M}(D') \in S] + \delta$$

To simplify the mathematical analysis, the literature has described other DP variants such as Concentrated DP. Chapter 11 uses this variant, so we detail its definition below:

Definition 1.6 (Concentrated DP [BS16]). A randomized algorithm $\mathcal{A} : \mathcal{X}^n \rightarrow \mathcal{Y}$ is $\frac{1}{2} \varepsilon^2$ -concentrated DP if, for any datasets $X, X' \in \mathcal{X}^n$ differing of a single element, we have

$$\sup_{\alpha \in (1, \infty)} \frac{1}{\alpha} D_{\alpha}(M(X) || M(X')) \leq \frac{1}{2} \varepsilon^2$$

with D_{α} the Rényi divergence.

Privacy-utility trade-off To reduce the ε , DP works commonly add noise during the computation [DR14]. This added noise creates a privacy-accuracy trade-off (sometimes called privacy-utility in the literature). The goal of DP works is to optimize this trade-off.

Definition 1.5 is called “central” DP because it was originally intended for setups with a trusted curator collecting all data centrally and publishing a privacy-preserving output. Central DP provides the optimal trade-off because the trusted curator can simply add a small noise at the end of the computation. However, it requires a trusted curator to process the data, which is not necessarily possible.

If there is no central trusted curator, local differential privacy (LDP) is an option that requires adding more noise but privatizes the data before it leaves the data owner (see Fig. 1.8). In particular, LDP introduces a local randomizer $\mathcal{R} : \mathbb{X} \rightarrow \mathbb{Y}$.

Definition 1.7. The local randomizer $\mathcal{R}$ is (ε, δ) -local differentially private if for any two data samples $x, x' \in \mathbb{X}$ and all subset S of $\mathbb{Y}$:

$$\Pr[\mathcal{R}(x) \in S] \leq e^{\varepsilon} \cdot \Pr[\mathcal{R}(x') \in S] + \delta$$

If $\mathcal{R}$ is (ε, δ) -local differentially private, then $\mathcal{M}(x_1 \dots x_n) = (\mathcal{R}(x_1) \dots \mathcal{R}(x_n))$ is (ε, δ) -DP.

The local randomizer used in LDP induces noisier results than central DP, but LDP makes no security or trust assumptions; less trust requires higher noises.

Several papers have proposed strategies to “amplify the privacy”; i.e., reducing the amount of noise needed to reach a certain privacy level. These solutions include iterative protocols [FMTT18], decentralization [CB22], random check-ins [BKM⁺20] and *shuffling* [EFM⁺19] (detailed in Section 5.4.3).

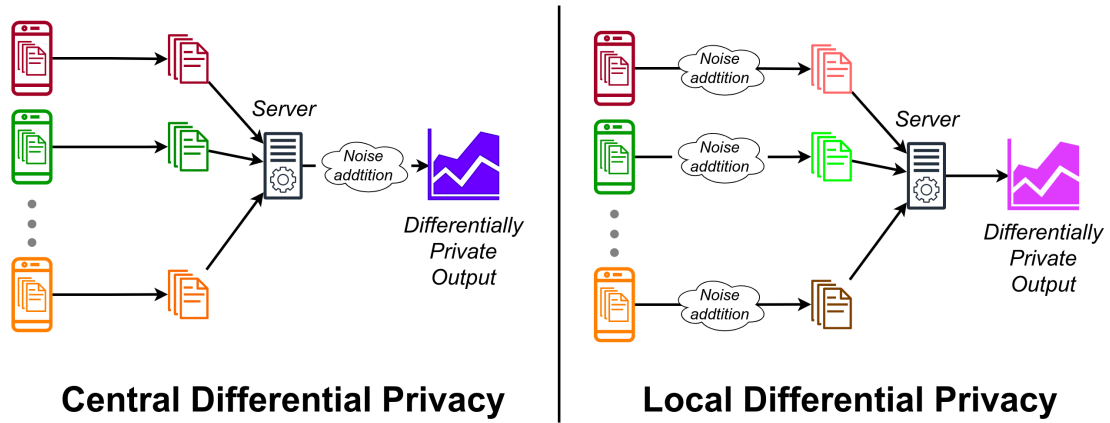


Figure 1.8 / Simplified representation of central and local DP deployments

Alternative definitions Over the years, several privacy definitions related to or inspired by differential privacy have been proposed, including k -anonymity [Swe02], t -closeness [LLV07], Pufferfish [KM14], and the Bayes security measure [CCPT23]. Despite these alternatives, differential privacy remains the gold standard in the privacy community due to its strong mathematical foundation and ability to support rigorous proofs of complex properties.

1.3 Privacy preservation

Building upon these privacy definitions, researchers have developed algorithms and protocols that satisfy them. This section presents three main approaches to privacy preservation: the cryptographic approach, the differential privacy approach, and hybrid methods combining both.

1.3.1 Cryptographic approach

Encryption schemes are defined by the algorithms (KeyGen, Enc, Dec). The key generation algorithm $\text{KeyGen}(1^\lambda)$ is a randomized algorithm taking as input a security parameter λ and outputting one key (for symmetric encryption) and two keys (for public-key encryption). The encryption function Enc takes as input a key k_{Enc} and a plaintext m : $\text{Enc}(k_{\text{Enc}}, m) = c$. The decryption function Dec takes as input a key k_{Dec} and a ciphertext c : $\text{Dec}(k_{\text{Dec}}, c) = m$. When an agent possesses the correct keys, we have $\text{Dec}(k_{\text{Dec}}, \text{Enc}(k_{\text{Enc}}, m)) = m$.

Cryptographers usually rely on cryptographic schemes (or related schemes like secret sharing presented in Chapter 2) to preserve privacy. The intuition is to perform the computation on encrypted (or secret-shared data) and to only reveal the final output. Through encryption, it is then possible to only reveal the output, as expected in Definition 1.2.

Security parameter The security parameter λ is a fundamental concept in cryptography that quantifies the security level in a cryptographic scheme; determining notably the size of the keys. A PPT (Probabilistic Polynomial-Time) adversary is a computationally bounded attacker whose runtime is polynomial in λ . Secure cryptographic schemes must ensure that the probability of a PPT adversary to break it is negligible in λ ; i.e., it decreases faster than any inverse polynomial function of λ .

1.3.2 DP approach

DP works commonly add noises to enforce a certain level of privacy [DR14]. The noise addition is performed via (sub-)algorithms called “mechanisms.” Our work is particularly interested in the Gaussian mechanism described in Proposition 1.9. This proposition details the relationship between the privacy level ε and σ^2 , when an algorithm output is noised using a Gaussian with variance σ^2 .

Proposition 1.9 (Gaussian Mechanism [DR14]). Let $f : \mathcal{X}^n \rightarrow \mathbb{R}^d$ be an arbitrary d -dimensional function and Δ_2 be its l_2 sensitivity. Let $F : \mathcal{X}^n \rightarrow \mathbb{R}^d$ be a randomized algorithm such that $\forall x, F(x) \leftarrow f(x) + e$ with $e \stackrel{R}{\leftarrow} \mathcal{N}(0, \sigma^2 I_d)$, with I_d the identity matrix of size d .

For any $\varepsilon \in (0, 1)$, if $\sigma^2 \geq \frac{2\Delta_2 \ln(1.25/\delta)}{\varepsilon^2}$, F is (ε, δ) -DP.

Unfortunately, the Gaussian mechanism is only compatible with classic Gaussian noises in $\mathbb{R}$. As observed in [KLS21], private computations relying on cryptographic protocols require discretizing the noise distribution, because the operations are commonly performed in Abelian groups (e.g., $\mathbb{F}_q$). This discretization has an important impact on the distribution properties. For example, while the sum of continuous Gaussian distributions is a continuous Gaussian, the sum of discrete Gaussian distributions is not exactly a discrete Gaussian [KLS21].

Thus, [CKS22, KLS21] redeveloped the DP results for discrete Gaussian distributions. The continuous and discrete Gaussian distributions provide close (but not equal) privacy guarantees [KLS21]. For our work, we are particularly interested in the privacy guarantees provided by a sum of discrete Gaussians. Proposition 1.10 reminds a key result of [KLS21]. In this proposition, they rely on an alternative DP definition: concentrated DP (see Definition 1.6). This relaxed definition provided a better framework to obtain a tighter privacy bound.

Proposition 1.10 (“Sum of discrete Gaussians” mechanism [KLS21]). Let $f : \mathcal{X}^n \rightarrow \mathbb{R}^d$ be an arbitrary d -dimensional function and Δ_p be its l_p sensitivity. Let $F : \mathcal{X}^n \rightarrow \mathbb{R}^d$ be a randomized algorithm such that $\forall x, F(x) \leftarrow f(x) + \sum_{i=1}^p e_i$, with each e_i drawn independently of a discrete Gaussian $\mathcal{N}_{\mathbb{Z}}(0, \sigma^2 I_d)$.

For any $\sigma \geq \frac{1}{2}$, F is $\frac{1}{2} \varepsilon^2$ -concentrated DP for

$$\varepsilon = \min \left\{ \begin{array}{l} \sqrt{\frac{\Delta_2^2}{p\sigma^2} + \frac{1}{2} \tau d}, \\ \sqrt{\frac{\Delta_2^2}{p\sigma^2} + 2 \frac{\Delta_1}{\sqrt{N\delta}} \tau + \tau^2 d}, \\ \frac{\Delta_2}{\sqrt{p}\sigma} + \frac{1}{2} \tau \sqrt{d}, \end{array} \right\}$$

with $\tau = 10 \cdot \sum_{k=1}^{p-1} e^{2\pi^2 \sigma^2 \frac{k}{k+1}}$

Privacy parameter To ensure strong privacy, ε is usually chosen to be below 1 [ACG⁺16]. The parameter $\delta \in [0, 1)$ relaxes the definition and allows for rare cases where the privacy loss would be higher than ε . To ensure strong privacy, the probability δ should be below $1/|D|$ (with $D \in \mathbb{X}^N$, a dataset) [ACG⁺16].

1.3.3 Combined Approaches

Finally, some works like [KLS21] combine cryptographic schemes with DP techniques to build protocols that offer an optimal trade-off between privacy and utility. Cryptographic protocols enable the secure implementation of the central DP model without relying on a trusted third party; the protocol itself simulates the role of a trusted curator while ensuring privacy through cryptographic guarantees. This hybrid approach leverages the strengths of both fields making it particularly suited for practical applications involving sensitive data.

2 Multi-Party Computations (MPC)

To perform computations on private data, cryptographers have designed a wide range of protocols known as Multi-Party Computation (MPC). MPC protocols allow several parties to jointly compute a function over their inputs while keeping those inputs secret, without the need for a trusted third party. This thesis builds on several well-established MPC protocols and introduces new ones tailored to operations on sparse data. This chapter then provides an overview of the MPC literature, with particular emphasis on Function Secret Sharing (FSS), as three of the contributions presented in this thesis are directly related to this primitive.

2.1 Secure computations on shared data

Depending on the context and definitions, MPC can encompass a variety of cryptographic primitives. Among these, secret sharing [Sha79] is one of the most fundamental and widely used. This section introduces secret sharing and the operations that can be performed on shared data. The protocols developed in this thesis rely primarily on these techniques. For completeness, Section 2.3 finishes this chapter with additional primitives that are sometimes used in MPC protocols and that are mentioned in other chapters of this thesis.

2.1.1 Secret sharing

Secret sharing enables a secret to be split into multiple *shares*, which are distributed among different parties in such a way that the secret can be reconstructed only when a sufficient number of shares are combined. This simple but powerful idea underlies many MPC protocols, as it allows computations to be carried out directly on shares without revealing the underlying values.

A *share* is a value that, by itself, reveals no information about the underlying secret. When all (or a sufficient subset of) shares are gathered, they can be combined to reconstruct the secret. Formally, a secret value x is considered *shared* when each party P_j holds a corresponding share $\llbracket x \rrbracket_j$.

A secret sharing scheme is defined by two algorithms:

- $\text{Share}(x) : \mathbb{G} \rightarrow \mathbb{G}^p$ generates p shares $\{\llbracket x \rrbracket_1, \dots, \llbracket x \rrbracket_p\}$ of a secret value x .
- $\text{Decode}(\llbracket x \rrbracket_1, \dots, \llbracket x \rrbracket_t) : \mathbb{G}^t \rightarrow \mathbb{G}$ reconstructs x from any set of at least t valid shares.

Such schemes are typically characterized by the tuple (t, p) , where t is the minimum number of shares required for reconstruction, and p is the total number of shares generated. A (t, p) -secret sharing scheme is secure against up to $t - 1$ colluding (corrupted) parties; i.e., $m = t - 1$.

Standard secret sharing schemes do not rely on a security parameter because they achieve *information-theoretic security*; meaning that even an adversary with unbounded computational power learns nothing from fewer than t shares.

Additive secret sharing In an additive secret sharing scheme, a secret $x \in \mathbb{G}$ is split into p shares $\{\llbracket x \rrbracket_1, \dots, \llbracket x \rrbracket_p\}$ such that $x = \sum_{j=1}^p \llbracket x \rrbracket_j$.

The shares are chosen uniformly at random (under the constraint mentioned above), ensuring that any subset of fewer than p shares reveals no information about x .

Shamir's secret sharing Shamir's scheme is based on polynomial interpolation. To share a secret x with threshold t , the dealer samples a random polynomial r of degree $t - 1$ such that $r(0) = x$, and gives share $r(j)$ to party P_j . Any set of t or more shares can reconstruct $r(0)$ via Lagrange interpolation, while fewer than t shares reveal nothing.

Replicated secret sharing This last scheme generalizes additive secret sharing to support arbitrary decoding threshold t . Each share is itself a set of additive shares given to different parties. The shares are distributed so that every subset of $t - 1$ parties collectively misses at least one additive component of the secret, preventing reconstruction.

Share conversion As each scheme has its own strengths and weaknesses, some MPC protocols also switch between schemes [CDI05] to always use the most appropriate scheme for the desired operation. Protocols relying on share conversions are called “mixed-mode” protocols. They are appreciated in recent works like [MR18, PSSY21] because they enable a wide range of operations, but they induce significant conversion costs.

2.1.2 Basic operations

When p parties share two values x and y (i.e., each party P_i holds the shares $\llbracket x \rrbracket_i$ and $\llbracket y \rrbracket_i$), they can execute *secure algorithms*; protocols that compute functions on these shared values without revealing them. The exact implementation of these operations depends on the underlying secret-sharing scheme; we therefore do not present their detailed protocols here. For a comprehensive description of these primitives, we refer the reader to [EKR18]. Instead, we list the operations commonly supported and briefly comment on their efficiency.

Notations In this thesis, any operation involving shared values corresponds to a secure variant; e.g., $\llbracket a \rrbracket + \llbracket b \rrbracket$ is the secure addition of a and b and $\llbracket \text{sort}(l) \rrbracket$ is the oblivious sort of l . Secure algorithms are executed jointly by all shareholders.

Arithmetic operations In common schemes such as additive and Shamir’s secret sharing, addition can be performed locally:

$$\llbracket x + y \rrbracket_i = \llbracket x \rrbracket_i + \llbracket y \rrbracket_i.$$

This makes addition (and subtraction) extremely efficient, as no interaction between parties is required.

Multiplication, in contrast, typically requires communication among the parties, making it more expensive. Division is often even costlier, sometimes requiring approximations, which can further impact efficiency.

Comparison operations Comparison operations (e.g., testing whether $x < y$) are also possible in MPC, but they are generally more communication-intensive than multiplication. They therefore increase both the communication cost and the number of communication rounds, and must be used carefully in protocol design.

2.1.3 Comparing MPC protocols

MPC protocols are usually compared based on *security* and *efficiency*.

The security dimension considers the threat model in which a protocol is secure and the hardness assumptions conditioning its guarantees. While the strongest security is achieved when a protocol remains secure against $p - 1$ malicious parties, such guarantees can come at a high performance cost. As a result, many protocols are designed for weaker adversary models, such as *honest-but-curious* adversaries or the assumption of an honest majority of parties, to achieve better efficiency.

Efficiency can be evaluated using the following criteria:

- **Communication cost:** volume of data exchanged between the parties.
- **Computation cost:** amount of computation performed by the parties.
- **Round complexity:** the number of communication rounds required.
- **Storage cost:** memory needed to execute the protocol.

In the literature, communication cost and round complexity are often the most regarded factors, because they directly influence execution time in practical deployments. However, this thesis pays particular attention to computation and storage costs, since these can become prohibitive when operating on sparse data, rendering an MPC protocol impractical even when its communication complexity is acceptable.

2.1.4 Matrix multiplication

As Chapter 8 presents novel MPC protocols for multiplying sparse matrices, we first briefly review the literature on secure matrix multiplication.

Let Z be the product of an $n_a \times n_b$ matrix X and an $n_b \times n_c$ matrix Y . The result Z is an $n_a \times n_c$ matrix whose entries are given by

$$Z_{ij} = \sum_{k=1}^{n_b} x_{ik} \cdot y_{kj}, \quad \forall i \in \{1, \dots, n_a\}, j \in \{1, \dots, n_c\}. \quad (2.1)$$

This definition applies to all matrices, but the concrete implementation of the multiplication may vary depending on the structure of the matrices involved (e.g., vectors, diagonal matrices, or triangular matrices). Most existing MPC works [CKR⁺20, MR18, MZ17, MG23, PS20, WTB⁺21] focus on *standard* matrices, without exploiting special structural properties. This subsection then concentrates on secure multiplication for standard matrices, and Chapter 3 will introduce secure operations (including multiplication) that leverage the data sparsity.

Secure multiplications on generic matrices Software libraries usually implement matrix multiplication via a triple nested loop, resulting in an $O(n_a n_b n_c)$ cost. While this cost has long been considered the lower bound for matrix multiplication, several works described algorithms with a sub-cubic cost [CW87]. [DLL⁺19] described an MPC protocol to execute this matrix multiplication with sub-cubic cost. These algorithms have constant factors limiting their practical interest (*even in plaintext*), so software libraries favor algorithms with a cubic cost.

The cubic-cost matrix multiplication (i.e., the naive implementation of Equation (2.1)) can easily be imported to MPC. However, recent works [CKR⁺20, MR18, MZ17] [MG23, PS20, WTB⁺21] described more efficient protocols under honest or dishonest majority assumptions. These works reduce the communication costs, but the computation cost remains asymptotically cubic. Moreover, the storage costs remain asymptotically equivalent to the plaintext algorithm, so they cannot support larger matrices than the plaintext algorithm.

Dishonest majority Beaver’s multiplication triplets are a popular building block to speed up secure multiplications via preprocessing steps [EKR18]. [MZ17] generalized this concept to matrices: matrix multiplication triplets. These triplets make the communication costs proportional to the input and output size. For example, the multiplication of an $n_a \times n_b$ matrix with an $n_b \times n_c$ matrix requires $O(n_a n_b + n_b n_c + n_a n_c)$ communications (instead of $O(n_a n_b n_c)$ with the naive algorithm). This approach reduces the asymptotic cost for matrix-matrix multiplications, but the communication costs of the vector-vector and matrix-vector multiplications remains asymptotically equal to the naive costs. Matrix multiplication triplets with malicious security were later described [CKR⁺20, MG23].

Honest majority Using the honest majority assumption, several recent works [KPPS21, LZLR23, MR18, PS20, WTB⁺21] designed protocols with a communication cost proportional to the output size. Hence, a square matrix multiplication only requires $O(n_a^2)$ communications, and a vector multiplication $O(1)$ communications. Before these recent protocols, De Hoogh [Hoo12] had already described a matrix

multiplication with equivalent costs based on Shamir’s secret sharing (SSS). The recent protocols [KPPS21, LZLR23, MR18, WTB⁺21] are more complex to understand than the SSS-based algorithm of [Hoo12] because they rely on share conversion protocols. Moreover, these protocols only support 3 parties, while the SSS-based solution supports an arbitrary number of parties. Nevertheless, the SSS-based solution only has honest-but-curious security, while [KPPS21, LZLR23, PS20, WTB⁺21] support malicious security.

To understand the gain provided by the honest-majority assumption, let us detail the optimized vector multiplication under SSS [Hoo12]. In SSS, each shared value is represented using a polynomial of degree t . To multiply two shared scalars in SSS, the parties locally multiply their local shares (obtaining a polynomial of degree $2t$) and then execute a degree reduction protocol (via a “resharing” protocol). For vector multiplication, we can delay the degree reduction and add all local multiplication results together. The addition is possible because all local multiplication results are shares of equal polynomial degree (i.e., $2t$). Then, a single degree reduction is done on the sum result.

2.1.5 Sorting and shuffling

“Oblivious sorting” designates MPC protocols for sorting shared values without revealing information about the values. Sorting networks [HICT14] are straightforward oblivious sorting solutions, as their execution is data-independent. Their cost is $O(n \log^2 n)$ for a list of size n . The oblivious radix sort [HICT14] is the state-of-the-art honest-majority protocol, with its optimal complexity of $O(n \log n)$ (and low constant factors). Recent papers optimized this protocol in a three-party setting [AF0⁺21, AHI⁺22].

“Oblivious shuffling” protocols randomly permute lists of secret values without revealing the permutation. [LWZ11] proposed an honest-majority protocol with linear costs, used in many recent protocols [AF0⁺21, AHI⁺22, HICT14]. A recent work [LK23] also described dishonest-majority protocols, but with supra-linear costs.

2.2 Function Secret Sharing (FSS)

[BGI15] extended the concept of secret sharing from values to *functions*, introducing *Function Secret Sharing* (FSS). An FSS scheme allows a function f to be shared among multiple parties in such a way that each party holds a key that can be used to obtain shares of $f(x)$ without interaction between the parties.

FSS has been applied in a wide range of applications, including private information retrieval (PIR) [GI14], anonymous communication systems [CBM15, NSD22], blockchain [ZYP24], privacy-preserving machine learning [KPM24, RTPB22], MPC precomputations [BGI19], and Oblivious RAM [DS17].

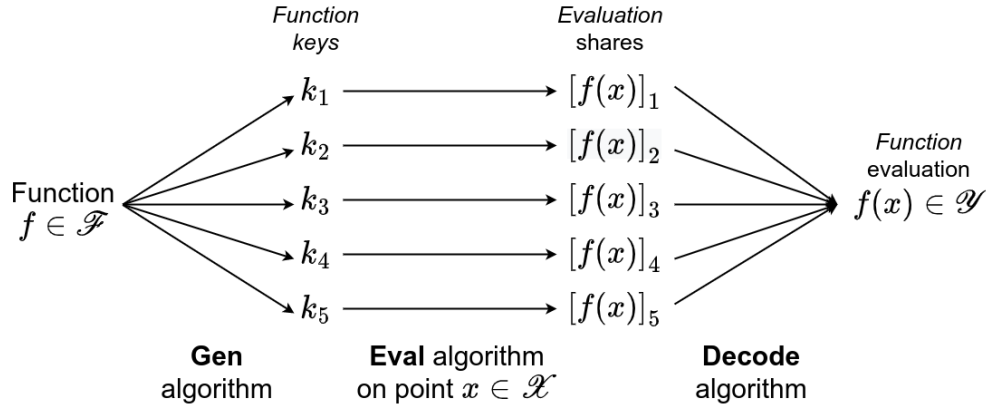


Figure 2.2 / Algorithms of an FSS scheme

Recently, several works [AFK⁺25, BBC⁺21, BGH⁺25] have explored FSS in the context of statistics and sparse data aggregation. Since sparse operations are central to this thesis, multiple chapters will present new FSS protocols optimized for such tasks. This chapter then provides an overview of the state of the art and of the applications to privacy-preserving statistics.

2.2.1 Definitions

We define p -party FSS schemes using three algorithms (illustrated in Fig. 2.2):

- **Gen** : $\mathcal{F} \times \mathbb{N} \rightarrow \mathcal{K}^p$: takes as input a function $f \in \mathcal{F}$ (all functions in $\mathcal{F}$ have the same domain $\mathcal{X}$) and a security parameter $1^\lambda \in \mathbb{N}$ and outputs p keys $k_1, \dots, k_p \in \mathcal{K}$. *This algorithm is executed by the secret holder.*
- **Eval** : $\mathcal{K} \times \mathcal{X} \rightarrow \mathbb{G}$: takes as input a key $k_i \in \mathcal{K}$ and a point $x \in \mathcal{X}$ and outputs a value $y_i \in \mathbb{G}$. *This algorithm is executed by each shareholder.*
- **Decode** : $\mathbb{G}^p \rightarrow \mathcal{Y}$: takes as input p shares $\{y_1 \dots y_p\}$ and outputs $f(x)$.

Gen is a probabilistic algorithm and Decode a deterministic algorithm. While in all existing schemes Eval is deterministic, our noisy schemes have a probabilistic Eval algorithm. The decoding algorithm is not systematically included in the scheme definition by existing works [BGI15], but we include it to have self-contained definitions (e.g., the correctness definition relies on it).

Definitions in the literature commonly use the notation $\mathbb{G}$ to refer to the share space, but the thesis is particularly interested in schemes efficient with $\mathbb{G} = \mathbb{F}_q$. Indeed, we focus on applications to statistics and ML that usually require share in $\mathbb{F}_q$ to aggregate several functions. Section 2.2.3 details these applications and motivate their need for efficient schemes in $\mathbb{F}_q$.

Function families A function family $\mathcal{F}$ [BCGI22] is a pair $(P_{\mathcal{F}}, E_{\mathcal{F}})$ where $P_{\mathcal{F}} \subseteq \{0, 1\}^*$ is an infinite collection of function descriptions $\hat{f}$, and $E_{\mathcal{F}}: P_{\mathcal{F}} \times \{0, 1\}^* \rightarrow \{0, 1\}^*$ is a polynomial-time algorithm defining the function described

by $\hat{f}$. In other words, each function description $\hat{f} \in P_{\mathcal{F}}$ describes a corresponding function f such that $f(x) = E_{\mathcal{F}}(\hat{f}, x)$.

For example, many FSS works focus on point functions; functions f such that $f(x) = \beta$ if $x = \alpha$, and $f(x) = 0$ otherwise. A description $\hat{f}$ for such functions is the tuple $(\alpha, \beta, \mathcal{X}, \mathcal{Y})$, with $\mathcal{X}$ the function domain and $\mathcal{Y}$ the output space. Note that all functions of a given family must share the same function domain and output space.

There exist schemes for various function families including point functions [BGI15] [BGI16, GI14], intervals [BGI16], and decision trees [BGI16]. Recently, there is also a growing interest in schemes supporting comparison functions: functions such that $f(x) = \beta$ when $x \leq \alpha$, 0 otherwise. These schemes called “Distributed Comparison Functions” (DCF) have applications notably in MPC pre-computations [BGI19] or private statistics [BMR25]. Like most FSS works [BCGI22], we focus especially on Distributed Point Functions (DPF) [GI14]; i.e., FSS schemes for point functions.

Security [BGI15] introduced two security definitions for FSS schemes: correctness (of the output) and privacy (of the input function).

Definition 2.3 (Correctness). Let $\mathcal{E} = (\text{Gen}, \text{Eval}, \text{Decode})$ be an FSS scheme for the function family $\mathcal{F}$. For any function $f \in \mathcal{F}$, for any point $x \in \mathcal{X}$, we have $k_1, \dots, k_p \leftarrow \text{Gen}(1^\lambda, f)$ and

$$\Pr[\text{Decode}(\text{Eval}(k_1, x), \dots, \text{Eval}(k_p, x)) - f(x) = 0] = 1.$$

Definition 2.4 ({Game-based} Privacy). Considering the following indistinguishability game for an adversary $\mathcal{A}$ composed of m corrupted parties:

- (1) The adversary provides two functions $f_0, f_1 \in \mathcal{F}$.
- (2) The challenger samples a bit $b \in \{0, 1\}$ and $(k_1, \dots, k_p) \leftarrow \text{Gen}(1^\lambda, f_b)$.
- (3) The adversary picks any subset $K_{\mathcal{A}}$ of at most m keys from $\{k_1, \dots, k_p\}$.
- (4) The adversary guesses $\hat{b} \leftarrow \mathcal{A}(K_{\mathcal{A}}, f_0, f_1)$.

A scheme is (m, p) -private if there exists a negligible function negl such that for all non-uniform PPT adversaries:

$$|P[b = \hat{b}] - 1/2| \leq \text{negl}(\lambda)$$

Existing works mostly relied on this game-based privacy definition given by [BGI15]. Section 11.2.2 formalizes a new security notion (function-private output) which is more naturally approachable via a simulation-based definition. For consistency, we then rely on a simulation-based definition for privacy. We propose

the following simulation-based definition:

Definition 2.5 (**{Simulation-based} Privacy**). Let $\mathcal{F}$ be a set of functions with a common domain $\mathcal{X}$. Let $\mathcal{E} = (\text{Gen}, \text{Eval}, \text{Decode})$ be an FSS scheme. Let $m, p \in \mathbb{N}$ with $m < p$. We say that $\mathcal{E}$ is private against a semi-honest adversary corrupting up to m of p parties if for any $A \subset \{1, \dots, p\}$ of size at most m , there exists a PPTA $\mathcal{S}$ such that for all PPTA $\mathcal{D}$, there exists a negligible function negl , such that for all $f \in \mathcal{F}$ and for all $\lambda \in \mathbb{N}$

$$|\Pr[\mathcal{D}(\{k_i\}_{i \in A}, f) = 1, (k_1, \dots, k_p) \leftarrow \text{Gen}(1^\lambda, f)] - \Pr[\mathcal{D}(\mathcal{S}(\lambda, \mathcal{F}), f) = 1]| \leq \text{negl}(\lambda) .$$

This simulation-based definition is a refined version of the definition proposed by Boyle [BCGI22]. This refinement is necessary to our work because we extend this definition to formalize the notion of function-private output.

2.2.2 State-of-the-art schemes

Two-party DPF [BGI15] presented the reference 2-party scheme; $O(\log N \log q)$ key size (for evaluation shares $\mathcal{U}(\curvearrowright)$ in $\mathbb{F}_q$). This scheme has been further optimized by various works: key size optimization [BGI16], “incremental” DPF [BBC⁺21], reusable keys [BGIK22], MPC-based key generation [DS17], verifiable DPF [dCP22]. Thanks to its logarithmic key size, this scheme is used in many application papers including in anonymous communications [NSD22], private statistics [BBC⁺21], and access control [SBYP23].

Three-party DPF [BKK020] described a dishonest-majority scheme with $O(\sqrt{N})$ key size. [BK022, ZYP24] later built honest-majority schemes with logarithmic key sizes.

Multi-party DPF Now that 2-party and 3-party schemes reached a logarithmic key size, improving general multi-party schemes is the natural next step. As this thesis proposes several new multi-party schemes, we thoroughly survey existing schemes, their advantages and disadvantages. Table 10.3 of Section 10.2 summarizes this overview.

The first multi-party DPF schemes was proposed by [BGI15]. They rely on a PseudoRandom Generator (PRG) to build a scheme with $O(\sqrt{N}q^p)$ key size ;for p parties and evaluation shares in $\mathbb{F}_q$. The exponential factor q^p makes it impractical for $q \neq 2$. This issue was highlighted by [BCGI22, KPM24]. [BCGI22] proposed a solution based on the Chinese Remainder Theorem to replace q^p with $\sum_i q_i^p$ ($\{q_i\}_i$ being the prime factors of q). Chapter 9 shows that this optimization significantly improves the key sizes, but the scheme remains impractical for any $q > 210$.

Over the last ten years, several directions have emerged to build multi-party schemes outperforming the PRG-based scheme proposed by [BGI15].

The first direction was DDH-based schemes [CBM15, KPM24]. These schemes have the best asymptotic key size $O(\sqrt{N} \log q)$ with small constant factors. However, the secrets are encoded using elliptic curve points, so they require a non-linear share decoding limiting the applications (as detailed in Chapter 10).

Second, [BK022] leveraged the honest-majority assumption to build more efficient schemes. They proposed two schemes: a PRG-based scheme with $O(\sqrt[4]{N} q^p)$ key size built upon [BGI15], and an information-theoretic DPF with $O(\sqrt{N} \binom{p}{m} \log q)$ key size. As the PRG-based scheme inherits the exponential factor from [BGI15], only the information-theoretic scheme is practical.

Third, some works [ABG⁺24, CKY25, DHRW16] used homomorphic secret sharing (HSS) to build FSS schemes. In particular, [CKY25] designed an HSS-based multi-party DPF with logarithmic key size. HSS is tightly related to FSS as it focuses on generating shares on which parties can evaluate functions without interactions between them. These schemes have significant constant factors in the key size, that limit their practicality. Existing FSS works [CBM15, BK022, KPM24, GWW25] usually cite them, but do not compare to them.

Fourth, [GWW25] recently relaxed the privacy definition via a new definition called “ ϵ -privacy” (which has no link with differential privacy). This relaxation enables them to modify the scheme of [BGI15] to obtain a key size of $O(\sqrt{N} \lambda^3 p^3 \log q)$. Due to the factor $\lambda^3 p^3$, this scheme is highly inefficient for realistic problem sizes; as shown in Chapter 10.

Finally, Krips and Pullonen-Raudvere [KP25] introduced two new hardness assumptions to build a multi-party scheme with logarithmic key size. While their work represents a promising approach to build multi-party schemes with logarithmic key size, the introduction of two new assumptions still requires further research to estimate the parameters under which these assumptions hold.

DCF [BGI15] adapted their 2-party DPF to build a two-party DCF scheme with $O(\log N \log q)$ key size. They also adapted their multi-party DPF to build a multi-party DCF with $O(\sqrt{N} q^{\frac{p-1}{2}} \log q)$; suffering from the same scalability issues as their DPF. Recently, [KPM24] proposed multi-party DCF inspired by the DPF of [CBM15]. The literature describes no other DCF; the DCF baselines are then weaker than those in DPF.

2.2.3 FSS Applications to Privacy-Preserving Statistics

While FSS has been initially designed for private information retrieval [GI14], several works have described over the years applications to statistics. In these applications, FSS enables the aggregation or analysis of sensitive data without revealing individual contributions, making it particularly relevant for scenarios

Algorithm 1 Simple DPF-based histogram-making protocol

```

1: Let  $\mathcal{E} = (\text{Gen}, \text{Eval}, \text{Decode})$  be a  $p$ -party DPF scheme.
2: Let  $\{P_1, \dots, P_n\}$  be a set of  $n$  data owners, each owning a value  $e_i \in \{1 \dots N\}$ .
   Let  $\{S_1 \dots S_p\}$  be a set of  $p$  servers.

3: // Secret value sharing (Client side)
4: for each data owner  $P_i \in \{P_1, \dots, P_n\}$  do
5:   Define  $f_i$  a point function s.t.  $f_i(e_i) = 1$ , 0 otherwise.
6:   Generate the DPF keys  $\{k_{i,1}, \dots, k_{i,p}\} \leftarrow \text{Gen}(\hat{f}_i, \lambda)$ .
7:   Send one key  $k_{i,j}$  to each server  $S_j$ .

8: // Share aggregation (Server side)
9: for each server  $S_j \in \{S_1, \dots, S_p\}$  do
10:  Initialize the vector share  $\llbracket h \rrbracket_j \in \mathbb{G}^N$  with zeros.
11:  for each key  $k_{i,j} \in \{k_{1,j}, \dots, k_{n,j}\}$  do
12:    for each  $x \in \{1, \dots, N\}$  do
13:      Evaluate the share  $\llbracket f_i(x) \rrbracket_j \leftarrow \text{Eval}(k_{i,j}, x)$ .
14:      Update  $\llbracket h[x] \rrbracket_j \leftarrow \llbracket h[x] \rrbracket_j + \llbracket f_i(x) \rrbracket_j$ .
15:  Send the vector share  $\llbracket h \rrbracket_j$  to the other servers.

// Result decoding (Server side)
16: Decode the histogram  $h \leftarrow \text{Decode}(\llbracket h \rrbracket_1, \dots, \llbracket h \rrbracket_p)$ .

```

such as distributed surveys [BBC⁺21]. This subsection surveys these applications, and identify their common requirements.

Histograms Histograms are a basic statistical method to represent data distributions. These statistics are popular in many research fields, and can sometimes be applied on sensitive data; e.g., to represent a body weight distribution in healthcare or a salary distribution in sociology.

Private histograms are a straightforward application of DPF mentioned in some existing works [BBC⁺21, DWA⁺24]. Algorithm 1 describes a basic DPF-based histogram-making protocol. Moreover, DPF can implement other related operations such as heavy hitters [BBC⁺21] or secure maximum [CMZH24].

Empirical cumulative distribution function (ECDF) ECDF is an estimation of the cumulative distribution function of a random variable. They are an essential building block in many statistics and ML computations.

For example, [BMR25] used them to compute ROC curves (to assess the accuracy of a classifier) or to perform a Hosmer-Lemeshow statistical test (to calibrate medical instruments). Their work proposed to compute private ECDF more efficiently via DCF.

Sparse vector aggregation [AFK⁺25] have proposed a sparse vector aggregation based on DPF. A sparse vector is a vector containing a minority k of non-zero values. Thus, such vectors can be represented as the aggregation of k point functions (one for each non-zero element in the vector). To build an FSS-based sparse vector aggregation, we can then use DPF (like in private histograms).

Common requirements All these statistics applications follow the same structure as Algorithm 1: (1) secret sharing, (2) value/share aggregation, (3) result decoding. The second step is particularly interesting because it requires aggregating shares locally (see Line 14).

This local aggregation induces two requirements. First, the FSS schemes must be efficient with output in a generic field $\mathbb{F}_q$ (for $q \gg 2$) because they need to aggregate private values up to a value q . This requirement was not present in historical FSS applications like PIR for which bit fields $\mathbb{F}_2$ were sufficient. Second, it requires a share homomorphism:

$$\begin{aligned} & \text{Decode}(\sum_i \llbracket f_i(x) \rrbracket_1, \dots, \sum_i \llbracket f_i(x) \rrbracket_p) \\ &= \text{Decode}(\llbracket \sum_i f_i(x) \rrbracket_1, \dots, \llbracket \sum_i f_i(x) \rrbracket_p) = \sum_i f_i(x) \end{aligned}$$

While these requirements seem trivial several multiparty schemes do not satisfy them. For example, PRG-based multi-party schemes [BGI15, BK022] are exponential in the field size, making them impractical for such applications. These requirements will guide our contributions in Part IV to improve FSS schemes, specifically for applications to statistics and sparse computations.

2.3 Other cryptographic primitives

While our core contributions design new FSS protocols or build upon secret-sharing protocols, some chapters also mention or discuss other cryptographic primitives that we present below. These primitives are useful building blocks for MPC protocols under different threat models than those of secret sharing protocols. For example, zero-knowledge proofs are the basis of many protocols secure against malicious attackers.

Homomorphic encryption Homomorphic encryption is a special form of encryption that supports arithmetical computations over encrypted values; e.g., additive schemes guarantee that $\text{Enc}(k, a) + \text{Enc}(k, b) = \text{Enc}(k, a + b)$. Fully Homomorphic Encryption (FHE) [Gen09] schemes support at the same time multiplications and additions. These algorithms can be particularly expensive for some operations (e.g., comparisons [MEHP25]), and the handling of the decryption key can sometimes require MPC protocols to spread the trust and avoid a trusted decoder.

Zero-knowledge proofs (ZKP) ZKPs [G094] are techniques to prove properties about ciphertexts (e.g., that two encrypted values are equal or that an encrypted value is in a given range) without disclosing anything else. A ZKP involves two agents: the prover (who provides the proof) and the verifier (who verifies its correctness).

Trusted Execution Environment (TEE) A trusted execution environment (TEE) [SAB15] (sometimes called “secure enclave”) is a secure section of the processor in which confidentiality and integrity for the processed data and code are ensured. Even the hardware owner cannot read or modify the data processed in the TEE. Anyone can audit the code executed in the environment, and the hardware design ensures that the data processed inside remains confidential.

They require no trust other than in the manufacturer. Many major chip manufacturers have already designed such environments; e.g., Intel SGX [Int15].

While they are attractive due to their minimal requirement for significant infrastructure changes, they are not without their limitations. Indeed, TEEs are theoretically secure, but their practical security can be compromised through side-channel attacks. Section 5.6.6 details these security shortcomings.

3 Sparse data in Privacy-Preserving Computations

The main objective of this thesis is to design new MPC protocols that can process sparse data efficiently. Such datasets arise naturally in many privacy-preserving applications, making their efficient processing a key challenge. This chapter first introduces the concept of sparse data, then presents existing MPC protocols for sparse data, and finally skims through their applications in statistics and ML.

3.1 Sparse data

The term *sparse data* refers to data containing a large proportion of zero values. If stored using the default dense matrix format, these zeros occupy significant memory. When the proportion of zeros is high, using sparse data representations can drastically reduce storage requirements. Let $\text{nnz}(x)$ (resp. $\text{nnz}(X)$) denote the number of non-zero values in a vector x (resp. matrix X).

Algorithms that leverage sparsity are called “sparse algorithms”, while classical algorithms are referred to as “dense algorithms”.

Sparse data representation Several sparse data representations exist [BGS11], each optimized for different operations. In other words, there is no perfect sparse data representation.

In this thesis, we focus on the **tuple representation** [BGS11]: each sparse vector is stored as a list of non-zero tuples $t = (i, v_i)$, where i is the coordinate and v_i the value. We denote $\text{coord}(t) = i$ and $\text{val}(t) = v_i$.

Sparse data in practice Sparse data is prevalent in many real-world applications, such as recommender systems, text analysis, drug discovery, and social network analysis. Some representative datasets are shown in Table 3.1.

Why sparse algorithms matter One might consider using dense algorithms and simply allocating more memory, but this quickly becomes impractical. For instance, multiplying $10^6 \times 10^6$ matrices in Python would require roughly 19 TB of memory with dense algorithms, while sparse algorithms only need a few gigabytes (using NumPy and SciPy). The first motivation of sparse algorithms is to avoid the memory issues caused by dense algorithms. Furthermore, they often provide significant performance gains because their cost is proportional to the number of non-zeros. Thus, sparse algorithms are essential not only for memory efficiency but also for performance gains in high-dimensional datasets.

Dataset	Domain	Sparsity (%)
MovieLens 1M [HK16]	Recommender	95
Netflix [BL ⁺ 07]	Recommender	99
Yahoo Movies	Recommender	99.7
Tox21 [HXN ⁺ 16]	Drug discovery	99.9
20 Newsgroups [RL08]	Text analysis	99.9
Bookcrossing [ZMKL05]	Recommender	99.99
Flickr [CMG09]	Social network	99.999

Table 3.1 / Example sparse datasets and their sparsity rates

Sparsity-size correlation Real-world sparse datasets show an interesting phenomenon: the sparsity is correlated to the matrix size. Indeed, larger sparse datasets have in average a larger sparsity. Thus, **the larger datasets are, the more beneficial the sparse algorithms become**. The real-world datasets that we arbitrarily selected from [LLW17] satisfy the same trend: MovieLens 1M has 95% sparsity for 1.7K columns, Yahoo Movie has 99.7% sparsity for 12K columns, Bookcrossing has 99.99% sparsity for 340K columns, and flicker has 99.999% sparsity for 500K columns.

To understand this phenomenon, let us take the example of recommender systems on a marketplace (e.g., Amazon): the matrix quantifies the interaction of each consumer with each product. The dataset is sparse because each consumer only interacts with a small subset of all possible products. If the marketplace increases its number of products by a factor 1000, the consumer will not consume 1000 times more products. Even if the consumption increases a bit, existing datasets show that it does not follow the matrix growth; i.e., the sparsity increases.

Theoretical point of view The correlation between sparsity and matrix size is observable analytically. Sparse datasets can often be represented as graphs: e.g., a recommendation dataset is a bipartite graph linking n customers to m products.

The literature [ACL01] showed that, in many application domains, graph data follows a “power law” distribution. In particular, if we take a random vertex $v \in V(G)$ of a graph G , then the probability distribution of the degree $d(v)$ of v is given by:

$$f(x) = P(d(v) = x) = \frac{x^{-\gamma}}{Z(\gamma)} \quad (3.2)$$

for some $\gamma > 1$ where $Z(\gamma) = \sum_{i=1}^{\infty} i^{-\gamma}$ is a normalization constant. Typically, we have $2 \leq \gamma \leq 3$ [ALPH01], but application domains exist where other values of γ are adequate. Practical applications may deviate a bit from Equation (3.2),

especially for small degrees, as this equation is mainly intended to describe asymptotic behavior.

In a dataset with n rows (i.e, vertices) where every vertex has degree at most m , one can use a corrected normalization constant $Z(\gamma, m) = \sum_{i=1}^m i^{-\gamma}$. Remember that, in our case, the degree of a vertex corresponds to the number of non-zeros in a row. The expected degree of a vertex is:

$$\mathbb{E}_v[d(v)] = \sum_{i=1}^m \frac{i^{1-\gamma}}{Z(\gamma, m)}$$

We can bound $Z(\gamma, m) \geq 1 + \int_1^m x^{-\gamma} dx = 1 + 1/(1-\gamma)$ and $\sum_{i=1}^m i^{1-\gamma} < \int_1^m x^{-\gamma} dx = (1 - m^{2-\gamma})/(\gamma - 2) < 1/(2-\gamma)$. Therefore, we can bound the average degree by a constant K depending neither on n nor on m . We can rewrite the number of edges in graph G as $|E| = nK = O(n)$. Hence, the sparsity of the adjacency matrix of the graph G decreases with n :

$$\frac{|E|}{nm} = O(m^{-1}). \quad (3.3)$$

This formalizes the observed correlation: larger datasets are often sparser.

3.2 Secure operations on sparse data

While most MPC research has focused on dense data, several works such as [BGH⁺25,SGRP19] have proposed protocols specifically tailored for sparse data. These protocols aim to leverage sparsity to reduce communication, computation, and memory costs compared to their dense equivalent. In particular, they describe methods for secure sparse vector addition and sparse matrix multiplication. This section reviews the state of the art for these operations.

3.2.1 Addition

Secure vector addition is one of the most basic building blocks in privacy-preserving statistics and machine learning. Given n private vectors v_i , the goal is to compute $\sum_{i=1}^n v_i$ without revealing information about any v_i . This operation is often referred to as *secure aggregation* [MJÖC23]. These protocols typically generalize to matrix addition.

Three main approaches have been proposed for secure sparse aggregation: *sorting-based*, *FSS-based*, and *shuffling-based*.

Sorting-based Several MPC works [AHI⁺22,BGG⁺22,KCY23] propose protocols based on secure sorting. While each paper includes specific optimizations, Algorithm 2 shows the basic protocol that most of them build upon. In this approach, the secret sparse vectors x and y are shared in *tuple representation*. For example,

Algorithm 2 Secure sparse vector addition**Input:** $\llbracket x \rrbracket$ and $\llbracket y \rrbracket$: two secret-shared sparse vectors**Leakage:** Number of non-zero elements in each vector

```

1: function SPARSEVECTORADD( $\llbracket x \rrbracket, \llbracket y \rrbracket$ )
2:   Let  $\llbracket z \rrbracket \leftarrow \text{CONCATENATE}(\llbracket x \rrbracket, \llbracket y \rrbracket)$ 
3:   Let  $\llbracket z \rrbracket \leftarrow \text{SORTBYINDEX}(\llbracket z \rrbracket)$ 
4:   AGGREGATEEQUALINDICES( $\llbracket z \rrbracket$ )
5:   REMOVEPLACEHOLDERS( $\llbracket z \rrbracket$ )
6:   return  $\llbracket z \rrbracket$ 

```

a vector x with k non-zeros is represented as $\llbracket x \rrbracket = [(\llbracket i_1 \rrbracket, \llbracket v_{i_1} \rrbracket), \dots, (\llbracket i_k \rrbracket, \llbracket v_{i_k} \rrbracket)]$. The oblivious sorting is used to match (privately) the elements from x and y with equal coordinates and multiply them.

This algorithm has a communication and computation cost of $O((\text{nnz}(x) + \text{nnz}(y)) \log(\text{nnz}(x) + \text{nnz}(y)))$, but requires $O(\log(\text{nnz}(x) + \text{nnz}(y)))$ rounds of interaction.

FSS-based As discussed in Section 2.2.3, [AFK⁺25, BGH⁺25] used FSS to perform secure vector addition. Here, a sparse vector is represented as a multipoint function with k non-zero points. Shareholders evaluate these functions on the entire domain and locally sum the results.

This method requires only two communication rounds: one to distribute the function shares, and one to reconstruct the output. However, it incurs $O(n)$ evaluation cost over the domain, and reconstruction requires sending shares of the entire output vector. This approach reduces the load on data owners, but increases the computation for the computation parties. In practice, FSS-based approaches are effective for moderately large vectors, while sorting-based methods scale better for extremely large vectors.

Moreover, observe that Algorithm 2 leaks the number of non-zeros in each vector. This leakage is also present in the FSS-based approach because the FSS key size is proportional to the number of non-zeros. Chapter 4 explains why leakage is necessary to build secure protocols efficient on sparse data.

Shuffling-based A third option is to use *secure shufflers* to aggregate sparse vectors. Since secure shuffling is a special cryptographic primitive not introduced yet in this thesis, we defer a detailed discussion to Chapter 5, which surveys secure shufflers and their applications.

3.2.2 Multiplication

A few papers [CZW⁺21, CCL⁺21, SGRP19] studied secure sparse multiplications in a two-party setting. However, these protocols require the data owners to actively participate because one of the computation parties must *know the plaintext sparse data*. Hence, they do not support more than two data owners simultaneously, which is too limiting, as modern ML applications involve thousands of data owners.

Typical ML applications require a more generic MPC setup: outsourced training (presented in Section 3.3.2). In outsourced MPC, data owners simply share their data with computation parties and then disconnect. The computation parties are commonly assumed to have more computational power (and reliability) than average data owners. Once the computation parties have received shares from all parties, they can execute the MPC protocol (e.g., a multiplication).

This outsourced setting supports arbitrary number of data owners, contrary to existing works [CZW⁺21, CCL⁺21, SGRP19]. However, the adaptation of these works either results in an inefficient construction or is not straightforward because they require a shareholder to know the plaintext data. The secure multiplication of sparse matrices in the outsourced setting is then **a significant open problem** (addressed in Chapter 8).

Other related works Recently, [FGD⁺25, YCF⁺25] have described a sparse matrix-vector multiplication under FHE. As our thesis focuses on secret-sharing-based protocols, we cannot compare to them because the threat models significantly differ.

Some information-theory works [EXWB23, XBW22] studied the multiplication of a secret-shared matrix with a public matrix. Due to the public input, their setting differs from classic MPC where all inputs are secret.

Finally, [NWI⁺15] presented secure graph computations (a problem closely related to sparse operations due to the sparse adjacency matrices), but they do not present multiplication protocols.

3.3 Applications to statistics and machine learning

These MPC protocols have found several applications notably in privacy-preserving statistics and privacy-preserving machine learning (PPML). This section presents these applications, and the secure sparse operations that they can require.

3.3.1 Privacy-preserving statistics

Many statistical computations naturally exhibit sparsity when expressed in vector form. In this context, secure sparse additions are especially useful. The best example is *histogram computation* [AHI⁺22, AHI⁺22]: each party's input can be

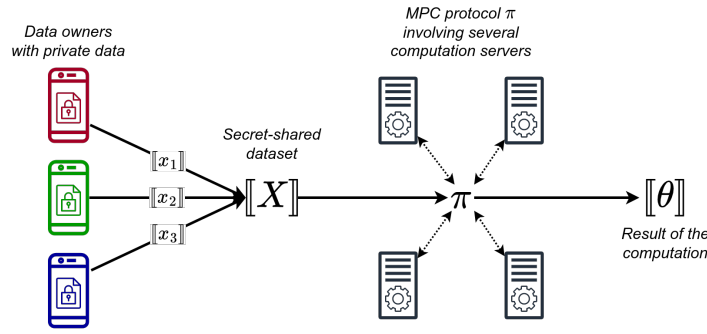


Figure 3.4 / Outsourced MPC setting

represented as a sparse vector with a single non-zero entry corresponding to the observed category, and the global histogram is obtained by summing these vectors.

When the histogram domain is very large (e.g., tracking counts over millions of possible items), algorithms optimized for data sparsity—such as those of [AHI⁺22]—become not only more efficient, but sometimes the only feasible approach. The same reasoning applies to related tasks such as *heavy hitters*, where the goal is to find the most frequent elements without revealing the full dataset.

Beyond histograms, other statistical measures can benefit from secure sparse operations. For instance, performing private set intersection cardinality when the universe of possible elements is large, but each party holds only a small subset. In all these cases, leveraging sparsity reduces memory usage, bandwidth, and runtime; making privacy-preserving statistics practical at scale.

3.3.2 Secure Outsourced Training

Many PPML works (including [MZ17, WTB⁺21, WGC19] and Chapter 8) focus on an “outsourced” computation setup: each data owner shares their secret data with a group of computation servers. These servers build a secret-shared matrix $\llbracket X \rrbracket$ containing all data owners’ inputs (e.g., one row per data owner¹) and perform secure computations on this matrix. For example, the servers can execute a protocol to train an ML model $\llbracket \theta \rrbracket$ on the dataset X , as in Fig. 3.4. In this setting, the data owners share their secret data and disconnect. Outsourced protocols support an arbitrary number of data owners because they separate input parties (i.e., data owners) from computation parties.

In these applications, MPC protocols optimized for sparse data become essential whenever the training dataset X is sparse. As existing works on secure sparse multiplications do not support this setting, this represents an essential gap in the PPML literature which is filled by Chapter 8.

While we are particularly interested in ML training, the outsourced setting can

¹This is sometimes referred to as “horizontal” data partitioning in PPML literature.

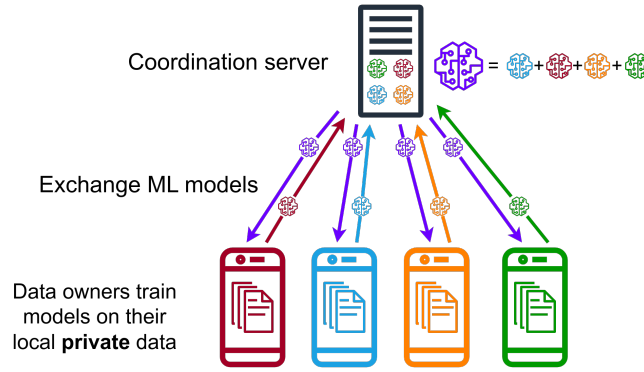


Figure 3.5 / High-level Federated Learning architecture

also be useful for secure ML inference (i.e., evaluation of an ML model on a private input). Some MPC protocols [JGB⁺24, KMY⁺24] have been applied to such operations, but FHE-based solutions such as [FSB⁺24] are usually preferred because their threat model fits nicely the “inference-as-a-service” architecture adopted by AI companies. Indeed, secret-sharing protocols require multiple parties to execute the secure computations, while FHE-based protocols can outsource all the computation to a single (untrusted) server.

3.3.3 Federated Learning

Federated Learning (FL) [KMA⁺21] is a PPML paradigm enabling to collaboratively train an ML model without exchanging (private) training data. While there exist several FL variants, the most popular setup is “horizontal FL:” each data owner owns a small dataset over the same set of features.

Traditional FL protocols are iterative decentralized algorithms with each iteration having the following structure: (1) Each data owner trains a local ML model on its private dataset, (2) Each data owner sends its local model to a server, (3) the server aggregates the local model and sends back a global model. Fig. 3.5 sketches a high-level representation of FL.

Such aggregation-based FL protocols enable to train a wide variety of ML models: any model which training relies on gradient descent [KMRR16] (e.g., neural networks or linear regression), k-means protocols [ZLH⁺22, LWCQ23, KKN20], Singular Vector Decomposition [CWZ⁺22], random forests [dSAC⁺20, LLL⁺20], and Gradient Boosted Decision Trees [CFJ⁺21].

Even though FL does not require training data exchanges, the ML models (or gradients) exchanged during these protocols can leak private information [NSH19, vDDH25]. To enhance privacy, many works [KMA⁺21] have relied on cryptographic protocols [MJÖC23] referred to as secure aggregation.

Recently, gradient sparsification techniques [SCJ18] were proposed to reduce communications in FL. To preserve privacy while benefiting from these efficiency gains, secure sparse aggregation protocols [BAT21] have become necessary.

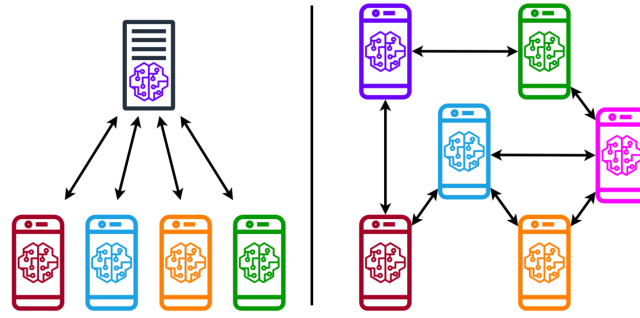


Figure 3.6 / Federated Learning (left) vs. Decentralized Learning (right)

Decentralized Learning While efficient, FL protocols can encounter scalability issues when the number of data owners grows, due to the bottleneck created by the aggregation step. This step requires either a trusted party or a cryptographic protocol to aggregate all gradients. Because aggregation is centralized, it can create a performance bottleneck.

To address this, some works consider a fully decentralized setting, known as decentralized ML [CB22]. Gradients are exchanged in a peer-to-peer manner without a central coordinator, eliminating the need for centralized aggregation. This paradigm is closely related to FL and is sometimes referred to as *fully decentralized* FL. Figure 3.6 presents this setup, with comparison to FL.

As in FL, gradient exchanges raise privacy concerns. However, deploying cryptographic protocols in a fully decentralized network is more challenging. Consequently, most existing approaches rely instead on differential privacy techniques [CB22,CEBM22] to preserve privacy.

4 Leakage in sparse computations: a sense of déjà vu

In MPC, the word *leakage* refers to any information revealed by the protocol that would not be revealed by an ideal trusted curator. When computing $f(x)$, any additional information beyond $f(x)$ revealed by the MPC protocol is considered leakage. In the MPC community, leakage often carries a strongly negative connotation, perceived as a side effect that must be avoided at all costs. Some works instead use the term *public knowledge* to refer to this notion, without raising the same negative concerns as the term leakage.

From this definition, it is natural to view leakage as undesirable. Yet this perception is not universal across cryptography. In encrypted search, for instance, leakage has long been acknowledged (and even accepted) as a necessary condition for efficiency [BGW24]. This chapter examines the role of leakage in secure sparse operations, making parallels with encrypted search to argue that leakage is essential for practical MPC protocols on sparse data.

The goal of this chapter is to show that MPC protocols for sparse data do not constitute an entirely new cryptographic problem. Related primitives have already been studied extensively, and the insights gained from those works can be leveraged to better understand and design MPC protocols optimized for sparse data.

4.1 Necessary leakage in secure sparse operations

The issue of leakage has appeared (sometimes explicitly, sometimes implicitly) in most works on sparse data in MPC. Several protocols, including [CCL⁺21, FGD⁺25] [SGRP19], explicitly require some form of leakage. For example, [SGRP19] assume that “the sparsity metric of the input is public,” where the definition of this metric varies across their algorithms (e.g., the total number of non-zeros, or the number of non-zero rows or columns). Although acknowledging its presence, these works do not discuss why such leakage is inherent to sparse operations.

On the contrary, [YCF⁺25] recently proposed an FHE-based sparse-dense multiplication without identifying the leakage in their own construction. Their algorithm relies on Low Diagonal Decomposition to split a large sparse matrix into smaller ones, which are then encrypted and uploaded to an untrusted server. They compare their “secure” algorithm to a “leaky” algorithm that reveals non-zero positions. Yet, their decomposition still leaks information because the number of output matrices is correlated with the number of non-zeros in the input, indirectly leaking sparsity information.

In short, leakage has been a recurring source of confusion in the community. Some

papers [SGRP19] briefly acknowledge it without examining its necessity, while others fail to identify the leakage present in their own algorithms. Part of this reluctance may come from the negative connotation the term “leakage” carries. We argue instead that **leakage is unavoidable, and in fact necessary, for efficient MPC protocols on sparse data.**

The reason is straightforward: in both plaintext and secure settings, algorithms on sparse data are considered efficient when they exploit sparsity to reduce cost. If the runtime or communication complexity of a protocol scales inversely with sparsity, then its execution necessarily reveals some information about that sparsity. In other words, any efficient sparse MPC protocol must leak (at least partial) information about the sparsity. The only way to avoid leakage entirely is to ignore sparsity and fall back on generic dense protocols, sacrificing efficiency. Thus, leakage is not a side effect to be eliminated, but the inherent cost of achieving efficient sparse computations.

We identify four typical leakage profiles in existing protocols (illustrated in Fig. 4.1). For each profile, we list algorithms/papers that rely on them:

- **Number of non-zeros in the matrix:** e.g., matrix-vector multiplication (Algorithm 6) in Chapter 8, vector addition (Algorithm 2) in Chapter 3, and the “non-leaky” scheme of [YCF⁺25].
- **Number of non-zeros per row:** e.g., matrix-matrix multiplication (Algorithm 9) in Chapter 8, and ELLPACK-based sparse-dense multiplication in [FGD⁺25].
- **Number of non-zero columns:** e.g., the schemes of [SGRP19] and the privacy-sensitive variant of [CCL⁺21].
- **Positions of non-zeros:** e.g., the “leaky” scheme of [YCF⁺25], the privacy-insensitive variant of [CCL⁺21], and CSR-based sparse-dense multiplication in [FGD⁺25].

Fig. 4.1 illustrates well that each leakage profile induces different privacy guarantees. Leaking the exact positions of non-zeros reveals far more than leaking only their count. Yet minimizing leakage is not always the right goal: additional leakage can often enable much more efficient protocols. Hence, the design of MPC for sparse data should not aim to eliminate leakage, but rather to obtain the best privacy-efficiency trade-off.

While new to MPC on sparse data, this trade-off is well established in encrypted search, where different leakage profiles have been standardized and analyzed. By drawing parallels to this literature, we can better understand the risks of leakage in sparse MPC and import existing insights into our problem.

The next section therefore introduces relevant primitives from encrypted search, together with their leakage profiles. We then return to connect these with our own leakage taxonomy in sparse computations.

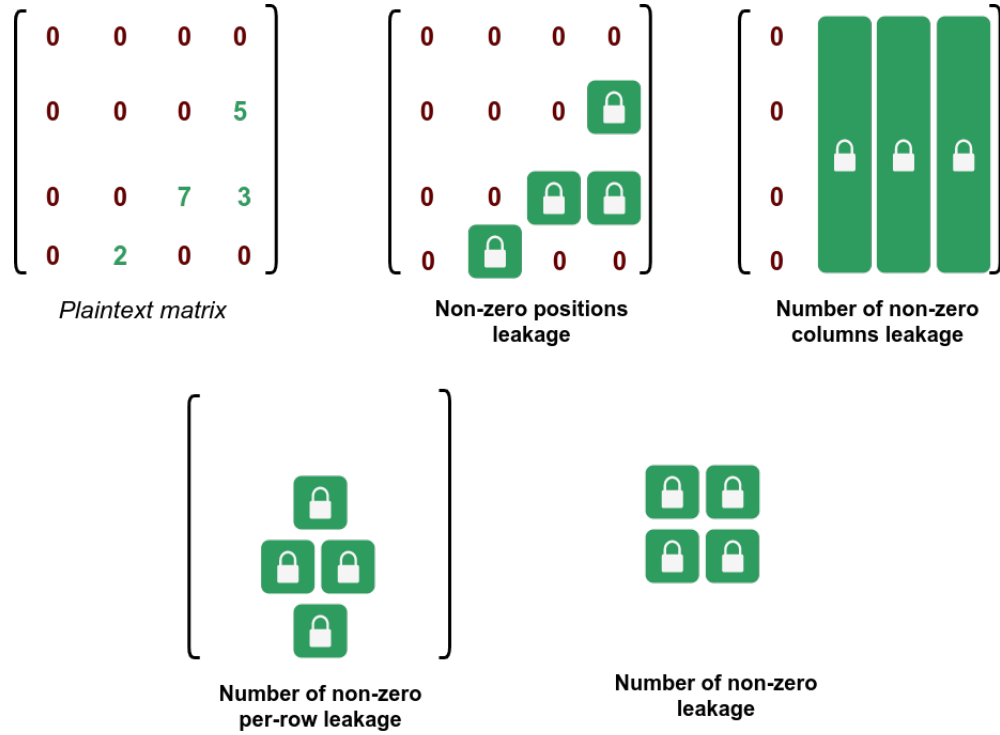


Figure 4.1 / Leakage profiles in MPC protocols for sparse data

4.2 Relation to encrypted search

This section first introduces several encrypted search protocols, their definitions, and their connections to sparse computations. Such parallels suggest that insights and results from the encrypted search community can help design secure and efficient computations on sparse data. At the end of the section, we discuss the leakage taxonomy commonly used in the encrypted search literature.

Although the literature proposes many protocols for secure search, we focus on three representative ones: Private Information Retrieval, Searchable Symmetric Encryption, and Oblivious RAM.

4.2.1 Private Information Retrieval (PIR)

As more and more data becomes publicly available online, accessing it can raise significant privacy concerns for users. Even if documents are public, querying them may reveal sensitive information, such as political views, religious beliefs, or sexual orientation. PIR was introduced to address this problem by allowing users to query public databases without revealing the subject of their query. In a PIR system, the user generates an “encrypted query” from a document identifier, and the server responds with an encrypted document that the user can later decrypt. Fig. 4.2 illustrates the general structure of PIR protocols.

The literature usually defines PIR protocols through four algorithms [BMW24] [HHC⁺23, MW22, PSY23]:

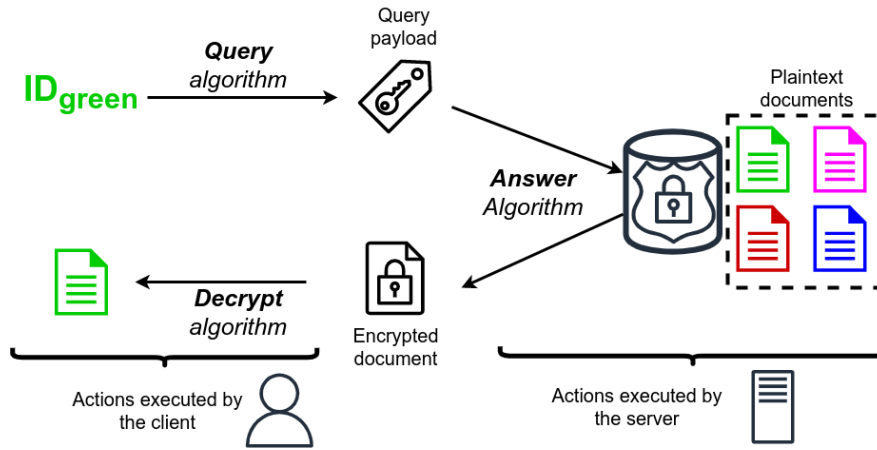


Figure 4.2 / PIR protocol

- $\text{Setup}(1^\lambda, DB)$: takes as input a security parameter 1^λ and the database DB . The output depends on the protocol (e.g., cryptographic keys [MW22] or a preprocessed database [BMW24]).
- $\text{Query}(id[, qk])$: takes as input a document identifier id and (for some protocols [MW22]) a “query key.” It outputs a query q .
- $\text{Answer}(q, DB[, pp])$: takes as input a query q , the database DB , and (for some protocols) public parameters or a cryptographic key. It returns an encrypted document d_{Enc} .
- $\text{Decrypt}(d_{\text{Enc}}, k[, pp])$: takes as input an encrypted result d_{Enc} , a decryption key k , and (for some schemes) other parameters. It returns the requested document.

This description mainly applies to single-server PIR protocols. There also exist multiserver PIR schemes [GI14, KOR19] that rely on MPC primitives such as DPF. In these settings, the client receives shares of the result from different servers and reconstructs the output locally.

Link with sparse computations Interestingly, the PIR problem can be reformulated as a sparse multiplication. The database can be seen as a dense vector of n entries, while the query is a binary vector with a single non-zero value (representing the one-hot encoding of the document identifier). The Answer algorithm consists of an element-wise multiplication of the two vectors, followed by a sum of the results (see Fig. 4.3). Since the query vector is sparse by construction, the computation is essentially a sparse-dense multiplication.

This perspective is made explicit in works like Spiral [MW22]. In Spiral, the client sends an encrypted identifier, and the server “unpacks” it into a sparse vector. In fact, their work indirectly provides an efficient way to encode and transmit a homomorphically encrypted sparse vector. Although this sub-protocol was not originally presented in those terms, it can be seen as a building block for efficient

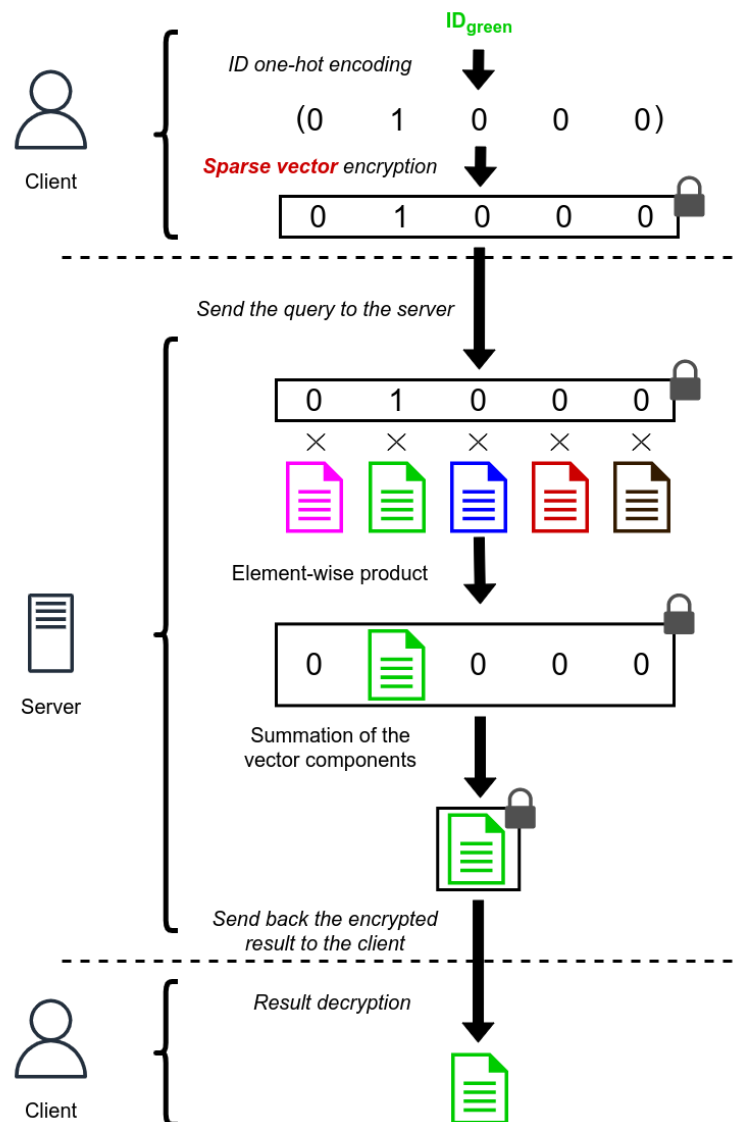


Figure 4.3 / PIR protocol represented as a sparse computation

sparse multiplications under FHE.

The notion of sparsity has also been addressed explicitly in recent PIR papers [PSY23] studying the problem of *keyword PIR*. Unlike standard PIR, where the database is a dense array of n documents, keyword PIR deals with sparse query spaces where most identifiers do not map to any document. While standard PIR corresponds to a sparse-dense multiplication, keyword PIR requires a sparse-sparse multiplication. They named their protocol *SparsePIR* to emphasize its sparse nature.

4.2.2 Searchable Symmetric Encryption (SSE)

While PIR enables private queries over public data, some applications also require outsourcing the storage of sensitive data. In this case, the problem is different:

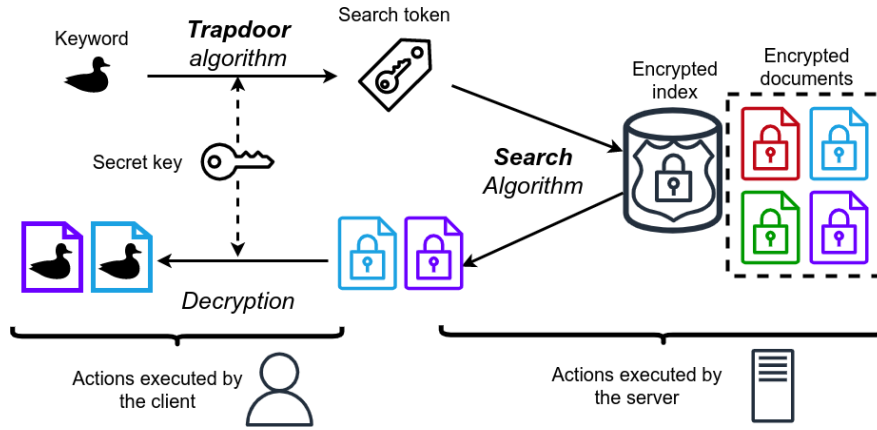


Figure 4.4 / SSE-based query protocol

the server must store encrypted data and still allow efficient search operations. This problem is addressed by SSE.

In an SSE protocol, a client outsources the storage of an encrypted dataset while keeping the ability to query it. Queries are also more expressive than in PIR: instead of retrieving a single item by identifier, the client may ask for all documents containing a keyword w . A typical SSE construction, described in [CGK006], relies on the client providing the server with encrypted documents and an encrypted index. This index enables the server to execute search queries without learning the keywords or the underlying plaintexts (as illustrated in Fig. 4.4).

Standard SSE schemes are usually defined by four algorithms [CGK006]:

- $\text{KeyGen}(1^\lambda)$: takes as input a security parameter 1^λ and outputs a key K .
- $\text{BuildIndex}(K, D_{\text{ind}})$: takes as input the key K and a dataset D_{ind} . For each $d \in D_{\text{ind}}$, let $\text{id}(d)$ be its identifier, which is assumed to leak no information about d . This algorithm builds an inverted index, encrypts it using K , and outputs the encrypted index $\mathcal{I}$, which is uploaded to the server.
- $\text{Trapdoor}(K, w)$: takes as input the key K and a keyword w , and outputs a search token td_w .
- $\text{Search}(\mathcal{I}, td_w)$: takes as input the encrypted index $\mathcal{I}$ and a search token td_w , and outputs the collection of encrypted documents matching the query. The client decrypts them locally.

The algorithms KeyGen , BuildIndex , and Trapdoor are executed by the client, while Search is run by the server. The server thus stores only encrypted documents and the encrypted index, with no direct access to plaintext.

This description covers the classic single-keyword setting. Many works have since extended it to support richer queries, such as conjunctive keyword search [DDHP22], range queries [MFET23], or searches over more complex data structures like graphs [FGPT24].

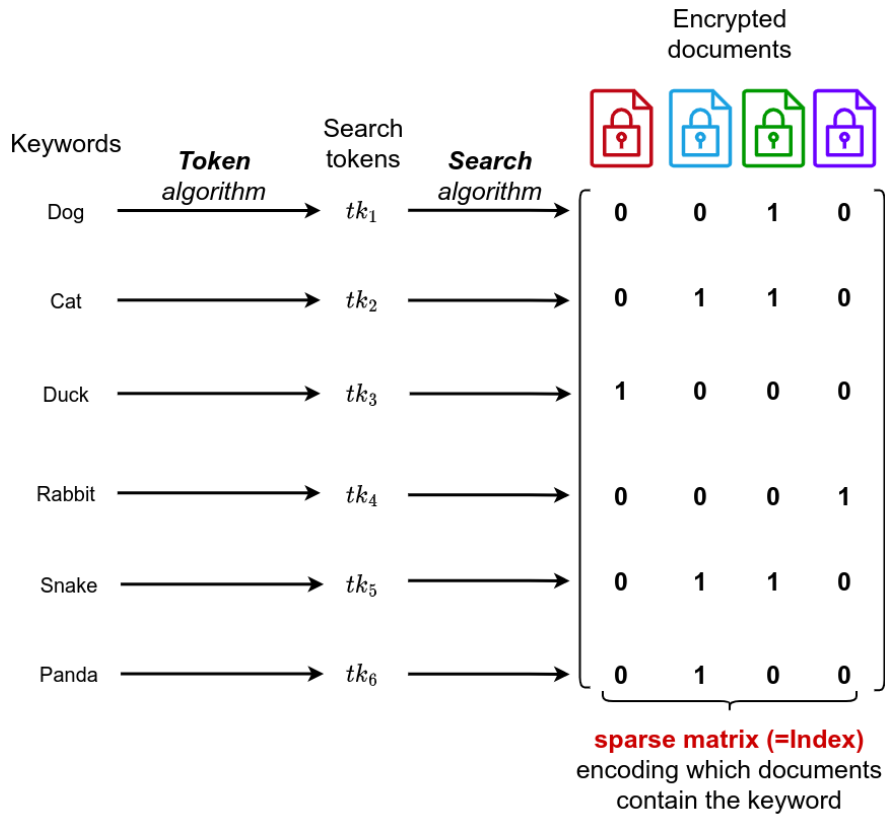


Figure 4.5 / SSE index represented as a sparse matrix

Link with sparse computations Like PIR, SSE can also be reformulated in terms of sparse data structures. The encrypted index corresponds naturally to a sparse matrix (see Fig. 4.5), where rows represent query tokens and columns represent encrypted documents. A search query then performs a form of sparse matrix multiplication, recovering the identifiers of documents that match the queried keyword.

SSE achieves performance that makes it practical for real-world deployment, but this efficiency comes at a cost: leakage [BGW24]. We will detail this issue in Section 4.2.4.

4.2.3 Oblivious RAM (ORAM)

Another connection between sparse computations and encrypted search arises with ORAM [G096]. The goal of ORAM is to provide a RAM-like structure in which read and write operations can be performed securely, without leaking memory access patterns. An ORAM is usually defined by two algorithms [AKL⁺20]:

- $\text{Init}([1^\lambda])$: initializes the RAM structure, optionally taking as input a security parameter if the protocol relies on computational hardness.
- $\text{Access}(\text{op}, \text{addr}, \text{data})$: performs a read or write operation $\text{op} \in \{\text{read}, \text{write}\}$ at address addr , with data in the case of a write.

For many years, ORAM was historically seen as a theoretical construction, as no practical implementation was known. Over the past two decades, significant progress has been made, notably in Distributed ORAM (DORAM) [DS17,FOSZ23,L013], where the RAM is secret-shared across multiple parties. Leveraging advances in MPC (e.g., FSS [DS17]), these works achieved much more efficient constructions, though ORAM remains an expensive primitive in practice.

ORAM is highly generic and can be used to implement many other cryptographic tasks, including searchable encryption [GMP15]. By simulating a secure RAM, it provides a straightforward way to implement any algorithm with data-dependent memory accesses. Yet, the overhead is often prohibitive, and many practical schemes deliberately avoid ORAM in favor of specialized protocols (e.g., in searchable encryption [Bos16]).

Despite this cost, ORAM is attractive for its strong security guarantees. Unlike most efficient SSE schemes, ORAM hides both search and access patterns (see Section 4.2.4), eliminating leakage that is otherwise hard to avoid. This again illustrates the common trade-off between efficiency and privacy in encrypted search.

Link with sparse computations Finally, we can link ORAM more directly to sparse computations. On the one hand, a read or write operation can be represented by a one-hot encoded vector, exactly as in PIR (see Fig. 4.3). This connection is made explicit in DORAM constructions such as [DS17], where DPFs are used to implement ORAM operations; a DPF being essentially an efficient way to secret-share a one-hot encoded vector¹. On the other hand, many plaintext algorithms on sparse data involve memory accesses that depend on the positions of non-zero elements. Thus, such algorithms could be naively implemented in MPC using DORAM. This link was already noticed in [DS17], which listed “sparse matrix methods” among potential applications of DORAM.

4.2.4 Leakage in encrypted search

As mentioned before, leakage is a concept that has been formalized in encrypted search and which interest (in terms of performance) is accepted. The literature distinguishes four main types of leakage:

- **Access pattern leakage:** reveals which documents are returned by a query.
- **Search pattern leakage:** reveals whether two queries are identical.
- **Volume pattern leakage:** reveals the number of documents returned by a query.
- **Trivial leakage:** reveals the database size or the total number of queries.

¹A one-hot vector of length N is equivalent to a point function with domain size N .

	Only trivial leakage	Private database	Efficiency
PIR	✓	✗	Medium
SSE	✗	✓	High
ORAM	✓	✓	Low

Table 4.6 / *Simplified view of the privacy-efficiency trade-off in encrypted search*

Access and search pattern leakage are the most common in the SSE literature. Volume pattern leakage is often treated as a minimum in settings where eliminating all leakage would make protocols prohibitively expensive (as in ORAM). Finally, trivial leakage is sometimes not even counted as leakage, since outsourcing encrypted data storage necessarily exposes the dataset size and query count to the server.

PIR and ORAM are usually associated with no leakage beyond the trivial kind. Both hide access and search patterns, and in their standard form they avoid volume pattern leakage as well, since each query always returns exactly one fixed-size document. However, ORAM-based protocols (e.g., in SSE [BKM20]) may still leak volume pattern depending on the overall design.

In contrast, SSE schemes typically leak access and search patterns. Several works try to reduce this leakage, for instance limiting it to volume pattern only [BKM20]. Still, leakage in SSE is not just incidental but often essential for achieving efficiency [BGW24], making leakage minimization a difficult task.

Besides leakage minimization, some works in SSE [CGPR15, XYW⁺19] have proposed leakage *mitigation* techniques. Instead of avoiding the leakage in a scheme, these works “neutralize” the leakage by making it unexploitable. For example, [CGPR15] proposed to include false-positive entries in the index to lower the accuracy of leakage-abuse attacks.

This tension illustrates the central trade-off in encrypted search. On one side, ORAM offers the strongest privacy guarantees but at very high cost. On the other, SSE provides practical efficiency but at the expense of additional leakage. Real-world systems must navigate this balance: in many applications, some leakage is the price of practicality. Table 4.6 summarizes this trade-off in a simplified form.

4.3 Volume pattern leakage, an acceptable leakage?

Building on insights from encrypted search, we can revisit the leakage profiles identified for secure sparse computations in Section 4.1.

From encrypted search to sparse computations While Section 4.2 showed how encrypted search problems can be expressed as sparse computations, we can push this parallel further by aligning their leakage profiles.

Recall from Fig. 4.5 that an SSE index can be represented as a sparse matrix: columns correspond to documents, rows to keywords/queries, and non-zero entries to document identifiers returned by a query. Under this interpretation, the parallels are straightforward:

- Trivial leakage $\rightarrow$ matrix dimensions.
- Volume pattern leakage $\rightarrow$ number of non-zeros (globally, per row, or per column).
- Access pattern leakage $\rightarrow$ positions of the non-zeros.
- Search pattern leakage $\rightarrow$ revealing which rows are equal (no protocol yet relying on such leakage).

Volume pattern leakage in secure sparse computations Some recent works [YCF⁺25] explicitly avoid leaking non-zero positions (i.e., access patterns), but without providing formal arguments supporting this choice. By leveraging the equivalence with encrypted search, we gain stronger justification: in SSE, access pattern leakage is known to enable attacks (including the one we describe in Chapter 6), whereas volume pattern leakage alone typically enables only weaker attacks [BKM20].

This suggests that volume pattern leakage is a reasonable target for secure sparse computations. Unlike SSE, where reducing leakage to the volume profile is expensive [BKM20], existing MPC works [SGRP19, YCF⁺25] (and our protocols in Chapter 8) show that efficient performance is still achievable with volume pattern leakage only.

Public sparsity knowledge in ML Another argument comes from the application domain. In ML, pipelines usually begin with exploratory data analysis (necessary to choose appropriate models), during which sparsity is routinely estimated to guide model selection. Thus, the number of non-zeros (globally or per row) is information that practitioners would often know in advance, independently of any MPC protocol.

Summary of arguments Taken together, these observations support volume pattern leakage as the acceptable leakage profile for sparse MPC:

- (1) A form of leakage is necessary to efficient MPC protocols on sparse data (as argued in Section 4.1).
- (2) Volume pattern leakage is the minimal non-trivial leakage profile.
- (3) Sparsity statistics are often needed by practitioners (e.g., in ML pipelines), and thus naturally public.

One could also argue that volume pattern leakage reveals little about individuals. Real-world sparse data often follow predictable distributions (e.g., power laws),

making global sparsity levels relatively predictable. However, this remains an informal argument. A rigorous way to validate it would be to design and test attacks that rely solely on sparsity information; an open problem for future work.

Leakage mitigation As mentioned in Section 4.2, techniques exist to mitigate leakage in encrypted search. Chapter 8 will rely on similar approaches to mitigate the volume pattern leakage in our secure sparse matrix multiplications.

5 How to securely shuffle? A survey about secure shufflers for privacy-preserving computations

This first contribution of the thesis is a bit special as it does not introduce new results. Recent research has highlighted the role of secure shufflers in providing “privacy amplification” for computations like sparse aggregation. However, existing works often treat secure shufflers as black boxes, overlooking their vulnerabilities and costs. Existing works remained quite vague on the implementations, but it is evident that some existing secure protocols could satisfy the requirements of those applications papers. Instead of designing yet another secure protocol, this chapter addresses this by identifying, categorizing, and comparing 26 secure shuffling protocols. We provide guidelines for selecting appropriate shufflers and highlight promising directions for future research.

This chapter is drawn from [DHPR25c].

5.1 Introduction

The literature has described many approaches to “enhance” privacy in privacy-preserving computations. Among all the approaches, secure shufflers play an important role as they offer at the same time: accurate results and high privacy guarantees, *including with sparse data* [LWY22]. A secure shuffler is a cryptographic protocol (involving one or multiple servers) that takes in messages from a group of senders, shuffles them, and forwards them to a receiver without revealing the origin of each message.

This line of work originates from [IKOS06] that proposed, in 2006, secure shufflers to perform privacy-preserving data aggregation: sending secret shares to a secure shuffler that will anonymize their origin, so any (untrusted) party can safely sum the shares and reveal *only* the aggregation result. Recent works in Differential Privacy (DP) [BEM⁺17, BBN19a, BBN19b, CSU⁺19, EFM⁺19, Che21] revived this interest in secure shufflers by formalizing the “privacy amplification” they provide. Based on DP analysis, shuffling-based private computations offer privacy-utility trade-off close to private computations performed by a trusted third party (i.e., a trusted party offering the optimal trade-off in DP). Thus, secure shufflers have been presented as an efficient solution to build privacy-preserving computations without requiring a trusted party.

Secure shufflers have found applications in numerous fields from privacy-preserving software monitoring [BEM⁺17] to Federated Learning (FL) [KMRR16, KMY⁺17]; a decentralized ML paradigm in which data owners collaboratively without sharing

their personal data. In these applications, the shufflers enable to compute efficiently and privately a form of data aggregation without revealing the individual values. Recently, [WDS⁺25] broadened the scope of applications as they highlighted the potential of secure shufflers in “private individual computations” (e.g., taxi hailing).

Most papers introducing such shuffling-based computations (e.g., [GPV19, CSU⁺19]) use an abstract shuffler rather than a concrete implementation. An advantage of considering an abstract shuffler is to focus on the statistical properties, which are relevant for the ML/statistics applications, and plug in later a concrete shuffler implementation. However, not all secure shufflers are equally appropriate for all applications, as shuffler and application papers make their own explicit or implicit security assumptions, which must be compatible.

Understanding of the properties of secure shuffler implementations would have several benefits. First, it allows for a fairer comparison between private computation algorithms. For example, existing papers do not tell whether shuffling-based data aggregation is more scalable than masking-based data aggregation [MJÖC23], since most papers on shuffling-based computations do not detail their shuffler implementation. Second, it contributes to a better formalization of the concept of “secure shuffler”. The literature relies on a vague definition leaving some blind spots; e.g., several papers [GPV19, CSU⁺19, KMA⁺21] present mix networks as an example of secure shufflers, without mentioning the possible de-anonymization and censorship attacks [SW06, SEV⁺15]. Third, it could lead to more consensus on what technique is appropriate in which use cases. For example, [GPV19] present mix nets as “highly scalable” while [GKLX22] point at the same shufflers as too expensive, which may seem contradictory without a deeper understanding of the context.

Implementing an ideal shuffler (as defined in the theory) is not straightforward. [BKM⁺22] pointed out the **need for a thorough understanding of secure shuffler implementations** as they highlighted that only one DP paper [BEM⁺17] detailed their implementation. Our survey comes to fill this gap.

Our contributions

- (1) Identify and **compare 26 existing secure protocols** fulfilling the minimum requirements of a secure shuffler.
- (2) Adapt their original security definitions to a **uniform set of security properties** that supports our comparison.
- (3) Survey the applications of secure shufflers in privacy-preserving computations.
- (4) Identify several **open problems** (e.g., robustness to data poisoning).
- (5) Present **guidelines** to choose a shuffler based on the properties of a privacy-preserving application.

Terminology In privacy-preserving computations, we typically have a group of data owners $\{P_1, \dots, P_N\}$ (each owning a private value x_i) willing to compute a function $f(x_1, \dots, x_N)$. However, they want to reveal as little information as possible. Our survey focuses on privacy-preserving computations during which the data owners are supported by a “secure shuffler” to preserve privacy.

A secure shuffler $\mathcal{S}$ is a set $\{S_1 \dots S_M\}$ of M shuffling servers. Each data owner can send none, one, or more messages to $\mathcal{S}$. Let n be the total number of messages sent to $\mathcal{S}$. The shuffler $\mathcal{S}$ outputs the shuffled message set $\{m_1 \dots m_n\}$; all outputs usually transmitted to a party (e.g., a bulletin board or a data analyser) who will further process it. In particular, the set $\{m_1 \dots m_n\}$ contains the messages in random order; i.e., does not reveal the sender of any of the messages. Section 5.2 further formalizes this concept.

5.2 Secure shuffler definition

This section formalizes the security properties desirable for a secure shuffler.

5.2.1 Basic definition

Our starting point is the definition of a shuffler $\mathcal{S}$. Based on papers studying shuffling-based DP [CSU⁺19, Che21], we propose the following naive definition:

Definition 5.1. A **Shuffler** $\mathcal{S} : \{0, 1\}^{n \times L} \rightarrow \{0, 1\}^{n \times L}$ takes as input a collection of messages $(m_1 \dots m_n)$ (for any $m_i \in \{0, 1\}^L$) and outputs $(\tilde{m}_1 \dots \tilde{m}_n)$, with $(\tilde{m}_1 \dots \tilde{m}_n) = \pi(m_1 \dots m_n)$ and π a uniformly randomly drawn permutation on the set $\{1 \dots n\}$.

This naive definition offers an intuitive understanding of the problem at hand, but it is too informal. On the one hand, it considers the secure shuffler as a unique and well-identified entity, while secure shuffling protocols commonly require the participation of multiple parties, notably the data owners. On the other hand, it does not reuse classic security frameworks necessary to provide formal security proofs.

Indeed, several secure shuffling techniques involve local preprocessing at the sending parties, e.g., mix networks (presented in Section 5.3.2) require performing multilayer encryption. While Definition 5.1 is an appropriate definition for a shuffling function executed locally, it does not capture the reality of a secure shuffling protocol, which typically involve multiple parties and algorithms. Thus, we propose to refine this definition, and characterize a shuffler using a pair of randomized algorithms $(\mathcal{P}, \mathcal{M})$:

- The local preprocessor $\mathcal{P} : \{0, 1\}^L \rightarrow \mathcal{Y}$ is executed by the sender on their message m_i .

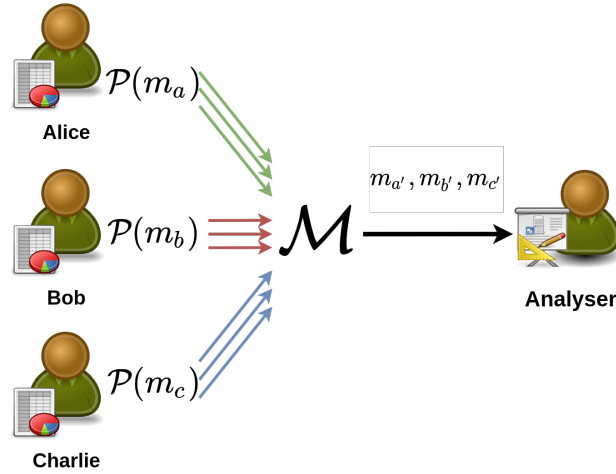


Figure 5.2 / Abstract shuffler design: (1) The senders pre-process their messages using $\mathcal{P}$, (2) They send their preprocessed messages to the mixer $\mathcal{M}$, (3) The mixer outputs a permutation of the messages, and (4) an analyser can use the shuffled messages in a privacy-preserving computation

- The mixer $\mathcal{M} : \mathcal{Y}^n \rightarrow \{0, 1\}^{L \times n}$ takes as input the pre-processed messages and outputs messages (i.e., a permutation of the input messages). The mixer is composed of M servers (where $M \geq 1$).

The local preprocessor $\mathcal{P}$ implicitly includes the communication between the senders and the mixer. The shuffle can then be expressed as $\mathcal{S}(m_1, \dots, m_n) = \mathcal{M}(\mathcal{P}(m_1), \dots, \mathcal{P}(m_n))$ (as represented in Fig. 5.2). The mixer $\mathcal{M}$ is the component that is usually considered when referring to a secure shuffler. For simplicity, we call “shuffling servers” the servers executing the mixer $\mathcal{M}$. For example, a shuffler based on a single trusted server is defined by $(\mathcal{P}_T, \mathcal{M}_T)$ with $\mathcal{P}_T(m) = m$ and $\mathcal{M}_T(m_1, \dots, m_n) = \pi(m_1, \dots, m_n)$; $\mathcal{M}_T$ is executed by the trusted server.

We have now a definition of “shuffler” but not yet of “secure shuffler”. Let us start from the following **informal** definition:

Definition 5.3. A shuffler $\mathcal{S}$ is **secure** against a threat model $\mathcal{A}$ if, for all messages $m_1 \dots m_n \in \{0, 1\}^L$, there exists a permutation π such that $\mathcal{S}(m_1, \dots, m_n) = \pi(m_1, \dots, m_n)$ where no adversary fitting the threat model $\mathcal{A}$ can infer anything about π .

This informal definition fits what the literature [KMA⁺21] implicitly considers as a “secure shuffler” (also referred to as “oblivious shuffler” in [BEM⁺17, GKLX22]). However, this definition is imprecise. Thus, we propose to formalize this notion using four new security properties to characterize a shuffler $\mathcal{S}$: **Anonymity**, **Correctness**, **Sender disruption resistance**, **Server disruption resistance**.

5.2.2 Security properties

Anonymity

This is the minimum requirement for shuffling-based private computations. More precisely, one is usually interested in sender anonymity because only the origin of the messages must remain anonymous. It is even the core concern of shuffler application papers that give more importance to the concept of “anonymous messaging” than to shuffling itself [IKOS06, CSU⁺19, GGK⁺20].

Absence of standard formalization While the first anonymization techniques [Cha81, Cha88] appeared in the 1980s, the first anonymity formalization paper [FJS07] was only published in 2007. Thus, most anonymization systems (up until now) have been built upon informal anonymity definitions. For example, even recent constructions such as Clarion [EB22] prove the security of their MPC protocol, but do not prove the anonymity itself; [EB22] does not even cite any formalization framework.

[PH10] proposed a recognized terminology for all the words related to anonymity [LDJ19]. These definitions are informal but are a commonly accepted starting point. They give the following definition for sender anonymity:

Definition 5.4. A sender P is anonymous with regard to sending if and only if P is not distinguishable within the set of potential senders.

Finding a formal definition for anonymity is a complex problem because anonymity is context-dependent. Several papers [BKM⁺13, GH13, KBS⁺19] proposed formal anonymity definitions, but they remain imperfect. For example, some of them exclude some de-anonymization attacks (e.g., traffic analysis attacks in [BKM⁺13]), which creates a blind spot in the theory that struggles to fit the reality of anonymity. Furthermore, [LDJ19] surveyed these works on anonymity formalizations and concluded that the formalization proposed “is still a theoretic approach and lacks a practical verification.” Hence, anonymity formalization is still an open problem for the security community that is still looking for a standard and general solution.

Theoretic definition Our survey does not intend to solve this long-running formalization problem. Instead, we propose a theoretic definition of the sender anonymity and then highlight five factors that can degrade this anonymity in practice. Our theoretic definition formalizes Definition 5.4 using standard security notions, especially indistinguishability.

Let $\mathcal{V}(\cdot)$ be the view/knowledge of an agent (e.g., a sender or server involved in the protocol), and let $\mathcal{H}$ be the set of honest senders. We propose the following theoretic anonymity definition;

Definition 5.5. (Anonymity) A shuffler $\mathcal{S}$ provides **anonymity against a threat model $\mathcal{A}$** if, for any message $m \in \mathcal{S}(m_1, \dots, m_n)$ (with $m_i \sim \mathcal{U}(\{0, 1\}^L), \forall i \in \{1 \dots n\}$), any PPT adversary A fitting the threat model $\mathcal{A}$, and any honest senders $P, P' \in \mathcal{H}$, we have:

$$|\Pr[m \text{ sent by } P | \mathcal{V}(A)] - \Pr[m \text{ sent by } P' | \mathcal{V}(A)]| < f(\lambda),$$

where f is a negligible function and λ a security parameter.

This definition means that an adversary can distinguish honest senders only with negligible probability. Suppose all the messages are unique, the probability of m being sent by the honest party P (i.e., $\Pr[m \text{ sent by } P | \mathcal{V}(A)]$) has two possible values: 0 if the adversary controls the sender of m , and $\frac{1}{|\mathcal{H}|}$, otherwise. If the messages are not all unique, we can have q honest senders who sent the same message m ; the probability of a (honest) P being the sender of m is $\frac{q}{|\mathcal{H}|}$.

Degrading factors Five factors can degrade this theoretic definition in practice:

- *Message content*: A message is anonymous if it is impossible to infer its sender from its content (e.g., the message “I am Alice” is straightforward to de-anonymize). This requirement is present in Definition 5.5 via the uniform distribution of the messages. This de-anonymization factor is not related to the shuffler implementation. In other words, the protocol using the shuffler must ensure that the distribution of the messages sent does not reveal their origin.
- *Metadata leakage*: In all surveyed techniques but one, a passive traffic observer can infer the size and number of messages sent by each sender. If all messages have a different size, one can link the shuffled messages to their sender by comparing the size of the sent and shuffled messages. Section 5.6.1 explains why avoiding this leakage would induce an impractical cost and discusses why it is tolerable in privacy-preserving computations.
- *Censorship*: in some systems, the adversary can censor some senders. In these cases, this decreases the quality of the anonymization because it reduces the number of potential senders. The second security property of this section (i.e., correctness) guarantees that censorship is not possible at the shuffler level. However, an adversary may still attempt a network-level attack (e.g., blocking the sender’s network traffic).
- *Uniformity of the anonymization*: for optimization purposes, some shufflers may result in non-uniform distributions over the permutation π ; i.e., the $f(\lambda)$ of Definition 5.5 is not negligible but still small enough to be useful in a number of practical applications. Section 5.6.2 discusses this relaxation.
- *De-anonymization attacks*: Section 5.2.3 presents a taxonomy of attacks

against secure shufflers. Some attacks are agnostic to the shuffler implementation, while others are implementation-specific.

Comparison with surveys on anonymous communications Our survey focuses on sender anonymity as it is the form of anonymity required by secure shufflers. However, some anonymous communication systems provide other forms of anonymity such as one-to-one anonymous communications [vdHLZZ15, ECZB21] (i.e., the two communicating agents know the identity of their correspondent, but there is a guarantee that no third party knows these agents are communicating). This is useful in a number of applications but incompatible with sender anonymity since the receiver knows the sender’s identity. Hence, such systems cannot be used as secure shufflers. Among all anonymous communication systems, only “anonymous broadcasters” satisfy the secure shuffling needs.

Over the years, several works including [EY09, SG23, STSP23, SSA⁺18] have surveyed anonymous communication systems. As [SG23] is the most recent and complete survey, we will focus our discussion on it, but our remarks hold for all surveys on anonymous communications. While both [SG23] and this chapter gravitates around the notion of anonymity, they provide different perspectives because they do not focus on the same functionality. On the one hand, they focus on anonymous communications. On the other hand, we focus on secure shuffling.

Since we focus on different functionalities, our survey relies on a different set of definitions. For example, the concept of correctness (defined in the next subsection) is not studied in [SG23]. Indeed, correctness is an important notion in privacy-preserving computations as we want to avoid a bias in the output results. The anonymous communications literature is much more interested in the notion of censorship resistance (which is implied by correctness but is not equivalent).

This scope difference has a visible consequence: while some solutions are covered in both works, some solutions are only present in [SG23] (e.g., one-to-one anonymous communications), and others are only present in our survey (e.g., TEE-based shufflers).

A core contribution of our survey is to identify secure protocols able to implement a secure shuffler, even though these protocols were not initially designed for it. While some anonymous communications systems can implement this functionality, many other secure protocols provide the sender anonymity necessary to secure shuffling. For example, Section 5.3 highlights ZKP protocols initially used for electronic voting [Adi08] and TEE solutions imagined for secure contact discovery [NDCP24].

Correctness

Next to anonymity, a good shuffler should also satisfy correctness. This property covers various security definitions present in existing security papers. On the

one hand, secure shufflers based on MPC (e.g., RPM [LK23]) prove a standard security property called “correctness” (i.e., the MPC protocol outputs the expected output). On the other hand, recent anonymous communication systems (e.g., Blinder [APY20]) often guarantee censorship resistance. This second security guarantee is essential, but it does not prevent all the threats covered by the “correctness.” Taking inspiration from MPC, we propose the following explicit definition for shuffling correctness:

Definition 5.6. (Correctness) A shuffler $\mathcal{S}$ is **correct** against threat model $\mathcal{A}$, if, in the presence of any PPT adversary $A \in \mathcal{A}$, for all uninterrupted protocol runs:

$$\Pr[\exists \pi, \mathcal{S}(m_1, \dots, m_n) = \pi(m_1, \dots, m_n)] \geq 1 - f(\lambda),$$

where f is a negligible function and λ a security parameter.

This property implies that the shuffle output contains all the input messages (and only those) because an adversary has only a negligible chance to alter the result.

Importance Shuffler correctness prevents three types of attacks (executed by malicious shuffling servers): censorship, message tampering, and message injection. In privacy-preserving computations, these attacks would impact all the users because it biases the final result. Indeed, a malicious server may try to censor a specific type of sender to bias an ML model or a statistical estimation. Hence, shuffle correctness is necessary for the overall computation correctness.

Moreover, some theoretical models (such as the Shuffle DP) rely on an abstract shuffler model fitting Definition 5.1, which implicitly requires correctness.

Formal secure shuffler definition Using the previous two security properties, we can formalize the secure shuffler definition as follows:

Definition 5.7. (Secure Shuffler) A shuffler $\mathcal{S}$ is **secure** against a threat model $\mathcal{A}$ if it **guarantees anonymity and correctness** against $\mathcal{A}$.

Sender disruption resistance

Even if we have a formal secure shuffler definition, additional security definitions can address other practical issues not covered by the intuitive definition.

Several ML works studied the resilience of distributed training protocols to malicious behaviors [CSX17, AAL18]. These works conceive protocols that succeed even in the presence of malicious or faulty agents. Protocol disruption is also possible in secure shuffling and is critical when thousands or millions of senders are involved. Controlling the honesty of millions of senders is too complex, so a

form of disruption resistance is necessary. We propose the following definition regarding the resistance to malicious senders:

Definition 5.8. (Sender disruption resistance) A shuffler $\mathcal{S}$ is **resistant to sender disruption** if malicious senders cannot force protocol abortion by sending ill-formed data or not sending data.

This property provides tolerance to drop-outs (i.e., the disconnections of senders before the protocol completion), which drew much attention in the secure aggregation literature [BBG⁺20, SBR22, BGL⁺22]. Secure aggregation protocols such as [BBG⁺20, BGL⁺22] only tolerate a fixed number of drop-outs, while sender disruption-resistant shufflers tolerate an unbounded number of drop-outs. Sender disruption resistance is stronger than drop-out tolerance since it also covers the reception of ill-formed data.

Definition 5.8 does not limit the number of malicious senders: a disruption-resistant shuffler can always produce an output. However, publishing the output is not desirable if the output size is too small to guarantee satisfying anonymity. Section 5.6.5 further discusses this issue.

Finally, Definition 5.8 specifies that the disruption can only be caused by missing or ill-formed messages to exclude Distributed Denial-of-Service (DDoS) attacks (i.e., flooding a server with a massive amount of requests to make it unavailable). These attacks (or any other network-level attack) are not specific to shufflers and can be mitigated using network security techniques [MR04].

Server disruption resistance

Finally, we can consider server disruption resistance:

Definition 5.9. (Server disruption resistance) A shuffler $\mathcal{S} = (\mathcal{P}, \mathcal{M})$ is **resistant to server disruption** if a malicious shuffling server (i.e., involved in $\mathcal{M}$) cannot force protocol abortion by sending ill-formed data or not sending data.

This property is called “robustness” in MPC protocols [APY20]. One may see it as an extra property because the number of servers is small enough to maintain a blocklist of untrustworthy servers. As for sender disruption resistance, server disruption resistance brings resilience to drop-outs.

In summary, (sender or server) disruption resistance is not necessary to define a secure shuffler but preferable for real-world applications.

5.2.3 De-anonymization attacks against secure shufflers

This subsection provides a taxonomy of de-anonymization attacks and discusses mitigation.

Intersection attacks

These attacks [WSF13,DS05] rely on the statistical analysis that can be made from several shuffle outputs to de-anonymize messages. Indeed, if a data owner participates in k shuffles, computing the intersection of the round results might leak some information about this data owner. These attacks against shufflers are specific to the use case and the data being shuffled. For example, the intersection attack is useless if the shuffled messages are secret shares, as in [IKOS06] protocol, but the intersection may de-anonymize the messages if we shuffle cleartext data.

[WSF13] propose a generic *Buddy* system compliant with any anonymization system to mitigate these attacks. The general idea is to ensure a large set of possible senders for a given message stream, so the intersection of multiple shuffles cannot lead to a de-anonymization. No result is available regarding the scalability of this generic mitigation technique for applications to privacy-preserving computations.

Trickle and flooding attacks

[SDS02] describe two general attacks (grouped under the term “blending attacks”): trickle attacks and flooding attacks.

Their idea is to force the shuffle output only to contain one honest message. The attacker can then easily infer the source of this communication. In trickle attacks, the adversary censors messages from everyone but one user. In flooding attacks, the adversary uses colluding malicious data owner to “flood” the batch with malicious messages and one honest message.

The trickle attack is only possible on a system insecure against censorship (i.e., with no correctness guarantee as defined in Section 5.2). The flooding attack is only possible if a malicious server colludes with data owners and can be mitigated using an authentication scheme, so a single malicious user cannot flood a batch alone. Some secure shufflers (e.g., mix nets) must use anonymous authentication systems [CMZ14] to mitigate these attacks.

Traffic analysis attacks

These attacks [MD05,SEV⁺15,AYR15] have been extensively documented for mix networks: a global adversary (e.g., a state agency) observes the packet transmission between the agents to trace the path of the messages. From the beginning, anonymization networks such as Tor were presented as possibly insecure against global attackers.

However, we can describe two simple countermeasures to mitigate these attacks. First, we can distribute the agents in several countries. This practice decreases the chance of having a global adversary able to observe all agents. Second, increasing the traffic makes message tracing harder.

Only mix networks is sensitive to this attack, while other secure shufflers are secure by design.

Shuffler-specific attacks

The previous paragraphs cover the main attacks affecting all secure shufflers. However, new secure solutions often bring new attack vectors, specific to the proposed technique.

For example, mix nets are sensitive to replay attacks (i.e., retransmitting a message received during a previous shuffle to infer its sender using the intersection of the shuffle outputs).

On the other hand, TEE-based solutions can be sensitive to side-channel attacks (i.e., attacks exploiting leakage from the protocol implementation, but not the protocol itself) in which the memory access patterns help de-anonymize a shuffle of some constructions [SJG22].

5.3 Comparison of the existing secure shufflers

This section identifies and compares 26 existing protocols implementing a secure shuffler. We distinguish six categories of secure shufflers (based on design similarities): mix networks, verifiable shufflers, DC nets, MPC-based shufflers, TEE-based shufflers, and other shufflers. We summarize the comparison in Tables 5.10 and 5.11, which respectively analyze scalability and security.

5.3.1 Methodology

We adopted two approaches to identify relevant works able to implement a secure shuffler. First, we performed a survey on Google Scholar using different combinations of keyword including “secure shuffle,” “anonymous broadcaster,” “verifiable shuffler,” “mix network,” “anonymous communications,” “shuffle differential privacy,” and “secret shuffle.” This first step provided a preliminary list of secure shufflers. Second, we used Google Scholar to find additional secure shufflers in papers that cited these constructions. This second step enables to identify follow-up works that may propose more unconventional solutions to securely shuffle. For example, [BBG⁺20] introduced a secure-aggregation-based shuffler, while the main focus of their paper is secure aggregation.

To classify a work as relevant, we need the proposed protocol to *at least* provide a form a sender anonymity (as defined in Section 5.2.2). This requirement is satisfied by many secure protocols from anonymous communications to MPC-based shuffling. Thus, our surveying process required examining works from various security topics including anonymous communications, secure enclaves, and electronic voting. Through this process, we identified 26 secure shufflers that we have grouped into 6 categories based on their design similarities.

<i>Technique</i>	<i>Per-Sender comm.</i>	<i>Per-Sender comp.</i>	<i>Per-Server comm.</i>	<i>Per-Server comp.</i>	<i>M</i>
Trusted shuffler	$O(1)$	$O(1)$	$O(n)$	$O(n)$	1
Mix networks ($k = \text{path length}$)					
Fixed-route [Cha81,DDM03,FM02]	$O(1)$	$O(k)$	$O(n)$	$O(n)$	≥ 1
Free-route [Cha81,DMS04,CAB ⁺ 15]	$O(1)$	$O(k)$	$O(nk/M)$	$O(nk/M)$	$\gg 1$
Advanced [CDJ ⁺ 17,PHE ⁺ 17]	$O(1)$	$O(k)$	$O(nk/M)$	$O(nk/M)$	$\gg 1$
Trellis [LSD23]	$O(1)$	$O(k)$	$O(nk/M)$	$O(nk/M)$	$\gg 1$
TORaaS [HMPS14]	$O(1)$	$O(k)$	$O(n)$	$O(n)$	1
Verifiable shufflers					
Classic [FS01,Nef01,BG12]	$O(M)$	$O(1)$	$O(M\sqrt{n} + Mn)$	$O(n \log n + Mn + M\sqrt{n})$	> 1
Riffle [KLDF16]	$O(1)$	$O(M)$	$O(n)$	$O(n)$	> 2
Atom [KCDF17] (k : group size, $\alpha = \frac{k}{M}$)	$O(k)$	$O(1)$	$O(n \cdot \alpha + k\sqrt{n \cdot \alpha})$	$O(\alpha \log \alpha)$	$\gg 1$
Lattice-based [ABGS23]	$O(M)$	$O(1)$	$O(Mn)$	$O(Mn)$	> 1
DC networks					
Dissent [CF10]	$O(Mn)$	$O(Mn)$	$O(n^2 + nM)$	$O(Mn)$	> 1
Riposte [CBM15]	$O(\sqrt{n})$	$O(\sqrt{n})$	$O(n\sqrt{n})$	$O(n^2)$	3
Spectrum [NSD22]	$O(\log n)$	$O(\log n)$	$O(n \log n)$	$O(n^2)$	2
Sabre [VSH22]	$O(\log n)$	$O(\log n)$	$O(n \log n)$	$O(n^2)$	3
MPC-based shufflers					
AsynchroMix [LYK ⁺ 19]	$O(M)$	$O(M)$	$O(M \cdot n \cdot \log^2 n)$	$O(M \cdot n \cdot \log^2 n)$	≥ 4
PowerMix [LYK ⁺ 19]	$O(M)$	$O(M)$	$O(n^2)$	$O(n^3)$	≥ 4
Blinder [APY20]	$O(M\sqrt{n})$	$O(M\sqrt{n})$	$O(n\sqrt{n})$	$O(n^2)$	≥ 5
Clarion [EB22]	$O(M)$	$O(M)$	$O(Mn)$	$O(Mn)$	≥ 3
RPM [LK23]	$O(M)$	$O(M)$	$O(Mn\sqrt{n})$	$O(Mn^2)$	≥ 3
Ruffle [AKK ⁺ 23]	$O(M)$	$O(M)$	$O(Mn)$	$O(Mn)$	$= 3$
TEE-based shufflers					
Prochlo [BEM ⁺ 17]	$O(1)$	$O(1)$	$O(n)$	$O(n)$	1
ORShuffle [SJG22]	$O(1)$	$O(1)$	$O(n)$	$O(n \log^2 n)$	1
Butterfly Shuffle [GWC ⁺ 23]	$O(1)$	$O(1)$	$O(n)$	$O(n \log n)$	1
DBucket [NDCP24]	$O(1)$	$O(1)$	$O(n)$	$O(n \log n)$	1
Other shufflers					
Rumor-based [FKO ⁺ 17] ($\alpha = \text{node degree}$)	$O(\alpha \cdot n)$	$O(\alpha \cdot n)$	$O(\alpha \cdot n)$	$O(\alpha \cdot n)$	≥ 1
NIDAR [BHMS21]	$O(1)$	$O(1)$	$O(n)$	$O(n^{1+1/2})$	1
Sec.-Agg.-based [BBG ⁺ 20]	$O(\log^2 n + n \log n)$	$O(n \log^2 n)$	$O(n^2 \log n)$	$O(n^2 \cdot \log^2 n)$	1

General notations: n = number of messages, M = number of shuffling servers.

Simplifications: Message size is $O(1)$. Security parameter λ is $O(1)$. Setup costs not included.

Table 5.10 / Scalability comparison of the secure shuffling solutions

Technique	M	Anonymous vs. ...malicious servers	Disruption resistance		Correct
			Sender	Server	
Mix networks (k = path length)					
Fixed-route [Cha81,DDM03,FM02]	≥ 1	$M - 1$	✓	✗	✗
Free-route [Cha81,DMS04,CAB ⁺ 15]	$\gg 1$	$k - 1$ per path	✓	✓	✗
Advanced [CDJ ⁺ 17,PHE ⁺ 17]	$\gg 1$	$k - 1$ per path	✓	✓	✗
Trellis [LSD23]	$\gg 1$	$f\%$	✓	✓	✓
TORaaS [HMPS14]	1	$k - 1$ per path	✓	✓	✗
Verifiable shufflers					
Classic [FS01,Nef01,BG12]	> 1	$M - 1$	✓	✓	✓
Riffle [KLDF16]	> 2	$M - 2$	✗	✗	✓
Atom [KCDF17] (k : group size, $\alpha = \frac{k}{M}$)	$\gg 1$	$k - 1$ per group	✓	✓	✓
Lattice-based [ABGS23]	> 1	$M - 1$	✓	✓	✓
DC networks					
Dissent [CF10]	> 1	$M - 1$	✗	✗	✗
Riposte [CBM15]	3	1 out of 3 servers	✓	✗	✗
Spectrum [NSD22]	2	$M - 1$	✓	✗	✓
Sabre [NSD22]	3	$M - 1$	✓	✗	✓
MPC-based shufflers					
AsynchroMix [LYK ⁺ 19]	≥ 4	$t < M/3$	✓	✗	✓
PowerMix [LYK ⁺ 19]	≥ 4	$t < M/3$	✓	✗	✓
Blinder [APY20]	≥ 5	$t < M/4$	✓	✓	✓
Clarion [EB22]	≥ 3	$M - 1$	✓	✗	✗
RPM [LK23]	≥ 3	$t < M/2$	✓	✓	✓
Ruffle [AKK ⁺ 23]	$= 3$	$t < M/2$	✓	✓	✓
TEE-based shufflers					
Prochlo [BEM ⁺ 17]	1	1	✓	✗	✓
ORShuffle [SJG22]	1	1	✓	✗	✓
Butterfly Shuffle [GWC ⁺ 23]	1	1	✓	✗	✓
DBucket [NDCP24]	1	1	✓	✗	✓
Other shufflers					
Rumor-based [FKO ⁺ 17] (α = node degree)	≥ 1	t_α malicious agents out of $N + M$	✓	✓	✓
NIDAR [BHMS21]	1	1	✗	✗	✗
Secure-Aggregation- based [BBG ⁺ 20]	1	1	✗	✗	✓

General notations: n = number of messages, M = number of shuffling servers.

Table 5.11 / Security comparison of the secure shuffling solutions

Complexity analysis To compare the scalability of the techniques, we systematically present communication and computation costs for both the servers and the data owners. In Table 5.10, the “per-sender” costs correspond to the (data owner) cost for one message sent. The “per-server” costs correspond to the cost paid by each server to shuffle the whole batch of n messages. We emphasize the scalability in the number of messages n and the number of servers M .

We make a few simplifying assumptions. First, we assume the message size L to be $O(1)$, but Section 5.3.8 analyses the scalability with the message size. Second, we assume that any security parameter λ is $O(1)$; this parameter is present in cryptographic schemes to tune their security (i.e., larger implies more secure). Third, the cost of the protocol setup is not part of the complexity analysis, because it is usually a one-time process.

Security analysis For each technique, we systematically identify which security properties are satisfied: anonymity, correctness, sender disruption resistance, and server disruption resistance.

As motivated in Section 5.2, analysing a protocol anonymity is complex, so we focus three distinguishing criteria: threat model, anonymity distribution, and de-anonymization attacks. Table 5.11 has no dedicated column for anonymity distribution and de-anonymization attacks because these “degrading factors” only concern a few solutions. The full version of this chapter [DHP25c] contains a more complete version of Table 5.11 which highlights other specificities of the listed solutions such as non-uniform distribution or known de-anonymization attacks. We cannot include this more complete version due to space issues.

Contrary to anonymity, correctness and disruption resistance are binary properties that are either present or absent (represented by check and cross marks in Table 5.11).

5.3.2 Mix networks (mix nets)

Mix networks are a well-known anonymization technique thanks to Tor [DMS04] (a popular mix network used notably by activists and journalists for anonymous Web browsing). The main idea of mix networks [Cha81] is to make a packet transit through several relay nodes before reaching its destination. The packets are “onion-encrypted”: one layer of encryption is added for each relay in the path. Upon receiving the packet, each relay node removes one layer of encryption and forwards the decrypted payload to the next relay. This mixing process prevents the relay nodes from linking the sender to the receiver. For secure shuffling, all the messages are sent to the same agent (i.e., the analyzer or a public bulletin).

This anonymization technique is lightweight (k public-key encryptions for a k -node path) and adds a small amount of latency due to the relays. We distinguish free-route networks (Fig. 5.12b) where the sender freely chooses the path from

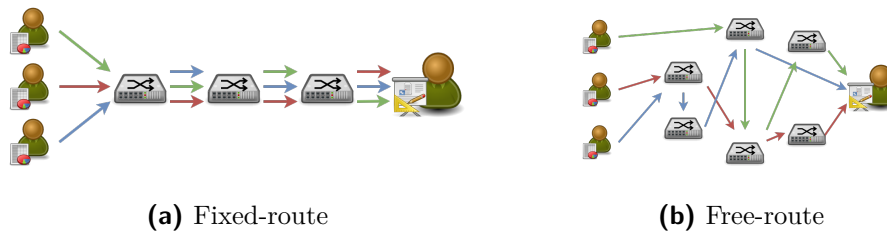


Figure 5.12 / Mix networks

fixed-route networks (Fig. 5.12a) where the path is pre-defined. In the latter, the number of relays usually equals the number of servers.

Remark about semi-trusted shuffler. Some papers, such as [HAVK22], use a semi-trusted shuffler that acts as a proxy. A mix network being a proxy chain, the semi-trusted shuffler is a 1-node mix network with a non-colluding receiver.

We compare five types of mix networks: fixed-route, free-route, advanced, Trellis, and TORaaS.

Scalability

Mix networks are highly scalable because they require only lightweight encryption and decryption. Some of them can even balance the load across the servers. The sender cost is affordable even with resource-constrained devices. Finally, the popularity of Tor is a concrete proof that it works at scale.

Security

Mix networks are anonymous as long as at least one server per relay path is honest. They suffer from two types of de-anonymization attacks based on their design: traffic analysis attacks (i.e., a global adversary observing to trace packets path) and replay attacks (i.e., a node forwarding twice the same packet and observing which message appears twice in the shuffle output). While some mix nets resist traffic analysis attacks, replay attacks are a general problem that can be mitigated using an anonymous authentication system.

Mix networks (except Trellis) guarantee no correctness because any relay node can drop packets (and then censor some of the senders). However, they have sender disruption resistance.

Noticeable solution: Tor as a Shuffler (TORaaS)

Let us analyze a deployed secure shuffler that was never formally introduced in literature but implicitly present in some papers: TORaaS (i.e., shuffling through the Tor anonymization network). In TORaaS, we use the existing Tor network instead of deploying a new mix network dedicated to the shuffling task. The

shuffler traffic is mixed with all Tor traffic. The practitioner only needs to deploy a Web server accessible via Tor.

We distinguished this solution from other free-route mix networks because Tor has some specific and well-documented flaws which are important to discuss. This solution is so simple and attractive that it must be analyzed independently so a reader can choose it safely.

While this solution is attractive due to its simplicity, practitioners should carefully consider the possibility of de-anonymization attacks [AYR15]. Even though there exists mitigation techniques, Tor is still particularly sensitive to such attacks as it is a free-route mix network.

5.3.3 Dining Cryptographers networks (DC nets)

These secure shufflers take inspiration from the Dining Cryptographers problem [Cha88, GJ04]. This anonymity problem is named after a scenario where three cryptographers are dining together and learn that their meal has been paid for, but they want to know whether one of them paid or if it was the NSA; without revealing who paid if it was one of them.

The solution proposed by [Cha88] is to exchange complementary binary masks between the participants (as represented in Fig. 5.13). Then, each participant publishes a masked value. Once aggregated, the masked values reveal the anonymous bit of information. This masking trick was reused as a foundation for several anonymous protocols referred to as “DC nets”. Most anonymization systems based on DC nets complement DC nets with other cryptographic solutions to combine scalability and security. Recently, [STSP23] wrote a survey focused on DC nets.

We identified four major DC nets: Dissent [WCFJ12], Riposte [CBM15], Spectrum [NSD22], and Sabre [VSH22].

Scalability

Until 2012 [WCFJ12], DC nets had been considered a purely theoretical construction compared to the highly scalable mix networks. Dissent [WCFJ12] marked a turning point because practical applications could be imagined, but scalability was still very limited. Recent solutions, such as Riposte [CBM15] and Spectrum [NSD22], are the first scalable DC nets. Riposte was the first anonymous broadcaster able to shuffle 1M messages within a few hours. It is not as scalable as mix networks but requires fewer servers (i.e., 2 or 3).

Security

DC nets were initially proposed to offer theoretically proven anonymity. Dissent is the only surveyed technique hiding the size and number of sent messages by each user. This leakage avoidance (and its cost) is discussed in Section 5.6.1.

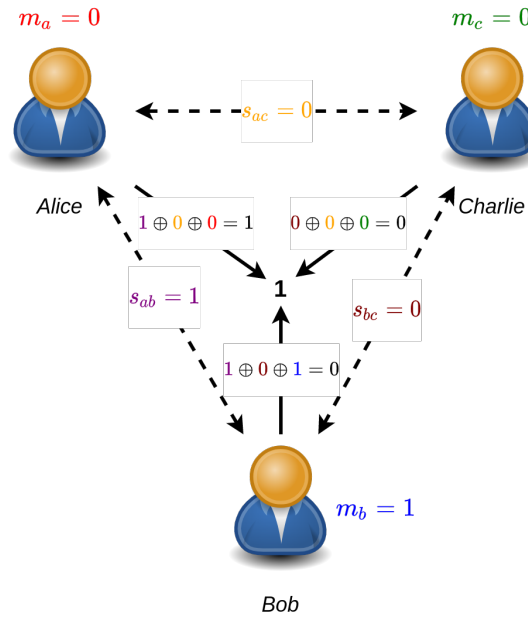


Figure 5.13 / Dining Cryptographers problem

Each DC net has a rather unique architecture, so they do not share many common points, except their foundation: the DC problem. For simplicity, we only highlight the properties of the state-of-the-art method: Spectrum [NSD22]. It is most optimized for two servers and is anonymous against one malicious server. There are no known de-anonymization attacks specific to Spectrum (or DC nets in general). Spectrum has correctness and is resistant to sender disruption only.

5.3.4 Verifiable shufflers

Verifiable shufflers are also called verifiable mix nets because they can be represented as fixed-route mix networks (see Fig. 5.12a) where each node must prove it performed a shuffle. Verifiable shufflers are a line of work distinct from mix networks because they focus on the shuffle proof more than on the network architecture. For example, the encryption schemes and the applications are very different. Verifiable shufflers are usually small networks of mix nodes performing expensive computations (i.e., proofs and verifications).

For each shuffle, a mix node produces a Zero-Knowledge Proof (ZKP), proving the correctness of the shuffle. Before the shuffle, the mix node commits to a secret random permutation and the proof shows that this random permutation was respected. Verifiable shufflers batch the incoming messages before applying the permutation. This batching protects against traffic analysis attacks.

For more details about verifiable shufflers, we recommend the survey paper of [HM20].

We identify four major verifiable shufflers in the literature: Classic verifiable

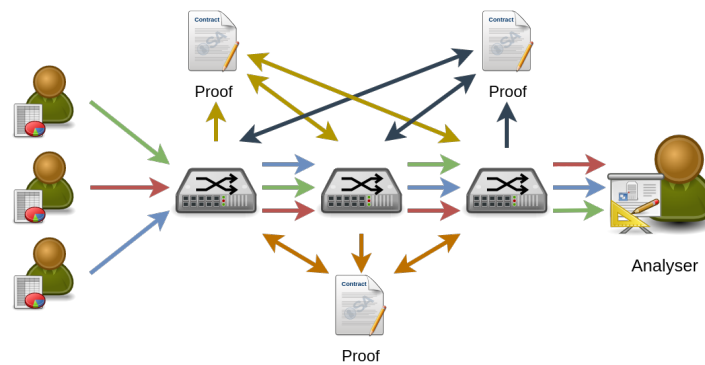


Figure 5.14 / Classic verifiable shufflers

shufflers [BG12], Riffle [KLDF16], Atom [KCDF17], lattice-based verifiable shuffle [ABGS23].

We use [BG12] as reference for classic verifiable shuffling, but some recent works [HPP20, FLSZ17] have described various optimizations such as homomorphic proofs [HPP20] (making the verification cost constant in the number of nodes in the network).

Recently, [BBG23] used verifiable shufflers to build an “alternating shuffler.” Their protocol splits the messages to shuffle into several subsets and shuffle them independently. An alternating shuffler outputs a permutation that is imperfect, but good enough to provide DP guarantees.

Scalability

The classic verifiable shufflers [BG12] have some scalability bottlenecks (i.e., cannot scale above tens of thousands of messages), but it does not prevent concrete use cases. However, the most recent adaptations of verifiable shufflers to anonymous communications, namely Riffle [KLDF16] and Atom [KCDF17], do not have this scalability issue. For example, Atom can shuffle a million message in less than an hour.

Security

They are anonymous as long as one of the servers is honest. Thanks to the ZKP, correctness is guaranteed contrary to mix networks. Only Riffle is not resistant to sender disruption, while Atom is the only verifiable shuffler with server disruption resistance. No de-anonymization attack specific to verifiable shufflers is known.

5.3.5 MPC-based shufflers

An MPC-based shuffler is a group of M servers that securely collaborate to output shuffled messages. The data owners, willing to anonymously broadcast a message,

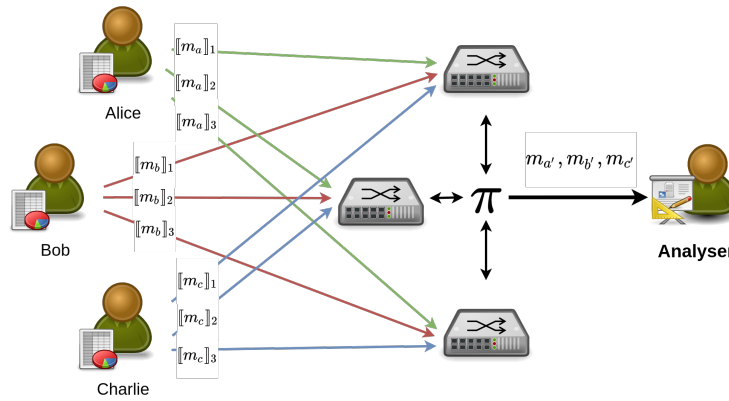


Figure 5.15 / MPC-based secure shufflers (π : an MPC protocol)

send a share of their secret message to each server. The servers then use an MPC protocol to obtain the shuffled messages.

We identified six major MPC-based shufflers in the literature: AsynchroMix [LYK⁺19], PowerMix [LYK⁺19], Blinder [APY20], Clarion [EB22], RPM [LK23], and Ruffle [AKK⁺23].

Some of these shufflers (e.g., Clarion) are built upon generic “secret-shared shuffle protocols.” This generic building block could be further optimized thanks to recent advances [CGP20, GZZ24, SYB⁺24].

Scalability

PowerMix and AsynchroMix hit their scaling limits at a few thousand messages. However, the most recent solutions (Blinder, RPM, Clarion, and Ruffle) can scale up to millions of shuffled messages (at a time). Blinder even leverages GPUs to speed up the process.

Security

All these techniques (except Clarion) share similar security properties: anonymous if there is a majority of honest servers, correctness, and sender disruption resistance. Blinder even has server disruption resistance. On the other hand, Clarion only has sender disruption resistance (i.e., no correctness and no server disruption resistance), but it is anonymous against $M - 1$ malicious servers. No de-anonymization attack specific to these solutions has been yet described.

5.3.6 TEE-based shufflers

[BEM⁺17] introduced the first TEE-based secure shuffler. The only step to conceive a TEE-based secure shuffler is to provide a shuffle code executable on the chosen architecture (e.g., Intel SGX [Int15]). Shuffling requires simple operations, but the implementation must be oblivious: the server owner should not be able

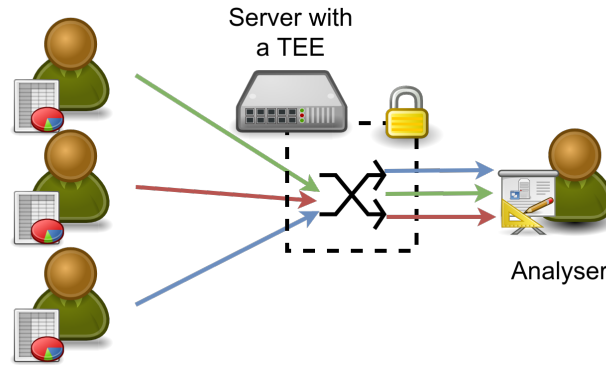


Figure 5.16 / TEE-based secure shufflers

to infer information about the data based on the memory access patterns. In other words, TEE-based shufflers behave as a proxy in a secure environment that guarantees data confidentiality. We identified four TEE-based secure shufflers: Prochlo [BEM⁺17], ORShuffle [SJG22], the Butterfly Oblivious Shuffle [GWC⁺23], and DBucket [NDCP24].

Scalability

Prochlo uses a “stash shuffle” with an asymptotic cost equivalent to a trusted shuffler (e.g., $O(n)$ computations and communications for the server, and $O(1)$ for the sender). ORShuffle achieves a server cost of $O(n \log^2 n)$, and the Butterfly Shuffle and DBucket $O(n \log n)$.

Security

TEE is satisfying as long as the user fully understands the implications of the trust assumption made. Section 5.6.6 discusses the strength of this assumption.

Moreover, flawed TEE implementations can be sensitive to side-channel attacks. For example, [SJG22] highlighted that Prochlo is sensitive to some recent attacks.

In the absence of side-channel attacks, TEE-based shufflers are anonymous against a malicious server. They satisfy shuffle correctness and sender disruption resistance. However, there is no server disruption resistance since the server owner can disconnect the server to disrupt the shuffle.

5.3.7 Other shufflers

Finally, we identified three other secure shufflers that do not fit in any of the previous categories: rumor-based shuffler [FK0⁺17], NIDAR (Non-Interactive Differentially Anonymous Router) [BHMS21], and secure-aggregation-based shuffler [BBG⁺20].

<i>Technique</i>	<i>Per-Sender</i>		<i>Per-Server</i>	
	<i>Comm.</i>	<i>Computations</i>	<i>Comm.</i>	<i>Computations</i>
Classic. verif. shufflers, Atom	$\mathcal{C} \cdot O(L)$	$\mathcal{C} \cdot O(L^3)$	$\mathcal{C} \cdot O(L)$	$\mathcal{C} \cdot O(L^3)$
Riposte, Spectrum, Sabre	$\mathcal{C} + O(ML)$	$\mathcal{C} + O(L)$	$\mathcal{C} + O(nL)$	$\mathcal{C} \cdot O(L)$
Sec.-Agg.-based	$\mathcal{C} + O(nL)$	$\mathcal{C} + O(nL \log n)$	$\mathcal{C} + O(n^2L)$	$\mathcal{C} + O(n^2L \log n)$
Rest of the shufflers	$\mathcal{C} \cdot O(L)$			

Notations: L is the message size, $\mathcal{C}$ is the “original cost” of the technique (see Table 5.10).

Table 5.17 / Scaling of the surveyed techniques with the message size

Both rumor-based and secure-aggregation-based solutions are present in this survey for completeness. They are not practical and offer no scalability or security improvements compared to the rest of the secure shufflers surveyed in this chapter.

NIDAR is particularly interesting because it is a single server solution (based on a cryptographic primitive called “Functional Encryption”) that does not require a secure enclave. Its scalability and security are imperfect, but it opens this direction with promising first results, especially its DP-inspired relaxation of anonymity definition. The main weaknesses of NIDAR are the lack of sender disruption resistance and the need for a trusted setup.

5.3.8 Scalability with the message size

This last comparison subsection studies the scalability of the techniques with the message size because message sizes can vary significantly in privacy-preserving computations: from small integers to large vectors of floating-point numbers. Most techniques have a linear dependency on the message size, but a few have different relationships.

The comparison is summarized in Table 5.17. We denote the original cost of a given technique as $\mathcal{C}$. Hence, a cost of $\mathcal{C} \cdot O(L)$ means all the original costs are multiplied by $O(L)$.

The computation costs of proof and verification protocols usually correspond to the number of modular exponentiations needed. Since modular exponentiations have a cubic cost, it induces a $O(L^3)$ dependency in classic verifiable shufflers [BG12] and Atom [KCDF17].

All the DC-nets have a linear dependency on the message size. For Riposte, Spectrum, and Sabre, an additive term (linear in the message size) appears because of a cryptographic primitive they use (i.e., Function Secret Sharing [GI14]). All the TEE-based and MPC-based shufflers have a linear dependency on the message size. Among the other shufflers, only secure-aggregation-based shufflers have a non-linear dependency on message size.

5.4 Applications to privacy-preserving computations

5.4.1 Basic shuffling-based aggregation

[IKOS06] introduced the first shuffling-based private computation. In particular, they presented a shuffling-based protocol to aggregate private values. Their work is the seminal work that inspired most recent works on shuffling-based private computations [BBGN19a, WDS⁺25].

The protocol begins with each data owner splitting their private value into several additive shares of the private value. The shares are individually sent (i.e., one message per share) to a secure shuffler that publishes its shuffling output. One can sum all the shuffled shares to obtain the *exact* sum of the initial private values.

[IKOS06] showed that each data owner only needs to send a logarithmic number of shares in order to have shares indistinguishable from perfect randomness. To reconstruct the input secret values based on the shuffled random shares, an adversary would need to compute an exponential number of share combinations. The shuffling combined with the share randomization then guarantees that an attacker cannot realistically harm the input privacy, but allows to efficiently perform a data aggregation while preserving input privacy.

5.4.2 Private statistics

Following the first anonymous communication systems, follow-up works [BS06] [DWA⁺23, GHJ⁺20, HMPS14] have built systems based on message anonymization to compute private aggregate statistics (e.g., mean). As detailed in Section 5.2, the sender anonymization layer used in these work can be seen as a shuffler. This equivalence between message anonymization and shuffling is common in Differential Privacy where papers alternatively use “anonymous messages” [GGK⁺21, GMPV20] and “shuffled messages” [EFM⁺20].

On the one hand, some systems [BS06, HMPS14] relied on shuffler to anonymize the message sender. For example, [BS06] built a privacy-preserving data collection system based on such message anonymization. In their system, each data owner sends their private data to the shuffler and the shuffler outputs a set of “anonymous” messages. This approach provides a weak privacy preservation as it does not mitigate any privacy leakage caused by the message content.

On the other hand, other works [DWA⁺23, GHJ⁺20] built inspired by [IKOS06]: they require each data owner to generate secret shares and to send these shares to the shuffler. In [IKOS06], the secret sharing will prevent the secret reconstruction. Thus, these works provided stronger privacy guarantees than [BS06, HMPS14] because they mitigate any privacy leakage from individual messages; only the aggregated result is revealed.

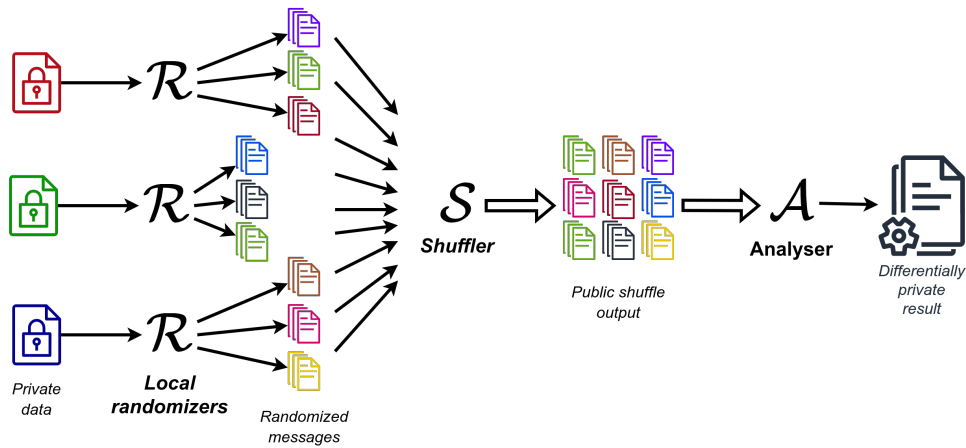


Figure 5.18 / Shuffle Differential Privacy: (1) The data owners split their private value into several random messages using the randomizer $\mathcal{R}$, (2) they transmit these messages to $\mathcal{S}$, (3) the shuffler outputs a random permutation of the messages and (4) the analyzer can perform any relevant operation on these messages

5.4.3 Differentially Private computations

While private statistics systems such as [DWA⁺23, GHJ⁺20] already provide strong privacy guarantees, works in DP propose an even stronger privacy amplification mitigating privacy leakage both from the shuffled messages and from the aggregated output.

This DP framework based on shuffler is called “the shuffle DP model” [GKMP20] and consists of two randomized algorithms (see Fig. 5.18):

- A local randomizer $\mathcal{R} : \mathcal{X} \rightarrow \mathcal{Y}$
- A shuffler $\mathcal{S} : \mathcal{Y}^* \rightarrow \mathcal{Y}^*$ that randomly permute its inputs.

The protocol $\mathcal{P} = \mathcal{S} \circ \mathcal{R}$ is (ϵ, δ) -shuffle differentially private if $\mathcal{S}(\mathcal{R}(x_1) \dots \mathcal{R}(x_n))$ is (ϵ, δ) -differentially private. The output $\mathcal{S}(\mathcal{R}(x_1) \dots \mathcal{R}(x_n))$ can then be used by any analyzer $\mathcal{A} : \mathcal{Y}^* \rightarrow \mathcal{Z}$ to provide a differentially private result (e.g., a histogram or a sum).

Several papers [BBGN19b, CCG24, EFM⁺19, FMT23] showed that the shuffling induces a privacy amplification because it requires less noise than local DP to offer comparable privacy guarantees. The privacy-accuracy trade-off of Shuffle DP approaches the trade-off of Central DP without requiring a trusted party. This good trade-off without a trusted party assumption often motivates the use of shuffle DP in privacy-preserving computations. For an extensive overview on the benefits of Shuffle DP compared to local and central DP, we refer to Cheu [Che21].

While shuffle DP is usable for any statistical computation, it provides a privacy-utility close to central DP mostly for sum operations. The first shuffle DP papers

focused on private summation [BBGN19a, BBGN20, GGK⁺20], but recent works described more complex statistical operations: private quantiles [ABG⁺25], private uniformity testing [BCJM21, Che21], private histograms [BC20, BGG⁺22, WDS⁺25] and other related computations [GGK⁺21, LWY22, HAVK22].

5.4.4 Federated Learning

As explained in Section 3.3, Federated Learning protocols can rely on secure aggregation to prevent leakage information leakage. We can implement **shuffling-based secure aggregation** in two manners. On the one hand, we can perform an “exact” secure aggregation using the protocol proposed by [IKOS06]. On the other hand, we can perform a “differentially private” secure aggregation using the DP summation mentioned in Section 5.4.3. Several FL papers have recently used this shuffle-DP aggregation [GDD⁺21, LCC⁺21, XZH24].

Following the growing interest in gradient sparsification [SCJ18], the design of secure sparse aggregations has become interesting to combining privacy with efficiency. To support such operations, [LWY22] presented a **shuffling-based sparse vector aggregation**.

5.4.5 Towards new shuffling-based applications

Previous applications all use secure shufflers to perform a form of private data aggregation. However, secure shufflers can also find applications in other types of privacy-preserving protocol.

For example, [WDS⁺25] recently proposed a shuffling-based taxi-hailing service. In this application, users willing a taxi can submit their request and location through a secure shuffler and the secure shuffler will provide DP guarantees on their location data. More generally, the authors introduced the notion of “private individual computations:” a broad range of “permutation-equivariant” computations such as taxi-hailing and information retrieval. In such protocols, each user receives a personalized service/output, while secure shufflers amplify the privacy. Thus, their work opened the way to new applications of secure shufflers.

Another recent example is the shuffling-based private information retrieval described by [GIK⁺24].

5.4.6 Applications unrelated to privacy-preserving computations

Our survey focuses specifically on application to privacy-preserving computations as they are the protocols that revived the interest in secure shufflers. However, there exist other “historical” applications that we do not survey: web anonymization [DMS04], anonymous messaging [SG23], electronic voting [Nef01], and blockchain operations [RMK14].

5.5 How to choose a secure shuffler?

Our survey presents a wide range of possible shufflers, but there is no absolute best choice. The goal of this section is to guide ML/DP researchers in choosing a secure shuffler for each use case. We first present the guidelines and then apply to three concrete use cases.

Scalability trilemma: shuffle time, message size, and number of messages. In theory, any shuffler can shuffle numerous large messages, but it may take ages. In these guidelines, we simplify this time-size-number trade-off and focus on the number and size of the messages. Implicitly, we consider a secure shuffler suitable if it can shuffle the messages within a few hours. This assumption is in line with the security literature, which rarely provides experimental results of shuffles exceeding 12 hours, such long shuffle times are usually associated with unsuccessful methods.

5.5.1 Guidelines

As guidelines, we propose a list of eight simple questions. These guidelines are more a “rule of thumb” than a formal method, but they should help quickly identify one or several suitable solutions for a use case. The practitioner must refer to Tables 5.10, 5.11, and 5.17 to filter the shufflers based on their answers. This subsection presents the guideline questions one by one, explains their meaning and the kind of expected answer.

Question 1: How many messages are sent during one shuffling round?

Possible answers: Less than 1K, Less than 100K, Less than 1M, More than 1M.

Meaning: Two factors influence the number of messages: the number of data owners, and the private computation task. First, the influence of the number of data owners is straightforward. Second, the type of private computation tasks may produce more or less private messages to shuffle: in gradient aggregation for FL, each message contains a whole gradient, while in histogram-making, each item might be shuffled independently

Rule of thumb Since the asymptotic costs reported in Table 5.10 are hard to relate to a concrete number of messages, we propose a *rule of thumb* to link the asymptotic costs to a number of shuffled messages within a few hours. We propose four thresholds to categorize the techniques based on their scalability with the number of messages. These thresholds are inspired by the observations of the experimental results reported in the surveyed papers.

With less than 1K messages, we can use any technique. *With less than 100K messages* (but more than 1K), we must avoid all techniques with a server cost above $O(n^2)$. For example, PowerMix must be avoided (AsynchroMix being tightly

related to it must also be avoided). *With less than 1M messages* (but more than 100K), we must avoid all techniques with a server cost above $O(n \log n)$. For example, Spectrum is no longer suitable. Note that Blinder is an exception that can leverage GPU computations to scale to millions of messages. *For more than 1M messages*, we recommend using only linear (or sublinear) techniques in n . For example, Trellis scales to dozens of millions using load balancing.

Question 2: How large are the messages?

Possible answers: Less than 1KB, few KBs, few MBs, More than 10 MBs.

Meaning: The main factor influencing the message size is the private computation task because it defines the content of the messages. For example, in gradient aggregation, gradients can produce large messages, while histogram-making requires small messages if we shuffle the items independently (i.e., one item = one message).

Rule of thumb Similarly to the previous guideline question, we propose a rule of thumb to simplify the interpretation of the asymptotic costs. We propose four thresholds to compare the scalability with the message size (refer to Table 5.17 complexity comparison). *With less than 1KB per message*, any technique should be suitable. *With few KBs per message*, we should avoid techniques that are polynomial in the message size (e.g., classic verifiable shufflers). *With few MBs per message*, we should use techniques scaling linearly with the message size and with a sub-quadratic scaling with the number of messages. For example, this rule filters out PowerMix. *For more than a few MBs per message*, one will need to look among the very few techniques compatible with activities such as anonymous whistleblowing requiring large message exchanges (e.g., Riffle).

Questions 3: Are the data owners resource-constrained?

Possible answers: Very constrained, Slightly constrained, or no.

Meaning: If the answer to this question is “very constrained” (e.g., for smartphones), one should avoid all solutions with a sender cost dependent on n . For example, Blinder and Spectrum would be filtered out due to their sender costs. If the answer is “slightly constrained”, one should avoid all solutions with a sender cost with a dependency more than logarithmic on n . For example, Blinder has a square-root dependency and would be filtered out, while Spectrum has a cost logarithmic in n , which is acceptable in this case.

Questions 4: Is there a trusted third party?

Possible answers: Yes or No.

Meaning: In some protocols, a trusted third party (e.g., a public institution) might be needed for setup or public key exchanges. A trusted third party has an

auxiliary role in the protocol, which induces costs (e.g., performing an occasional trusted setup). If there is no trusted third party, one cannot use all the methods with “trusted setup” as an additional assumption in Table 5.11. For example, NIDAR must be avoided.

Questions 5: How many independent/non-colluding servers are available?

Possible answers: Any positive integer.

Meaning: This question is critical to define the threat model (see Table 5.11). For example, if only two servers are available, it is impossible to securely deploy techniques such as Blinder, which requires at least five independent servers to be secure. We call “independent servers”, servers controlled by independent entities (e.g., two different research institutions). While ML researchers can own multiple physical servers, they are not independent (because they represent “colluding” servers sharing their information).

To filter the techniques, we need to keep all techniques whose constraint on M (see Table 5.11) fits your answer. For example, if the answer is 2 (i.e., $M = 2$), one cannot use any MPC-based solutions but can use Spectrum. Moreover, if the involved entities accept it, one can use a technique requiring less than two servers (e.g., TEE); one entity accepting to delegate the shuffling work.

Questions 6: Do you need a correctness guarantee?

Possible answers: Yes or No.

Meaning: Section 5.2 defines the notion of correctness and presents its importance in secure shuffling. It can be mandatory for some applications such as DP, so this question only aims to filter based on the presence of this property (see Table 5.11).

Questions 7: Are the data owners and/or the servers trusted for availability?

Possible answers: Yes or No ($\times 2$).

Meaning: Similarly to the previous question, this question focus on a security property defined in Section 5.2: disruption resistance. If a data owner (resp. server) is untrusted for availability, it can disconnect at anytime (i.e., disrupting the protocol). If there is no trust for availability, only the solutions with sender (resp. server) disruption resistance must be kept.

It is usual in security to trust servers for availability even if they are untrusted for data privacy. As motivated in Section 5.2, sender disruption resistance is particularly mandatory when the number of data owners grows since it becomes hard to assume that all will remain available.

Questions 8: Do you have some application-specific requirements?

Possible answers: None, auditability, GPU availability, etc.

5.5.2 Examples

To better explain how to use these guidelines, we propose three example applications for which we choose a secure shuffler. For each example, we give a brief presentation, the answers to the guideline questions, and the secure shuffler choice. For the sake of simplicity, we detail *only for the first example* how all the unsuitable techniques are filtered out by the guideline questions.

Each use case results into a different choice, and one even has no suitable solution. This observation confirms no universally perfect secure shuffler exists, and gaps remain in the literature.

Medical surveys

A group of hospitals wants to deploy a privacy-preserving surveying system. Each hospital agreed to deploy a shuffling server. The data owners (i.e., survey participants) answer the surveys on their mobile phones and submit their answers through the shuffler. We suppose the practitioner wants to use shuffling-based data [IKOS06] aggregation to preserve privacy.

Answers to the guideline questions: (1) There are at most 10K messages (i.e., $n = 10K$) per shuffle. Indirectly, we assume that it is unlikely to have millions of responses to a single medical survey. (2) A user message is a survey response with less than 1 KB. (3) Very constrained: the device remains online for a short period of time. Concretely, the submission should be completed a few seconds after the user presses the “submit” button. (4) Yes: national health agency. (5) There are 12 hospitals involved (i.e., $M = 12$). (6) We want correctness to ensure the correctness of the medical study. (7) Data owners not trusted for availability. Servers trusted for availability. (8) Specific requirements: auditability, and no hardware investment.

Secure shuffler choice. Mix networks are avoided because they do not guarantee correctness (Question 6). Atom and Riffle are not resistant to sender disruption (Question 7). All DC nets require a sender cost dependent on n (Question 3). AsynchroMix and PowerMix cannot handle so many data owners (Question 1). Blinder requires too much computation and communication on the sender side (Question 3). Clarion has no correctness guarantee (Question 6). Rumor-based shuffle requires too much computation and communication on the sender side (Question 3). NIDAR and secure-aggregation-based shufflers have no correctness guarantee (Question 6).

We end up with classic verifiable shufflers and TEE. We argue **classic verifiable shufflers** are the adequate solution for this use case because they can be deployed

instantly without investing in new hardware. It also satisfies the auditability requirement since the (shuffle) proof transcript can be kept and audited later.

AI-assisted diagnostics

Ten hospitals want to collaborate and use their respective medical data to train a complex ML model to assist the practitioners in their diagnostics. To train these models, the hospitals want to use shuffling-based FL. Each hospital agrees to deploy a shuffling server. Hence, each hospital acts simultaneously as a shuffling server and a data owner.

Answers to the guideline questions: (1) Less than 1K messages per round: each data owner sends one gradient per round. (2) Messages are gradients whose size can reach few MBs. (3) No resource constraint on the data owner side because the hospitals are the data owners. (4) Yes: health control agency. (5) Ten independent servers ($M = 10$). (6) Correctness guarantee wanted. (7) Data owners and servers trusted for availability. (8) Specific requirement: GPU available.

Secure shuffler choice. All MPC-based shufflers except Clarion (i.e., due to its lack of correctness) meet the expectation. However, we argue **Blinder** is the obvious choice because it outperforms AsynchroMix and PowerMix. Moreover, Blinder can take advantage of the available GPUs.

Finally, TEE is again an alternative, but the trust assumption must be accepted, and the hospitals must choose a unique host for the secure enclave.

Smart keyboard application

This last use case was used as motivation in the paper that sparked the interest in FL [KMRR16]: a Google-like company develops a smartphone keyboard application and wants to improve the next-word recommendations. Each smartphone locally trains a model based on the application usage, and a central server securely aggregates the local models to produce a global recommendation model. We want a single-server secure shuffler to perform the secure aggregation step because the company does not want to involve any third party. Google-like services have millions (or even billions) of users, so the scalability expectations are high.

Answers to the guideline questions: (1) Tens of millions of data owners. (2) Messages are gradients whose size can reach a few MBs. (3) Very constrained. (4) No trusted third party. (5) One server available: one company = one independent server. (6) No correctness guarantee needed. (7) Data owners not trusted for availability. Server trusted for availability. (8) Specific requirement: the company does not want to rely on an open-source community such as TOR network due to the economic interest of the application.

Secure shuffler choice. There is **no solution** satisfying all the requirements. The best solutions are TEE-based shufflers, but Prochlo suffers from side-channel attacks, and DBucket has a non-linear complexity (i.e., $O(n \log \ln)$) that could be

acceptable in practice. However, TEE might be unsatisfactory since it requires trusting an independent hardware manufacturer.

5.6 Perspectives on the literature

This section introduces several general perspectives on the literature. The subsections are independent of each other and discuss misconceptions or subtle details that are often overlooked in the literature. Moreover, these perspectives enable to identify several promising future works.

5.6.1 Preventing metadata leakage is unnecessary

This chapter focused on the most conventional definition of sender anonymity, but some works also proposed stronger anonymity notions than Definition 5.5: referred to as ultimate anonymity [GH13] or perfect anonymity [BKM⁺13]. This stronger anonymity forbids the metadata leakage mentioned in Section 5.2: the size and number of sent messages.

From a theoretical point of view, it combines sender anonymity with a notion called “unobservability”. Unobservability ensures that the observation of the traffic leaks absolutely no information. [GH13] show that ultimate anonymity is inefficient because each sender must generate a large amount of cover traffic to prevent metadata leakage. Indeed, each sender must send $O(n)$ data to be a possible sender of the whole batch, even if the sender has small or no data to send. This result explains why Dissent (the only secure shuffler satisfying ultimate anonymity) is the least scalable technique.

While leakage sounds like a side effect to avoid at any cost, this metadata leakage is acceptable in many applications, especially in privacy-preserving computations.

On the one hand, privacy-preserving computations usually involve messages containing fixed-sized vectors or atomic elements (e.g., floating-point numbers). Thus, they are all the same size. On the other hand, many privacy-preserving computations define the number of messages sent by each data owner independently of their private data. For example, in shuffling-based secure aggregation [IKOS06], each data owner must send the same number of messages. To sum up, (shuffling-based) privacy-preserving computations do not require anonymity stronger than the sender anonymity defined in Section 5.2 because the message metadata is often fixed and known publicly.

5.6.2 Imperfect shuffling/anonymity can be acceptable

Shuffling-based private computations (including in DP) usually assume a perfect shuffle (i.e., uniform anonymity distribution as defined in Section 5.2.2). However, [IKOS06] claimed that only a “sufficient uncertainty” was necessary; i.e., imperfect

shuffle could be acceptable. Recently, [BBG23] showed that this intuition was true as their results holding also for an imperfect shuffle.

These results should incentivize DP researchers to build DP algorithms robust to “slightly imperfect” shuffling and security researcher to build protocols leveraging this imperfection. Nevertheless, further theoretical work is also necessary to standardize this notion of imperfect shuffling.

5.6.3 Avoid modifying the shuffler functionality

Even though shuffle DP works agree on the intuitive definition of a shuffler, there is no consensus regarding the exact capability of a shuffler. Most papers [CSU⁺19, Che21] rely on the most basic definition as we do (i.e., the shuffler simply outputs a permutation), but some papers [YDL⁺23] complete this functionality with other simple tasks such as adding unique identifiers to the messages.

We argue that such modifications should be avoided because these modified functionalities cannot be implemented naively with all existing secure shufflers. Thus, DP researchers should stick to the most basic and generic shuffler functionality to guarantee that the shuffling functionality can be implemented securely. We recommend DP researchers to adopt a “generic-shuffler” approach, assuming a generic behavior as defined in Definition 5.1. Such an approach would guarantee that their DP algorithm can be implemented using any of the shufflers listed in our survey. While most works already adopt this approach, our findings highlight its importance.

One may find our recommendation contradictory with the acceptability of an imperfect shuffle, but the imperfect shuffle does not modify the functionality (i.e., the shuffler still outputs a permutation), it simply modifies the security definition.

Finally, this generic approach should also be completed with some security discussions. Indeed, shuffle DP works assume a perfect shuffler and do not discuss the consequences of a flawed shuffler. For example, what are the consequences if the shuffle output was partially censored (i.e., no correctness)? Such security discussions are absent from existing DP works, but would provide a much better understanding of the privacy limitations. Using the security definitions proposed in Section 5.2, DP researchers would have a standardized set of properties to discuss.

5.6.4 Information-theoretic Shuffle DP is impractical

Computational DP [BN08, MPRV09] is a relaxed variant of DP where the DP guarantees hold against computationally-bounded adversaries. This definition is opposed to “information-theoretic” DP (i.e., the initial DP definition proposed by Dwork [Dwo06]), where the guarantees also hold against computationally unbounded adversaries. This separation is classic in security, where we oppose computational security to information-theoretic security.

A DP algorithm is “computationally differentially private” if one of its components is only computationally secure. For example, as mentioned in [BCV16], information-theoretic DP does not permit using cryptographically secure pseudo-random generators in place of perfect randomness.

This constraint (i.e., no computationally secure component) is problematic for Shuffle DP. As mentioned in Section 5.3, all secure shufflers assume the existence of a secure communication channel between the senders and the shuffler. However, Maurer [Mau99] showed that information-theoretically secure communication channels are impractical. Therefore, we should only assume computationally secure communication channels in the context of shuffle DP. Hence, **only computational shuffle DP can be implemented efficiently**.

Relying on “information-theoretic” shuffle DP is then impractical because any real world deployment requires a computational hardness assumption. Thus, shuffle DP applications should take this mandatory assumption into consideration in their analysis. Such consideration should incentivize more “computational” DP results; known to require less noise than information-theoretic DP [MPRV09].

This conclusion holds for all DP protocols assuming the existence of a secure communication channel between two parties.

5.6.5 Malicious data owners are a serious threat

Our security properties hold against an unbounded number of malicious data owners. Anonymous communications systems commonly assume a “sufficiently high” number of honest senders, as anonymity holds within the set of honest senders. For example, [LSD23] highlighted this limitation when presenting their anonymous broadcaster.

This vague assumption is acceptable because the number of malicious data owners has no influence on the security properties of a secure shuffler. However, this number has an influence on the privacy and correctness of the overall shuffling-based private computation.

Threat to privacy First, malicious data owners can decrease the privacy guarantees in Shuffle DP computations because the DP noise is proportional to the number of honest data owners. Indeed, if a malicious data owners does not generate a noise following the agreed protocol, the output result will provide a weaker privacy than expected. Hence, knowing at least a lower bound on the number of honest data owners is necessary.

To verify that malicious data owners add an appropriate noise, [BC22] introduced the notion of verifiable DP (generalized to Shuffle DP by [BAT24]); leveraging zero-knowledge proofs to verify the DP noise generation and addition.

Threat to correctness Malicious data owners also threaten the computation correctness because they can bias the final results via data poisoning attacks. For example, in FL, this takes the form of model poisoning attacks [CJG21, Car21, SSTS21] in which attackers submitting local models that would introduce undesired behaviors in the global model. Such poisoning can have serious consequences as it can, for example, add a backdoor to an ML model [SSTS21].

Norm bounding is a popular mitigation technique [SHKR22]. This simple mitigation has been incorporated into secure aggregation protocols using ZKPs [BGL⁺22, RGJvdM22].

However, no mitigation compatible with shuffling-based protocols has been yet described. This verification is trivial when each data owner sends a single message because the verification can be done on the shuffle output. However, in multi-message Shuffle DP [Che21] or shuffling-based secure aggregation [IKOS06], each data owner splits its private value into several messages. A zero-knowledge verification is necessary to verify claims about the initial private value while protecting anonymity.

5.6.6 Trusted hardware is not a silver bullet

In theory, trusted hardware (i.e., TEE) is an appealing assumption that can solve privacy issues: the protocol is secure as long as the data goes through the trusted hardware (even with a malicious server owner). Several papers promoted TEE in FL [BEM⁺17, BLAB21, ZWC⁺21, ZJF⁺22] motivated by its performances and (theoretical) security. However, one must also understand the strength of the assumption required by TEEs: the data owner fully trusts the manufacturer. TEE also requires trusting the protocol implementation because there exist other flaws: side-channel attacks [BMD⁺17, WCP⁺17].

[SJG22] presented an excellent example of the risk of TEE. While some shuffle DP papers used Prochlo [BEM⁺17] (a TEE-based secure shuffler) as motivation, [SJG22] argued that recent attacks against TEE make Prochlo insecure. Hence, even if Prochlo seemed at first totally secure, it turned out to be insecure in practice due to TEE vulnerabilities.

Finally, as TEE relies on particular hardware, one requires compatible hardware. It is a solution that cannot be deployed anywhere easily; it requires a dedicated hardware investment.

Conclusion and future directions

This survey highlighted the interest and properties of secure shufflers. These secure protocols can support privacy-preserving computations involving thousands or even millions of data owners. However, guaranteeing the security properties

(conditioning the privacy guarantees) requires some attention. We identified and compared six categories of secure shufflers. We provided guidelines and examples to help an ML researcher choose the adequate secure shuffler depending on the use case. Hence, secure shufflers are a promising line of work for privacy-preserving computations, but some gaps remain in the literature:

Optimization of anonymous broadcasters for the “few-receivers/many-senders” setup. Many anonymization systems, motivated by applications such as anonymous whistleblowing, are optimized for setups with few senders and many receiver. However, secure shuffling usually involved an opposite setup many senders and very few receivers (i.e., the analyzer). Thus, leveraging the properties of this setup may improve multi-server secure shuffling.

Linear complexity for 1-server secure shuffling. Recent works [BEM⁺17] [SW21, BHMS21] approached linear complexity (i.e., $O(n)$ communications and computations) for 1-server secure shuffling, the optimal complexity for 1-server shuffling.

Prochlo has linear complexity but is vulnerable to side-channel attacks. Recent secure TEE-based solutions [NDCP24, GWC⁺23] avoided these attacks, but have a complexity of $O(n \log n)$. Finally, NIDAR verifies fewer security properties than the rest of the solutions and only has subquadratic costs.

Hence, research is still necessary to achieve a linear complexity with satisfying security properties (or to provide an impossibility result for this complexity).

Robustness to data poisoning Shuffling-based privacy-preserving computations are particularly sensitive to data poisoning attacks (see Section 5.6.5). The anonymization provided makes it hard (or even impossible) to detect “poisonous” inputs from the shuffle output. Attackers can easily provide maliciously crafted input biasing the result (e.g., adding a backdoor to a model in FL).

The literature describes no mitigation compatible with shuffling-based protocols. Thus, research is needed to build new mitigations specifically for shuffling-based privacy-preserving computations.

Formalization of the “imperfect” shuffle DP model. Several recent works [BHMS21, LSD23, ZS22, BBG23] proposed relaxed anonymity definitions to conceive more scalable secure shufflers. While some works [ZS22, BBG23] adapted the shuffle DP model to their “imperfect” shuffling, no DP results exist for the relaxation of [LSD23]. Thus, further theoretical work is still needed to formalize these relaxations, and present DP results for shufflers with a relaxed anonymity definition.

Computational shuffle DP. Section 5.6.4 explained why only computational Shuffle DP is practical. While [MPRV09] showed that computational DP can reach better accuracy guarantees than information-theoretic DP, no work has yet studied this relaxation in Shuffle DP. Hence, analogous investigations for Shuffle

DP would be necessary to take advantage of our impossibility result and reach higher accuracy guarantees.

New types of shuffling-based private computation. While most existing applications are using secure shufflers to perform a form of data aggregation, recent works [GIK⁺24, WDS⁺25] recently opened the way to other privacy-sensitive applications such as information retrieval or taxi hailing could be enabled by secure shufflers. Thus, further research into new shuffling-based private computations is a promising direction. The concept of “private individual computation” introduced by [WDS⁺25] represents a valuable framework to structure such research effort.

Part II

Risks of Access Pattern Leakage

This second part studies the risk induced by access pattern leakage. As explained in Chapter 4, state-of-the-art works in sparse computations have systematically avoided this leakage, but have not provided evidence of its risks. As no existing secure sparse computation relies on this leakage, we cannot prove this risk directly for sparse computations.

Instead, we study access pattern leakage in the related setting of searchable encryption, where it is present in many schemes. We propose a new attack exploiting this leakage with high accuracy, and we revisit existing attacks using statistical methods to better assess their practical impact. These results demonstrate the risks of leakage in practice and indirectly highlight the dangers of similar leakage in sparse computations, motivating the need for MPC protocols that minimize leakage.

All chapters in this part rely on the notations presented in Table 6.1.

Leakage-Abuse Attack against Searchable Encryption using Non-Indexed Documents

Existing attacks against SSE [IKK12, CGPR15] only achieve decent results with strong attacker's models (e.g. at least 20% of previously known documents or require additional knowledge such as on query frequencies) and they give no metric to evaluate the certainty of recovered queries. This hampers their practical utility and questions their relevance in the real-world.

We propose the *refined score attack* which achieves query recovery rates of around 85% without requiring exact background knowledge on stored documents; a distributionally similar, but otherwise different (i.e., non-indexed), dataset suffices. The attack starts with a few known queries (around 10 known queries in our experiments) and then iteratively recovers queries based on confidence scores.

This chapter is drawn from [DHP21].

6.1 Introduction

The access pattern leakage of SSE schemes has been exploited in several passive attacks. [BKM20] divided these attacks into two categories: (1) known-data attacks where the attacker has partial or complete knowledge of the documents indexed by the server, and (2) inference attacks where the attacker only knows some (non-indexed) documents similar to the indexed documents. An attacker who can run known-data attacks is more powerful than an attacker who is restricted to inference attacks. While [IKK12] and [CGPR15] motivated their attacks as inference attacks, decent results were only achieved in the setting of known-data attacks. Recently, [BKM20] proposed an attack that conceptually only work as known-data attacks and do not support inference attacks at all.

In conclusion, no practical inference attack has been proposed so far that achieves an accuracy higher than 50% even under advantageous conditions (e.g., the client's keyword universe is small AND known by the attacker)¹. Also, we discuss orthogonal lines of research focusing on other attacker models or schemes with different leakage profiles.

Other types of attack exist, but are out of scope: active attacks [LCL24, ZSCL24] [ZWX⁺23, ZKP16], attacks exploiting the query frequency [LZWT14, OK21], and attacks against other schemes [FMA⁺20, GLMP18, KKN016, KPT19, KPT20, LMP18].

¹This chapter was written in 2021, and several new attacks followed the publication of this contribution. Chapter 7 provides an up-to-date state-of-the-art.

Notation	Meaning	Size
Base attacker's knowledge		
D_{atk}	Similar document set	n_{atk}
$\mathcal{Q}$	Observed queries (i.e., a list of trapdoors)	l
$R_{\mathcal{Q}}$	Query results (i.e, document IDs for each $td \in \mathcal{Q}$)	l
Known $\mathcal{Q}$	(trapdoor, keyword) pairs known by the attacker	k
Derived attacker's knowledge		
$\mathcal{W}_{\text{atk}}$	Vocabulary extracted from D_{atk}	m_{atk}
C^{atk}	Word co-occurrence matrix built from D_{atk}	$m_{\text{atk}} \times m_{\text{atk}}$
C^{query}	Trapdoor co-occurrence matrix built from $R_{\mathcal{Q}}$	$l \times l$
Unknown by the attacker		
D_{ind}	Encrypted documents indexed by the server	n_{ind}
$\mathcal{W}_{\text{ind}}$	Queryable vocabulary (=keyword universe)	m_{ind}

Table 6.1 / Notations for the chapters on SSE attacks

This chapter focuses on (passive) inference attacks against static SSE schemes with access and search pattern leakage.

Our contribution This chapter describes an attack that works *without* knowledge of the indexed documents and only uses similar data. At the same time, our inference attack achieves recovery rates of up to 90%. The documents known by the attacker only need to be distributionally close to the indexed documents. For example, an attacker can mount a successful attack exploiting information from a previous data breach, even if the breached documents have been identified as such and were purged from the index to mitigate future known-data attacks.

Our attack is based on a confidence metric used to score trapdoor-keyword pairs. The score should be maximized when the trapdoor (i.e., a query token) is paired with its (correct) underlying keyword. We start with our *score attack* that computes a confidence score of each trapdoor-keyword pair and returns, for each trapdoor, the pair with the highest score. This base inference attack reaches a recovery rate of 60% while assuming around 25% of known queries.

We then propose an improvement strategy reducing drastically the amount of attacker's knowledge necessary, especially regarding the known queries. Our refined score attack, an iterative refinement strategy, reaches recovery rates of up to 85% with only 10 known queries in our experiments over different datasets of varying size. More specifically, the iterative scoring approach recovers further queries by adding previously recovered queries that had a high confidence score to the set of known queries. Our attack has a low runtime and can be performed in less than two minutes. Finally, we show that both padding and obfuscation countermeasures can successfully mitigate our attack. However, these countermeasures induce storage and communication overheads.

6.2 Threat model

The notation as introduced in this section and used throughout Part II is summarized in Table 6.1.

6.2.1 Attacker's behaviour

A passive attacker observes the trapdoors sent to the server and the server response, which includes the list of the matching document identifiers. These identifiers reveal no further information about the content of the document. The attacker can link a query to its response and create (Trapdoor, DocIDs) pairs. We consider two slightly different attacker models; both are applicable for our attack as described in Section 6.5:

- An *honest-but-curious* server follows the protocols but tries to recover the underlying keywords of the queries. To facilitate search on encrypted documents, the encrypted index is usually supposed to be stored on the server along with the encrypted documents as is the case in settings considered by the [IKK12] and [CGPR15] attacks. Such an attacker owns metadata about the encrypted documents: the total number of documents and their size.
- A *passive traffic observer* records the traffic of the database. This attacker only has pairs of (Trapdoor, DocIDs) and uses them to recover the underlying keywords. It could also represent a case where the index server does not store the encrypted documents. Such an index server ignores the number and the size of the indexed encrypted documents.

6.2.2 Attacker's knowledge

Similar document set The attacker's knowledge is focused on a similar document set $D_{\text{atk}} = \{d_1, \dots, d_{n_{\text{atk}}}\}$. A document set D_{atk} is similar if it is distributionally close to the indexed documents D_{ind} . Section 6.2.3 proposes a formal definition of document set similarity.

In a company, an employee's mailbox is a document set similar to her colleague's mailboxes. As another example, leaked confidential notes are a similar document set to recover the queries on the rest of the notes. Known-data attacks can also work on leaked documents, but the server can simply remove these documents from the index to avoid these attacks. Despite this removal, our "refined" attack is still effective. Since it is an inference attack, we do not need our documents to be indexed as opposed to the known-data attacks [IKK12, CGPR15, BKM20] that assume their known documents are part of the document set indexed by the server.

A vocabulary $\mathcal{W}_{\text{atk}}$ of m_{atk} keywords is extracted from D_{atk} . An index matrix is built from this document set: $\text{ID}_{\text{atk}}[i, j] = 1$ if the j -th keyword is contained in the i -th document, 0 otherwise. Then, the $m_{\text{atk}} \times m_{\text{atk}}$ co-occurrence matrix is

$C^{\text{atk}} = \text{ID}_{\text{atk}}^\top \text{ID}_{\text{atk}} \cdot \frac{1}{n_{\text{atk}}}$, where $n_{\text{atk}} = |D_{\text{atk}}|$.² Note that we use relative frequency numbers rather than absolute count numbers.

Keyword extraction algorithm The (keyword) distributional knowledge on document sets depends on how keywords are extracted from the documents. The attacker uses an algorithm to extract the vocabulary from her known document set. In the literature, all attack papers (im- or explicitly) assume that the attacker uses the same keyword extraction algorithm as the client. Whether this assumption is realistic or not has not been questioned in the literature, but we would like to stress the importance of this assumption here. We also assume the attacker and the client use the same extraction algorithm, and we leave the study of the case of different extraction algorithms for future work.

Observed queries We denote as $\mathcal{W}_{\text{ind}}$ the client's query-able vocabulary of m_{ind} keywords. The attacker does know neither this vocabulary nor its size. The attacker observes l unique trapdoors and obtains their corresponding search results. Let $\mathcal{Q} = \langle td_1, \dots, td_l \rangle$ be the set of observed trapdoors and $R^{\text{query}} = \{id(d) | (x \in \mathcal{W}_{\text{ind}}) \wedge (td = \text{Trapdoor}(x)) \wedge (d \in D_{\text{ind}}) \wedge (x \in d)\}$, the document identifiers returned for the trapdoor td .

Let $\text{DocIDs} = \{id_1, \dots, id_p\} = \bigcup_{td \in \mathcal{Q}} R^{\text{query}}$. We note that $p \leq n_{\text{ind}}$. Let ID_{ind} be the $p \times l$ index matrix built from the responses to the trapdoors as follows: $\text{ID}_{\text{ind}}[i, j] = 1$ if the response to the j -th trapdoor contains the i -th identifier, 0 otherwise. Finally, we can infer the $m_{\text{ind}} \times m_{\text{ind}}$ trapdoor co-occurrence matrix: $C^{\text{query}} = \text{ID}_{\text{ind}}^\top \text{ID}_{\text{ind}} \cdot \frac{1}{n_{\text{ind}}}$.

Known queries As in [IKK12] and [CGPR15], the attacker's knows the underlying keywords of k queries in $\mathcal{Q}$. The set of known queries is defined as follows:

$$\begin{aligned} \text{Known}\mathcal{Q} = \{ \langle kw_{\text{known}}, td_{\text{known}} \rangle & | (kw_{\text{known}} \in \mathcal{W}_{\text{ind}} \cap \mathcal{W}_{\text{atk}}) \\ & \wedge (td_{\text{known}} \in \mathcal{Q}) \wedge (td_{\text{known}} = \text{Trapdoor}(kw_{\text{known}})) \} \end{aligned}$$

6.2.3 Similarity definitions

Similar document set Let $\mathcal{C}(D, kw_a, kw_b)$ be the function returning the number of co-occurrences of keywords kw_a and kw_b inside the document set D . Let $\text{SimMat}(D_1, D_2, \mathcal{W})$ be a function returning an $m \times m$ similarity matrix of D_1 and D_2 over the vocabulary $\mathcal{W} = \{kw_1, \dots, kw_i, \dots, kw_m\}$. The function SimMat is defined such that:

$$(\text{SimMat}(D_1, D_2, \mathcal{W}))_{ij} = \frac{\mathcal{C}(D_1, kw_i, kw_j)}{|D_1|} - \frac{\mathcal{C}(D_2, kw_i, kw_j)}{|D_2|} \quad (6.2)$$

² $A^\top$ denotes the transpose of a matrix A

In other words, the ij -th element of the matrix returned by SimMat is the difference between the co-frequency of the keywords i and j in the document set D_1 and the co-frequency of the same two keywords in the document set D_2 . Thus, $\text{SimMat}(D_1, D_2, \mathcal{W})$ describes the similarity of D_1 and D_2 over the vocabulary $\mathcal{W}$ and the norm $\|\text{SimMat}(D_1, D_2, \mathcal{W})\|$ is a measure of the similarity of D_1 and D_2 . We only consider the Frobenius norm, but other norms can be considered as well.

We define D_{atk} and D_{ind} as two ε -similar document sets if for $\varepsilon \geq 0$ the following holds:

$$\|\text{SimMat}(D_{\text{atk}}, D_{\text{ind}}, \mathcal{W}_{\text{ind}})\| \leq \varepsilon \quad (6.3)$$

In other words, the closer the co-frequencies between the document sets are, the more similar the document sets are. In our definition, we only need to consider the similarity over the queryable vocabulary $\mathcal{W}_{\text{ind}}$ because those are the only keywords that are queried for by the client and to be recovered by the attacker.

Similar and queryable vocabularies To recover the queries, the attacker needs to have as many elements of the queryable vocabulary $\mathcal{W}_{\text{ind}}$ as possible in her similar vocabulary $\mathcal{W}_{\text{atk}}$. This creates a natural upper bound for the attack accuracy:

$$\text{AttackAccuracy} \leq \frac{|\mathcal{W}_{\text{ind}} \cap \mathcal{W}_{\text{atk}}|}{|\mathcal{W}_{\text{ind}}|} \quad (6.4)$$

In other words, the attacker can only recover the queries for which the underlying keywords are contained in $\mathcal{W}_{\text{atk}}$.

In the experiments presented in Section 6.5, $\mathcal{W}_{\text{atk}}$ contains most elements of $\mathcal{W}_{\text{ind}}$ since the average accuracy goes up to 95% which means that more than 95% of the keywords of the queryable vocabulary $\mathcal{W}_{\text{ind}}$ are contained in the similar vocabulary $\mathcal{W}_{\text{atk}}$.

Attacker assumptions An attacker's knows neither the indexed documents D_{real} nor the vocabulary $\mathcal{W}_{\text{ind}}$. Thus, she cannot calculate the exact ε -similarity between her dataset D_{sim} and the indexed dataset D_{real} . We assume that:

- (1) D_{sim} is ε -similar to the indexed document set D_{real} , for a sufficiently small ε (e.g. $\varepsilon = 0.8$ as in the Fig. 6.11).
- (2) $\mathcal{W}_{\text{sim}}$ contains most elements of $\mathcal{W}_{\text{real}}$ (especially $\mathcal{W}_{\text{real}}(\mathcal{Q})$, the underlying keywords of the queries observed by the attacker).

6.3 Score attack

On an intuitive level, our *score attack* uses a confidence metric that scores trapdoor-keyword pairs. This metric is called a *matching score* and should be maximized

when the trapdoor is paired with its correct underlying keyword. The attacker computes the matching score of every possible trapdoor-keyword pair. For each trapdoor, the trapdoor-keyword pair with the highest score is returned.

6.3.1 Extracting the known query co-occurrence sub-matrices

The attacker uses her known queries (= known correct trapdoor-keyword pairs) to project the keyword and the trapdoor co-occurrence matrices to a common sub-vector space. This projection is done by only keeping the columns of the known queries and sorting the columns using the known queries such that the i -th column is related to the i -th known query. Formally the projection works as follows:

For a keyword kw , we denote its position in $W_{sim} = (kw_1, \dots, kw_{m_{sim}})$ by $pos(kw)$, i.e., $pos(kw_i) = i$ for $kw_i \in \mathcal{W}_{sim}$. Likewise, we denote the position of a trapdoor td in the list of observed queries $Q = \langle td_1, \dots, td_l \rangle$ by $pos(td)$. We define the *projection of the trapdoor-trapdoor co-occurrence matrix C^{query} onto the known queries* as the $l \times k$ matrix C_{sub}^{query} such that: for all $i \in \{1 \dots l\}$ and all $j \in \{1 \dots k\}$ there exists a known query $q_j = (td_{known}, kw_{known}) \in \text{Known}\mathcal{Q}$ such that

$$C_{sub}^{query}[i, j] = C^{query}[i, pos(td_{known})]. \quad (6.5)$$

Likewise, we define the *projection of the word-word co-occurrence matrix C^{atk} onto the known queries* as the $m_{sim} \times k$ matrix C_{sub}^{atk} such that: for all $i \in \{1 \dots m_{atk}\}$ and all $j \in \{1 \dots k\}$ there exists a known query $q_j = (td_{known}, kw_{known}) \in \text{Known}\mathcal{Q}$ such that

$$C_{sub}^{atk}[i, j] = C^{atk}[i, pos(kw_{known})]. \quad (6.6)$$

In our notation, we use the subscript *sub* to emphasize that C_{sub}^{query} and C_{sub}^{atk} define co-occurrence *sub*-matrices. Such matrices are very convenient since we can directly compare a keyword and a trapdoor. We denote as $C_{sub}^{atk}[kw_i]$ (resp. $C_{sub}^{query}[td_j]$), the vector composed of the co-occurrences of keyword kw_i (resp. trapdoor td_j) with every keyword (resp. trapdoor) related to a known query. Thus, we can extract a k -dimensional vector describing each keyword or trapdoor. In the next section, we will define our confidence score based on the distance between a keyword vector and a trapdoor vector.

6.3.2 Confidence score and matching process

The sub-matrices C_{sub}^{atk} and C_{sub}^{query} are used to score a trapdoor-keyword pair. A score should be maximized when the pair is correct. The *scoring function* for a vector-norm $\|\cdot\|$ (e.g. the L2 norm) is defined as

$$\text{Score}(td_j, kw_i) = -\ln(\|C_{sub}^{atk}[kw_i] - C_{sub}^{query}[td_j]\|), \quad (6.7)$$

for all $kw_i \in \mathcal{W}_{atk}$ and all $td_j \in \mathcal{Q}$.

Algorithm 3 Score attack**Require:** $\mathcal{W}_{\text{atk}}, C_{\text{sub}}^{\text{atk}}, \mathcal{Q}, C_{\text{sub}}^{\text{query}}$.

```

1:  $\text{pred} \leftarrow []$ .
2: for  $td \in \mathcal{Q}$  do
3:    $\text{dists} = []$ .
4:   for  $kw \in \mathcal{W}_{\text{atk}}$  do
5:      $s = -\ln(\|C_{\text{sub}}^{\text{atk}}[kw] - C_{\text{sub}}^{\text{query}}[td]\|)$ .
6:     Append  $(kw, s)$  to  $\text{dists}$ .
7:   Sort  $\text{dists}$  in the descending order.
8:   Append  $(td, \text{dists}[0])$  to  $\text{pred}$ .
9: return  $\text{pred}$ 

```

Note that Equation (6.7) results in a high score when the distance between a keyword and a trapdoor is small. This distance can be obtained since $C_{\text{sub}}^{\text{atk}}[kw_i]$ and $C_{\text{sub}}^{\text{query}}[td_j]$ share a common vector space. The $-\ln$ function is used to transform the distance into a score to focus on the order of magnitude instead of distance values close to zero. For example, a distance of e^{-11} results in a score of 11. In our case, the distance is always less than 1 because we are using relative frequency matrices. We focus on orders of magnitude for interpretability matters. Even for an attack accuracy above 80%, the attacker needs to identify the correct predictions. The interpretability provided by a scoring approach is then necessary. We argue that there is a higher interpretative meaning when comparing two orders of magnitude scaled between 0 and ~ 30 (experimental upper bound) than comparing two small norms close to zero.

Our score attack uses the score function as follows: for each trapdoor, it goes through all keywords and returns the keyword for which the score is maximized. See Algorithm 3 for an algorithmic description. Note that the score is a confidence score, and the attacker can sort the predictions (i.e., the trapdoor-keyword pairs returned) based on their matching score to define the most likely predictions.

The norm $\|\cdot\|$ can be chosen freely. However, our experiments showed that the L2 norm maximizes the accuracy. This norm tolerates a high difference between one of the k components of the vector, i.e., when one of the co-occurrences in the attacker dataset is very far from its corresponding co-occurrence in the indexed dataset.

6.4 Experimental results

6.4.1 Methodology

Datasets We simulate the attacks using two publicly available datasets. First, we use the Enron dataset, which is widely used in the literature to simulate attacks. Like in [IKK12] and [CGPR15], we compose our Enron document set

by extracting every email contained in the folders `__sent_mail` to obtain 30109 documents. Second, we use data from the Apache mailing list archives. We use specifically the “java-user” mailing list from the Lucene project for the years 2002-2011. This second dataset contains 50878 documents. It was introduced in [CGPR15].

Keyword extraction The keyword extraction is exclusively done on the email content. Thus, keywords of the title or the recipients’ names cannot be queried. To obtain the list of the keywords, we stem the words using the Porter Stemmer and remove the stop words. For the Apache dataset, we systematically remove the mailing list signature proposing to unsubscribe. Otherwise, the keyword contained in this “Unsubscribe” message would be useless in the search since it appears in every email.

Attacker’s knowledge generation At the beginning of an experiment, the document set used (i.e., Enron or Apache) is divided randomly into two disjoint subsets. One subset is used to generate the index, i.e., the encrypted document set D_{ind} . The second subset is part of the attacker’s knowledge, i.e., the similar document set D_{atk} . Every experiment is done with non-overlapping document sets, that is, only as inference attack. The similar vocabulary $\mathcal{W}_{\text{atk}}$ is extracted from D_{atk} and is given to the attacker. The index vocabulary $\mathcal{W}_{\text{ind}}$ is extracted from D_{ind} and is not known by the attacker. The similar (resp. real) vocabulary extraction algorithm consists in extracting the m_{sim} (resp. m_{real}) most frequent keywords of D_{atk} (resp. D_{ind}). The underlying keywords of the queries are chosen uniformly at random from $\mathcal{W}_{\text{ind}}$. For each run, document and query sets are freshly chosen uniformly at random.

Hardware and software The experiments are done on a Debian server with a quad-core processor (64 bits, 2.1 GHz) and 8 GB of memory. The algorithms are implemented using Python 3.7. Specifically, we use the NLTK for the basic natural language processing: word tokenization, stemming, and stopwords.

Our code to simulate the attack and to obtain our results is publicly available:

<https://github.com/MarcTOK/Refined-score-atk-SSE>.

Result presentation We call *correct prediction*, a query for which the algorithm has returned the corresponding underlying keyword. We evaluate the performance of our attack using the term *accuracy*. The accuracy corresponds to the number of correct predictions divided by the number of unknown queries. Our accuracy excludes known queries and is always computed over the unknown queries (i.e., $\text{acc} = \frac{|\text{CorrectPred}(\text{UnknownQ})|}{|\text{Q}| - |\text{KnownQ}|}$). In other works such as [CGPR15], the term *recovery rate* is also used to define this concept. We use bar plots to present the results of our experiments. Each bar is obtained by computing the average result over 50 attack simulations. These bars are completed with error bars corresponding to $\mu \pm \sigma$ with μ the average accuracy and σ the standard deviation.

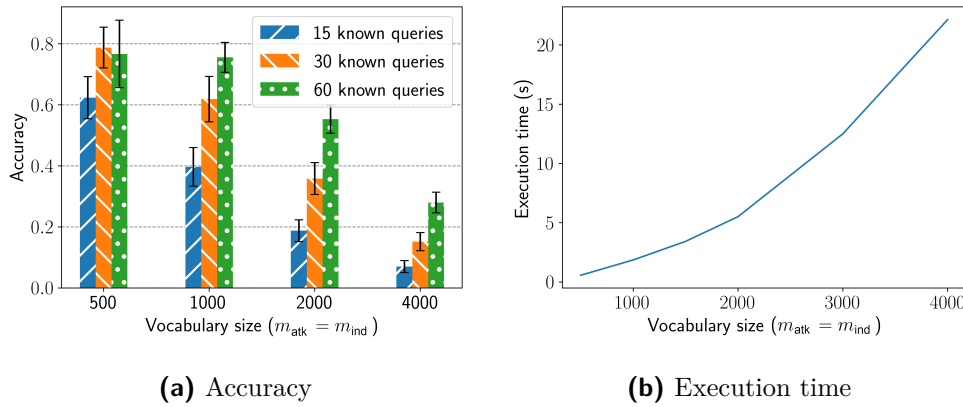


Figure 6.8 / Average accuracy and execution time of the matching algorithm for varying vocabulary sizes ($|D_{\text{atk}}| = 12K$, $|D_{\text{ind}}| = 18K$, $|\mathcal{Q}| = 0.15 \cdot m_{\text{ind}}$, $|\text{Known}\mathcal{Q}| = 10$)

6.4.2 Results

Fig. 6.8a shows the accuracy of the algorithm on Enron corpus for several vocabulary sizes. The server stores 60% of the corpus and the attacker knows the remaining 40%. The attacker has observed 15% of the possible queries. She knows either 15, 30, or 60 queries. When the vocabulary size is 1000 and the attacker knows 30 queries (20% of the queries observed), the average accuracy is 60%. When the vocabulary size is 2000 and the attacker knows 60 queries (20% of the queries observed), the average accuracy is 55%. Then, the base algorithm can be successful only if it has enough known queries. When the vocabulary is larger, the accuracy decreases if the number of known queries remains the same. The accuracy increases in function of the number of known queries (Fig. 6.8a). We argue that we could obtain better results for large vocabularies with more known queries. Obtaining so many queries is unrealistic since we would need a preliminary inference attack or a massive injection attack to obtain the knowledge required by this base inference attack. Therefore, the base score attack is only a practical attack when the size of the server vocabulary is below 2000.

In Fig. 6.8a, we can distinguish one surprising result when there are 60 known queries, and the vocabulary size is 500. This result is decreased compared to the previous one (i.e., 30 known queries), and the error bars are overlapping. We can explain that because, in this experiment, there are 75 queries for 60 known queries, i.e., only 15 unknown queries. It is the only experiment where there is a minority of unknown queries. We consider this result insignificant but keep it in this chapter for the complete discussion. Indeed, it does not make sense to know most of the queries before the attack and to consider the result obtained on recovering such a small amount of unknown queries.

In [CGPR15], it was assumed that the co-occurrences were too noisy to rely totally on them. The occurrence is much less noisy, but most of the keywords have an extremely close frequency. Therefore, it is impossible to identify a keyword just based on a single occurrence estimator except if we are sure to have perfect

knowledge as in [CGPR15] attack when they know nearly 100% of the encrypted documents. The co-occurrence is noisier, but its distribution is scattered enough to identify keyword-trapdoor pairs. The lack of precision is balanced by the number of co-occurrences available to perform the identification. There is a trade-off between the number of estimators and the precision.

6.4.3 Execution time

The complexity in time of the algorithm in Algorithm 3 is $\mathcal{O}(|\mathcal{Q}| \cdot m_{\text{atk}} \cdot k)$, if we consider the complexity of the norm $\|\cdot\|$ to be $\mathcal{O}(k)$. Fig. 6.8b describes the average execution time of this algorithm over 50 repetitions in function of the vocabulary size. We exclude the keyword extraction and the co-occurrence computation from this execution time. This experiment was done with Enron dataset. The similar document set represents 40% of the total dataset (12 044 emails), and the server document set 60%. Even with large document sets, the execution time is negligible (20 seconds) compared to attacks like [PW16], which needs 16 hours when $m_{\text{atk}} = m_{\text{ind}} = 1K$. Our implementation is already CPU-parallelized but can be further improved using GPU parallelization.

Besides its short runtime, this algorithm is deterministic and parameter-less. Non-determinism is present, for example, in the simulated annealing used by [IKK12]. Indeed, two runs of [IKK12] algorithm could result in two different results because of a random choice present in this algorithm. It becomes a problem when the attack is too long to be repeated many times with different initializations and/or when the attacker does not have a confidence metric to identify the correct predictions (as in [IKK12]). The [CGPR15] attack introduced an error-rate parameter, which needs to be set experimentally. However, it is unclear whether this parameter is specific to each document set or not and how to set it properly.

6.5 Refined score attack

6.5.1 Algorithm

Our base attack introduces a matching score that acts as a confidence metric: the higher the score is, the more likely the correctness of the keyword-trapdoor pair is. We can use this property to determine the most certain predictions: a keyword-trapdoor candidate (kw_i, td) will be considered as *certain* if its score is much higher than the scores of any other candidate (kw_j, td) . The *certainty of a prediction* kw_i for the trapdoor td is defined as:

$$\text{Certainty}(td, kw_i) = \text{Score}(td, kw_i) - \max_{j \neq i} \text{Score}(td, kw_j) \quad (6.9)$$

Based on this certainty, we propose a refinement process, which drastically reduces the number of known queries needed on the attacker's side: the matching is performed several times, and at the end of each round, the most certain predictions

Algorithm 4 Refined score attack

Require: $\mathcal{W}_{\text{atk}}, C_{\text{sub}}^{\text{atk}}, \mathcal{Q}, C_{\text{sub}}^{\text{query}}, \text{Known}\mathcal{Q}, \text{RefSpeed}$

```

1: final_pred  $\leftarrow \emptyset$ 
2: unknownQ  $\leftarrow \mathcal{Q}$ .
3: while unknownQ  $\neq \emptyset$  do
    // 1. Extract the remaining unknown queries
4:   unknownQ  $\leftarrow \{td : (td \in \mathcal{Q}) \wedge (\nexists kw \in \mathcal{W}_{\text{atk}} : (td, kw) \in \text{Known}\mathcal{Q})\}$ .
5:   temp_pred  $\leftarrow \emptyset$ .

    // 2. Propose a prediction for each unknown query
6:   for  $td \in \text{unknownQ}$  do
7:     dists  $\leftarrow \emptyset$  ▷ The candidates for the trapdoor  $td$ 
8:     for  $kw \in \mathcal{W}_{\text{atk}}$  do
9:        $s \leftarrow -\ln(|C_{\text{sub}}^{\text{atk}}[kw] - C_{\text{sub}}^{\text{query}}[td]|)$ .
10:      Append  $\{"kw": kw, "score": s\}$  to dists.
11:      Sort dists in descending order according to the score.
12:      certainty  $\leftarrow \text{score}(\text{dists}[0]) - \text{score}(\text{dists}[1])$ .
13:      Append  $(td, \text{kw}(\text{dists}[0]), \text{certainty})$  to temp_pred.

    // 3. Either stop the algorithm or keep refining.
14:   if  $|\text{unknownQ}| < \text{RefSpeed}$ 
15:     final_pred  $\leftarrow \text{Known}\mathcal{Q} \cup \text{temp\_pred}$ 
16:     unknownQ  $\leftarrow \emptyset$  ▷ Stopping criteria
17:   else
18:     Append the RefSpeed most certain predictions to KnownQ.
19:     Update  $C_{\text{sub}}^{\text{atk}}$  and  $C_{\text{sub}}^{\text{query}}$  based on the new known queries.
20: return final_pred.

```

are added to the set of known queries. We detail this process in Algorithm 4. This algorithm introduces a new parameter: the refinement speed RefSpeed to decrease attack runtime. However, if the refinement speed is chosen too large, it is very likely that wrong predictions are added to the known queries. Overall, the time complexity of the algorithm in Algorithm 4 is $\mathcal{O}(\frac{|\mathcal{Q}|}{\text{RefSpeed}} \cdot |\mathcal{Q}| \cdot m_{\text{atk}} \cdot k)$. Notice that the runtime of the refined score attack increases by the factor $\frac{|\mathcal{Q}|}{\text{RefSpeed}}$ in comparison to the base attack, thus RefSpeed decreases the runtime by a multiplicative factor. As a result, the refined score attack is finished in minutes, whereas Pouliot and Wright's attack [PW16] requires several hours.

Each iteration of this algorithm is divided into three phases:

- (1) Remove all (attacker-)known queries from the queries to be recovered.
- (2) For each unknown query, find the best matching keyword candidate and compute its certainty score.

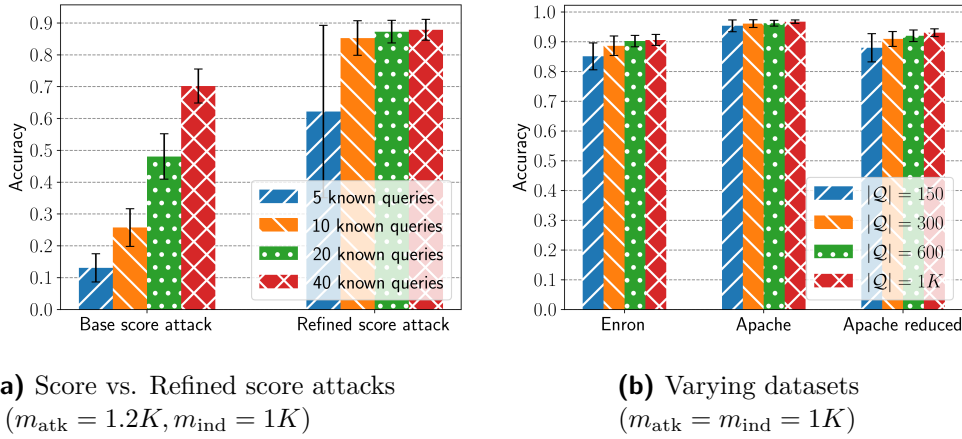


Figure 6.10 / Performances of the refined score attack (Common parameters: $|D_{\text{atk}}| = 12K$, $|D_{\text{ind}}| = 18K$, $|Q| = 150$, $\text{RefSpeed} = 10$)

- (3) If there are more than RefSpeed unknown queries, we keep refining the results: the most certain predictions (RefSpeed many) are added to the known queries. Then, add the columns corresponding to the new known queries to the co-occurrence sub-matrices ($C_{\text{sub}}^{\text{atk}}$ and $C_{\text{sub}}^{\text{query}}$) and start a new iteration. Otherwise, the algorithm stops and returns the queries imputed since the first iteration.

The main benefit of the refined score algorithm is the use of more information available to the attacker. The initial score attack only uses a small part of the co-occurrence matrices (i.e., the co-occurrence sub-matrices). This refinement iteratively imputes new known queries which increases the size of these sub-matrices. We optimize the use of the attacker's knowledge (the co-occurrence matrices) in order to minimize the amount needed for a successful attack (the known queries).

6.5.2 Experimental results

General comparison In Fig. 6.10a, we compare the accuracy of the base and the refined versions of the attack. We fixed $m_{\text{ind}} = 1K$ and $|Q| = 150$. Each bar represents the average accuracy over 50 simulations done with the same parameters. In Fig. 6.10a, we see that the base algorithm needs 40 known queries to reach 70% of accuracy while the refined score algorithm reaches 85% with only 10 known queries. Even with only 5 known queries, the refined score algorithm achieves 62% of accuracy with a standard deviation of 13 percentage points.

If not stated differently, we fix $|\mathcal{W}_{\text{atk}}| = |\mathcal{W}_{\text{ind}}|$ to simplify the experiment understanding. However, it is likely that the similar vocabulary and the queryable vocabulary are not identical. For the experimental results depicted in Fig. 6.10a, we used $|\mathcal{W}_{\text{atk}}| = 1.2K$ and $|\mathcal{W}_{\text{ind}}| = 1K$. By choosing the vocabulary sizes such that $|\mathcal{W}_{\text{atk}}| > |\mathcal{W}_{\text{ind}}|$, we increase the probability that $\mathcal{W}_{\text{atk}} \cap \mathcal{W}_{\text{ind}} = \mathcal{W}_{\text{ind}}$. In this case, all queries can be recovered theoretically. In other words, as the size of $\mathcal{W}_{\text{atk}}$

increases, the accuracy upper bound as stated in Equation (6.4) in Section 6.2.3 potentially increases.

The refined score attack yields highly accurate results within minutes. It recovers most of the queries and assumes less attacker's knowledge than [IKK12] and [CGPR15] attacks. In [CGPR15], [CGPR15] report the average accuracy of the [IKK12] attack is around 30% for an attacker knowing 95% of the indexed documents for $|\mathcal{W}_{\text{ind}}| = 500, |\mathcal{Q}| = 150, \text{Known}\mathcal{Q} = 8$. With the same parameters, [CGPR15] achieves 70% accuracy. In Fig. 6.10a, we see that for a vocabulary size twice as large and less known queries, i.e., $|\mathcal{W}_{\text{ind}}| = 1K, |\mathcal{Q}| = 150, \text{Known}\mathcal{Q} = 10$, the refined score attack also obtains 85% *without partial knowledge* of the encrypted documents.

Query set size In both the [IKK12] and [CGPR15] attacks, the number of observed queries was set as 15% of all possible queries. We investigate the role of this choice in Fig. 6.10b. With a fixed number of known queries, the attack accuracy increases with a larger query set. Intuitively, this demonstrates that our attack efficiently uses attacker's knowledge, i.e., more observed queries imply more attacker's knowledge. In contrast, both [IKK12] and [CGPR15] had steady results even for an increasing number of known queries, indicating some inefficient use of attacker's knowledge.

Different email corpus We compare the accuracy on Enron and on Apache in Fig. 6.10b. For these simulations, we used three document sets: Enron ($|D_{\text{atk}}| = 12K, |D_{\text{ind}}| = 18K$), Apache ($|D_{\text{atk}}| = 20K, |D_{\text{ind}}| = 30K$) and 'Apache reduced' ($|D_{\text{atk}}| = 12K, |D_{\text{ind}}| = 18K$). 'Apache reduced' is the Apache dataset truncated in order to have as many emails as in Enron. Apache has slightly better results than Enron while the emails contain a richer vocabulary and longer emails. Thus, our attack could be effective on a wide range of documents. Moreover, the bar plot shows that 'Apache reduced' has lower results than Apache. Our results on 'Apache reduced' are closer to those on Enron. Since Apache and 'Apache reduced' share a common distribution and only differ in size, it shows (once again) that our attack is sensitive to the amount of attacker's knowledge. In this case, the part of the attacker's knowledge which is increased is the similar document set.

Document set similarity Recall the similarity definition for the document sets from Section 6.2.3. For a better understanding of this new definition, we performed two experiments:

- (1) Using Enron as an attacker document set and Apache as an indexed document set
- (2) Fixing the size of the indexed document set and attacking it with similar document sets of varying size.

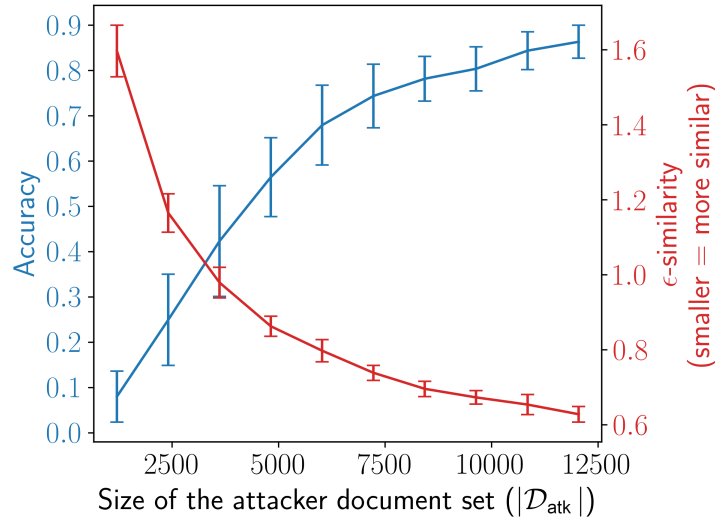


Figure 6.11 / Accuracy and ϵ -value of set similarity for varying attacker document set sizes with Enron ($|D_{ind}| = 18K, m_{atk} = m_{ind} = 1K, |Q| = 150, |KnownQ| = 15$)

During the first experiment, over 50 repetitions, we recovered at best 5 queries. This bad performance is explained by the fact that the Enron dataset has a low similarity with the Apache dataset ($\epsilon = 10.2$). Further, the attacker vocabulary ($\mathcal{W}_{sim}$) and the queryable vocabulary ($\mathcal{W}_{real}$) only have 56% of their keywords in common.

Recall from Section 6.2.3 that the joint keywords are an upper bound for the attack accuracy. The disjoint vocabulary set combined with the high ϵ value between Enron and Apache explains the low attack accuracy for the first experiment. Enron is composed of emails sent in a company, while Apache is composed of emails from a mailing list dedicated to a highly technical project. This important difference in the nature of the emails results in two very different keyword distributions (i.e., a very low similarity between the document sets).

We show the results of the second experiment in Fig. 6.11. By varying the size of the attacker dataset, the co-occurrence matrices of the attacker dataset and the indexed dataset diverge more or less. In other words, this size reduction applies noise to the attacker's word-word co-occurrence matrix compared to the one computed with the complete dataset. We preferred to apply this size reduction instead of applying a synthetic gaussian noise (as is done in the [IKK12] attack paper) to the matrix because the added noise is more realistic this way. Fig. 6.11 shows that reducing the attacker document set size leads to increased ϵ values. Hence, it is an efficient way to decrease similarity.

In Fig. 6.11, we observe that the smaller the attacker dataset is, the less similar the document sets are. Hence, the less accurate the attack results are. When the attacker dataset size is divided by 2, e.g., from 12K documents to 6K, we still achieve an average accuracy of 68%. Thus, the refined score attack shows a

certain degree of robustness against decreased similarity. However, if we further reduce the size of the dataset, the accuracy is also further reduced until we have an ineffective attack.

6.5.3 Attack analysis

Role of the amount of information The refined score attack is sensitive to the amount of information given to the attacker. The more information the attacker has, the higher the attack accuracy is. This holds true for each piece of information owned by the attacker: document set, observed query set, and known queries. This was not the case in the previous attacks, especially for [IKK12] and GCPR, which had identical results with and without known queries. [IKK12] presented an accuracy of 80% regardless of the percentage of known queries (from 0 to 25% in their article). [CGPR15] only presented results without known queries for their count attack even if it could use them.

Technical comparison with related attacks Technically, all query-recovery attacks solve a matching problem between trapdoors and keywords based on specific background information available to the attacker. [IKK12] assumes partial knowledge of the indexed documents together with known trapdoor-keyword mappings. [IKK12] describes the matching problem as an optimization problem that minimizes the distance between the trapdoor co-occurrence matrix and the keyword co-occurrence matrix. [CGPR15] makes similar assumptions while, in practice, it does not require known trapdoor-keyword mappings. [CGPR15] iteratively filters keyword-trapdoor candidates for which the differences between the occurrences (computed from attacker documents and from observed queries) do not match. [BKM20] and [OK21] both propose attacks using query volume information only. [BKM20] assumes an attacker can identify known documents in the index and thus still requires partial knowledge of indexed documents. They represent two bipartite graphs: one connecting indexed documents to the queries and one connecting known documents to keywords; then, they match query nodes with keyword nodes by iteratively refining the candidates using multiple filtering steps. [OK21] formulate an optimization problem based on maximum likelihood estimators, which assumes a distributional knowledge of the indexed documents plus knowledge about query frequency.

Instead of partial index information, we focus on a few attacker-known keyword-trapdoor pairs, which we use to score every keyword-trapdoor candidate. We then iteratively add pairs with the highest scores to the attacker-known pairs to improve our knowledge and refine further predictions. This avoids complex optimization problems and the requirement of knowledge about indexed documents. The scoring and its corresponding iterative refinement are the core novelties of our attack.

As highlighted in [BKM20], all prior attacks require exact knowledge of the queryable vocabulary (i.e., the client's keyword universe). Our attacker does not require such knowledge and builds her own vocabulary. Considering the following

setup: $|D_{\text{atk}}| = 12K$, $|D_{\text{ind}}| = 18K$, $m_{\text{atk}} = m_{\text{ind}} = 1K$, $|Q| = 300$, $|\text{Known}Q| = 15$, with an exact knowledge of the queryable vocabulary, we obtain an average accuracy of 92%. On the other hand, when the attacker builds her own knowledge, we obtain an average accuracy of 87%. This accuracy decrease is a direct consequence of the accuracy upper bound presented in Equation (6.4) of Section 6.2.3.

6.6 Attack mitigation

6.6.1 Existing countermeasures

To mitigate leakage-abuse attacks, several countermeasures have been proposed in [CGPR15,XYW⁺19]. We divide these countermeasures into two categories: padding (formalized by [CGPR15]) and obfuscation. [IKK12] proposed a first countermeasure that could be assimilated into padding. Padding consists in adding fake entries, i.e., fake keyword-document pairs. The user can easily filter these false-positive results when they receive the database response. With padding, there is no entry removal because it could impact the search results (i.e., no false negative results). To harden this countermeasure, [XYW⁺19] proposed a method to produce fake entries that cannot be distinguished from the real entries by an attacker. [CLRZ18] presented a new kind of countermeasure: obfuscation. First, it uses code erasure to divide the documents into shards. Thanks to code erasure, false negative results are allowed because the user does not need every shard to reconstruct the document. After having computed the shards, the algorithm adds and removes shards from the results. The removal rate is chosen so the reconstruction rate for matching documents is close to 100%. Thus, false-negative shards do not result in false-negative documents.

[CLRZ18] also presented an improved attack scenario where the attacker knows which shards belong together. In this case, the countermeasure corresponds to padding because the attacker knows that all the reconstructed documents are either matching document or a false-positive result. Moreover, he knows that the proportion of matching documents which is not reconstructed is negligible. Therefore, if the attacker only keeps the reconstructed files, he would have all the matching documents plus some false-positive results.

These countermeasures have been proposed to mitigate known-data attacks, but they are also suitable for inference attacks since they alter the co-occurrence matrix C^{query} inferred from the queries. Therefore, padding and obfuscation should be also effective in mitigating our attack.

6.6.2 Experimental results

To test the possibility of mitigating our attack, we implemented the padding presented in [CGPR15] and the obfuscation presented in [CLRZ18]. For padding, we use the countermeasure proposed by [CGPR15], which is well established but

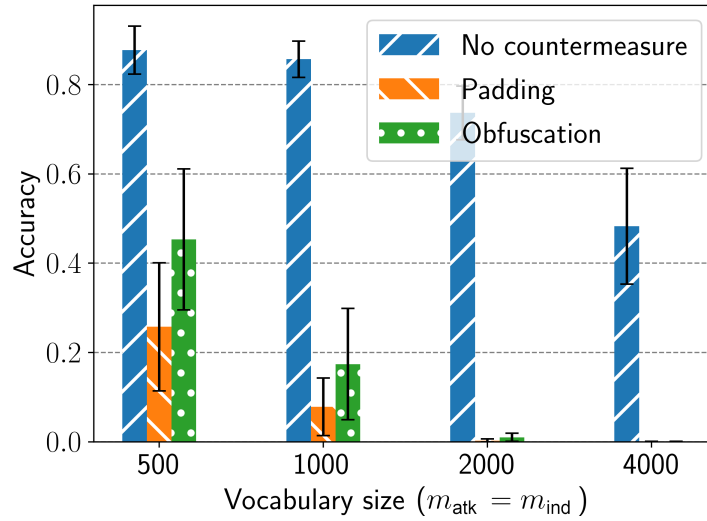


Figure 6.12 / Comparison of the accuracy for countermeasures ($|D_{\text{atk}}| = 12K$, $|D_{\text{ind}}| = 18K$, $|\mathcal{Q}| = 0.15 \cdot m_{\text{ind}}$, $|\text{Known}\mathcal{Q}| = 15$; Padding: $n_{\text{pad}} = 500$; Obfuscation: $p = 0.88703$, $q = 0.04416$)

the hardening proposed by [XYW⁺19] would not provide highly different results since we do not try to filter fake entries. Fig. 6.12 describes the average accuracy of the refined score attack over 50 simulations for several vocabulary sizes. For the padding, we used a padding size $n_{\text{pad}} = 500$. For the obfuscation, we used the parameters used by Chen against the “improved” attack: $p = 0.88703$ the rate of false-positive shards, $q = 0.04416$ the rate of false-negative shards.

Fig. 6.12 clearly shows good mitigation from both countermeasures. For small vocabularies, the accuracy can still be considered as too high. However, as the vocabulary size grows, the accuracy becomes small and negligible for big vocabularies. This figure should not be used to compare the efficiency of the countermeasures. Padding performs better than obfuscation because the padding size we chose is high. For example, when $|\mathcal{W}_{\text{ind}}| = 1K$, the number of entries is increased by 32% because of padding. When $|\mathcal{W}_{\text{ind}}| = 4K$, the number of entries is increased by 166% because of padding. These fake entries create several types of overheads, including storage, communication and computation. [CLRZ18] chose $p = 0.88703$ and $q = 0.04416$ to minimize the overheads then, it is likely that obfuscation can achieve results equivalent or better with bigger overheads. We leave the comparison of these countermeasures and their overheads for future work. Our experiments highlight the importance of hiding the document access pattern to mitigate the refined score attack.

Our attack provides a matching score which can help to identify good predictions. When $|\mathcal{W}_{\text{ind}}| = 500$, the average accuracy for padding is 35% and for obfuscation 47%, the refined score attack can successfully identify a non-negligible part of the correct predictions thanks to the matching score. It is needed to define a

maximum query recovery rate, so the countermeasure parameters are chosen so that there is no attack with an accuracy higher than this threshold. An analogy with encryption security is possible: the attack accuracy is the attacker advantage and the maximum query recovery is the threshold under which the advantage is considered negligible. We leave this direction for future work.

Conclusion

We introduced a highly effective inference attack against SSE. The refined score attack achieves an accuracy (i.e., query-recovery rate) of 90% while only using documents *similar* to the encrypted documents. Previous attacks could only achieve equivalent results by assuming that attacker knows a significant part of the encrypted documents (from 20% for [BKM20] to 70% for [CGPR15]). Our attack provides devastating results while avoiding the strong assumption that the attacker knows a part of the encrypted documents. Unlike the existing attacks we assume few known queries (around 10 known queries in our experiments over different datasets of varying size) rather than knowing the plaintext of a substantial part of the encrypted documents. We argue that it is more realistic to obtain few known queries (e.g. using active attacks) than to obtain a part of the documents indexed. Thus, the refined score attack is more easily performed while previous attacks had a limited number of realistic use cases. One conclusion of our experiments is that the more knowledge the attacker has, the better our refined score attack performs. Despite the simplicity of this statement, it was not observed in previous attacks. This sensitivity to the information amount highlights an optimized attacker's knowledge use as opposed to a relative underutilization of this knowledge by some existing attacks. Finally, we showed that the existing countermeasures (padding and obfuscation) can effectively mitigate the refined score attack.

7 Revisiting the attacker's knowledge in inference attacks against Searchable Symmetric Encryption

While the previous chapter (along with other works [OK21,OK22]) showed that accurate inference attacks are possible, it vaguely assumes an attacker's knowledge containing data “similar” to the indexed data. These attacks are indeed highly accurate, but one question remains to assess their practicality: how likely is it for an attacker to obtain “similar enough” data?

This chapter provides novel statistical tools usable on any inference attack to analyze its sensitivity to data similarity. First, we introduce a mathematical model based on statistical estimators to analytically understand the attackers' knowledge and the notion of similarity. Second, we conceive statistical tools to model the influence of the similarity on the attack accuracy. We apply our tools on three existing attacks to answer questions such as: is similarity the only factor influencing accuracy of a given attack? Third, we show that the enforcement of a maximum index size can make the “similar-data” assumption harder to satisfy. In particular, we propose a statistical method to estimate an appropriate maximum size for a given attack and dataset. For the best known attack on the Enron dataset, a maximum index size of 200 guarantees (with high probability) the attack accuracy to be below 5%.

This chapter is drawn from [DLHP25].

7.1 Introduction

Recently, [DHP21] and respectively presented [OK22] the *refined score attack* and the *IHOP attack*, which **both achieved high recovery rates as inference attacks**. As explained in Chapter 6, before [DHP21], attacks against SSE were only effective in a known-data setting. These works are then the first efficient inference attacks.

Inference attacks require the attacker's data to be “sufficiently similar” to the indexed data (as formulated in [DHP21]). However, no existing work discussed whether this vague assumption was realistic.

Existing works [DHP21,DST23] highlighted the link between dataset similarity and attack accuracy. While this observation is interesting, it does not answer a core question: **how likely is it for an attacker to obtain a “sufficiently similar” dataset?** This question is the *starting point of this chapter*.

Related work Recent works in SSE aimed at assessing the practicality of the attacks. On the one hand, [KKM⁺22] developed a Python framework to standardize the attack simulation commonly used in attack papers. On the other hand, multiple works provided different perspective on the leakage of SSE schemes [BGW24, KM23, KMPP22]. While these works analyzed the SSE leakage, we study another part of the attacker's knowledge: the attacker's dataset.

Our contributions This chapter provides several statistical tools to better analyze the sensitivity of SSE attacks to data similarity. In particular, we propose:

- (1) A *mathematical model for the attacker's knowledge* based on statistical estimators. This enables an analytical study of the similar-data assumption.
- (2) A *statistical method to analyze the accuracy of any attack based on a similarity metric*. We apply it to compare three existing attacks and study questions such as: is similarity the only factor influencing the attack accuracy.
- (3) Based on our analysis of the similarity, we show that *setting a maximum index size can limit the attack accuracy*. We also provide a statistical method to estimate a maximum size based on a specific attack and dataset.
- (4) A method to simulate attacker's data issued from a data breach; a recurrent motivation for the existence of similar data. Applying it on the Apache dataset shows that data breaches do not always provide *similar enough* data.

Preliminaries

We will reuse the same notations as in Chapter 6 (see Table 6.1).

Attacks As this chapter is a follow-up of Chapter 6, it also focuses on (passive) inference attacks against static SSE schemes with access and search pattern leakage. In particular, it demonstrates our statistical method on the score attacks (introduced in Chapter 6) and the IHOP attack [OK22], because they are the only two efficient inference attacks¹. IHOP formulates the attack problem as an assignment problem (i.e., assigning queries to keywords) and solve it using an efficient iterative algorithm based on linear assignment. This attack does not require known queries contrary to the score attacks.

Attacker's model This chapter considers the same threat model and attacker's knowledge as Chapter 6 (see Section 6.2). The only difference is that we assume that the attacker's knows the number of indexed documents and the keyword universe. These two conservative assumptions simplify the statistical analysis and ensure a more understandable argumentation. We reuse the same notations reported in Table 6.1.

¹Recent works such as [HCSL24] written after or concurrently to this contribution have introduced new efficient inference attacks.

Empirical conclusions Section 7.3 and 7.4 present robust statistical methods to obtain empirical conclusions (e.g., parameters influencing the attack accuracy). We demonstrate our methods on state-of-the-art attacks (i.e., the Score, Refined Score, and IHOP attacks), but these methods are usable on any future passive query-recovery attacks. Thus, our empirical conclusions hold for these attacks specifically, and our methods should be applied on any future attack to confirm the conclusions on them.

Empirical conclusions are inherent to statistical approaches: since statistics draw conclusions based on experimental data, the conclusions hold for any data drawn from the same distribution... any new data distribution (i.e., new attack in our case) requires to reproduce the analysis. Conclusions conditioned to a data distribution is then common in statistics, and they have a clear motivation: they enable *studying phenomena too complex to be analyzed analytically*.

Our main goal is to provide novel statistical tools to analyze and understand attacks against SSE. Our empirical conclusions demonstrate the potential of our statistical tools and put a new light on the existing literature. Nevertheless, note that conclusions from Section 7.2 hold for all attacks because they rely on theoretical analysis.

Experimental setup We perform our experiments on Debian Bullseye with a 4-core processor and 8 GB of memory. We use three datasets: the commonly used Enron email dataset [KY04] (30,109 emails contained from `_sent_mail` folders), the Apache mailing dataset (the 50,878 emails from the “java-user” mailing list from the Lucene project for 2002-2011), and the Blog Authorship dataset [SKAP06] (681,288 blog posts written by 19,320 authors), never used in an existing attack paper. Due to hardware constraints, we only use 200K posts simultaneously (picked uniformly at random) from this last dataset.

Since we are studying inference attacks, the dataset is split into two disjoint subsets of varying sizes to create D_{atk} and D_{ind} . By default, we perform this split uniformly at random, but Section 7.5 explores an alternative splitting method.

The keyword extraction and accuracy measurement is performed like in Chapter 6.

Our Python codebase (partially reusing the source code from [DHP21, OK22]) is publicly available².

7.2 Revisiting the notion of similar data

Essential notations Let D_{atk} (resp. D_{ind}) be the attacker’s dataset (resp. indexed dataset) and n_{atk} (resp. n_{ind}), its size. The index can be queried on any word w contained in the keyword universe $\mathcal{W}$ (which contains m keywords). The keyword co-occurrence matrix C^{atk} (resp. C^{ind}) is extracted from D_{atk} (resp. D_{ind}).

²<https://github.com/MarcTOK/Statistical-Leakage-SSE-attacks>

The matrices are built based on the keyword universe $\mathcal{W}$, so they are of dimension $m \times m$. Let $y \xleftarrow{R} \mathcal{Y}$ denote the sampling of y from the probability distribution $\mathcal{Y}$, and p_{ij}^X denotes the following probability $\mathbb{P}(\text{keywords } (i, j) \text{ appear in } d \xleftarrow{R} D_X)$.

7.2.1 Dataset similarity

[DHP21] introduced the notion of dataset similarity to quantify the divergence between the attacker's dataset and the indexed dataset.

Let C^{atk} (resp. C^{ind}) be an $m \times m$ co-occurrence matrix such that: $C_{ij}^{\text{atk}} = \text{Coocc}(D_{\text{atk}}, w_i, w_j)$ (resp. $C_{ij}^{\text{ind}} = \text{Coocc}(D_{\text{ind}}, w_i, w_j)$) for all keywords $w_i, w_j \in \mathcal{W}$. The query co-occurrence matrix C^{query} is a restriction of C^{ind} with an unknown permutation (NB: the attacks aim to find the permutation): if $q_a = \text{Trapdoor}(K, w_i)$ and $q_b = \text{Trapdoor}(K, w_j)$ then $C_{ab}^{\text{query}} = C_{ij}^{\text{ind}}$. [DHP21] defined the $m \times m$ similarity matrix of D_{ind} and D_{atk} over the keyword universe $\mathcal{W}$:

$$\text{SimMat} = \frac{C^{\text{ind}}}{n_{\text{ind}}} - \frac{C^{\text{atk}}}{n_{\text{atk}}} \quad (7.1)$$

Using this matrix, they proposed a similarity metric ε :

Definition 7.2 (ε -similarity). D_{ind} and D_{atk} are ε -similar if $\|\text{SimMat}\| \leq \varepsilon$

Definition 7.2 uses the Frobenius norm (i.e., the equivalent of Euclidean norm for matrices) as matrix norm. In the rest of this chapter, we refer to ε as the smallest value that satisfies this inequality (i.e., $\varepsilon = \|\text{SimMat}\|$).

Alternative metrics [GPP23] introduced two other similarity metrics: “absolute distance” and “Modified Probability Score”. The absolute distance has the same formula as the ε -similarity but uses the infinity norm instead of the Frobenius norm. The Modified Probability Score relies on conditional probabilities.

Metric choice There is no universally better metric; the choice of metric is subjective. This chapter uses the ε -similarity because its formula is convenient for mathematical analysis. In other contexts, the Modified Probability Score might be preferable: e.g., if the link between the metric and conditional probability is necessary for mathematical analysis.

Dataset similarity assumption All inference attacks assumes that there exists a sufficiently small ε such that D_{ind} and D_{atk} are ε -similar. If we assume no similarity, the attacker could use random data. This chapter will explore what is a “sufficiently small” ε , and whether such similarity can be obtained in practice.

7.2.2 The attacker's knowledge is inherently noisy

To analyze the similarity assumption, we first need a proper mathematical model for the co-occurrence matrices. As in machine learning (ML), we consider a dataset as a sample of a random distribution. The properties of the underlying probability distribution could bring more or less uncertainty depending on the use case. We will model this uncertainty and show its impact on the similarity assumption.

Mathematical model Let D_{atk} and D_{ind} be two datasets composed of documents that are represented as binary vectors of length m . Each vector component i indicates whether the keyword w_i is contained in the document. Furthermore, let $\mathcal{X}_{\text{atk}}$ denote the random variable that describes the experiment of sampling a document (i.e., a binary vector of length m) from the same probability distribution as given by the documents in D_{atk} . In other words, $\mathcal{X}_{\text{atk}}$ is a vector of dependent Bernoulli random variables. It implies that the co-occurrence matrix C^{atk} is composed of realizations of dependent Binomial variables, where $C_{ij}^{\text{atk}} \stackrel{R}{\leftarrow} \mathcal{C}_{ij}^{n_{\text{atk}}, p_{\text{atk}}} = \mathcal{B}(n_{\text{atk}}, p_{ij}^{\text{atk}})$. The probability p_{ij}^{atk} corresponds to the probability that the keywords i and j both appear in a document from D_{atk} .

Acknowledging the dependence between the Binomial variables is essential because it significantly complicates the mathematical analysis. To be convinced of the dependence, let us consider the probabilities p_{ij} and p_{ii} (i.e., the probability of keyword i appearing in a document). We have $p_{ij} \leq p_{ii}$ because the event “keyword i appears in a document” contains the event “keywords i and j appears in the same document”.

Let $\mathcal{X}_{\text{ind}}$ be the random variable for dataset D_{ind} defined with the same procedure as $\mathcal{X}_{\text{atk}}$ for dataset D_{atk} . Similarly, we deduce that the co-occurrence matrix C^{ind} is composed of realizations of dependent Binomial variables, where $C_{ij}^{\text{ind}} \stackrel{R}{\leftarrow} \mathcal{C}_{ij}^{n_{\text{ind}}, p_{\text{ind}}} = \mathcal{B}(n_{\text{ind}}, p_{ij}^{\text{ind}})$.

To sum up, $\mathcal{C}^{n_{\text{ind}}, p_{\text{ind}}}$ and $\mathcal{C}^{n_{\text{atk}}, p_{\text{atk}}}$ are two random matrix probability distributions from which C^{ind} and C^{atk} are drawn. We assume the two distributions to be independent because, in inference attacks, the attacker's dataset is considered “different”: implicitly, it means obtained from independent sources.

Statistical estimators The previously defined random distribution ($\mathcal{C}^{n_{\text{ind}}, p_{\text{ind}}}$ and $\mathcal{C}^{n_{\text{atk}}, p_{\text{atk}}}$) **cannot be known exactly**. We are in a classic statistics problem: we do not have access to the exact probabilities, but we know a dataset drawn from an *unknown* probability distribution. We can use the dataset to compute an *estimation* of an *unknown* probability. The interest of statistical estimation is then to approximate these unknown probabilities.

Finding a similar dataset is close to the problem of representative sampling for surveying: the results are not the probabilities themselves (e.g., political opinions) but an estimation of this probability. The larger the sample is, the more precise the estimation. If one compares two samples obtained from populations with

different probabilities, their respective estimators may be close to each other, but it is unlikely. For example, a survey on professors might provide similar results to the same survey on students. However, one understands that it is unlikely, especially with numerous questions (i.e., high-dimensional data).

Analogously, our attacker only observes samples from a random distribution (i.e., documents) but not the probabilities themselves. They can estimate the co-probabilities³ p_{ij}^{atk} (resp. p_{ij}^{ind}) by computing the co-frequencies $\frac{C_{ij}^{\text{atk}}}{n_{\text{atk}}} = \hat{p}_{ij}^{\text{atk}}$ (resp. $\frac{C_{ij}^{\text{ind}}}{n_{\text{ind}}} = \hat{p}_{ij}^{\text{ind}}$)⁴. The co-frequency is the *maximum likelihood estimator* of p_{ij} . The maximum likelihood estimator is an efficient estimator for p_{ij} and converges toward this (unknown) value. The larger D_{atk} (resp. D_{ind}) is, the more precise the estimation will be.

Using the statistical estimators, **we can reformulate the “similar-data” assumption**: an attack is successful if the statistical estimation of p_{ij}^{ind} and p_{ij}^{atk} are close enough (for all $i, j \in \{1 \dots m\}$). While this sounds nearly the same as the initial formulation, it opens new perspectives: the SSE attack problem boils down to a statistical estimation problem. We can then reuse results and methods from statistics literature to understand the shortcomings of SSE attacks.

7.2.3 The impact of dataset sizes on the similarity

A key property of statistical estimators is their convergence when the sample size tends to infinity. The sample size has then a direct impact on the estimation quality, and on the underlying similarity assumption. In our case, the sample size corresponds to the number of documents in the dataset. This subsection investigates *analytically and experimentally* how the dataset sizes influence the dataset similarity (using the ε -similarity).

Equality of co-probabilities First, we make the following assumption: $\forall i, j \in [m], p_{ij}^{\text{ind}} = p_{ij}^{\text{atk}}$. In other words, for all pairs of keywords w_i and w_j from $\mathcal{W}$, the probability of their joint appearance in D_{atk} is equal to the probability of the same event in D_{ind} ⁵.

Analysis of the ε -similarity Let us start from the co-probability estimators (i.e., $\hat{p}_{ij}^{\text{ind}} = \frac{C_{ij}^{\text{ind}}}{n_{\text{ind}}}$ and $\hat{p}_{ij}^{\text{atk}} = \frac{C_{ij}^{\text{atk}}}{n_{\text{atk}}}$) which are central in the attacks. Since they are maximum likelihood estimators, we have asymptotic normality of estimators ([GHI⁺83] Chapter 36, Theorem 3.3). Considering n_{atk} and n_{ind} are sufficiently large, we can approximate the distribution at finite distance by the normal distribution. Let $Z_{ij}^{\text{atk}} \sim \mathcal{N}(0, \sigma_{\text{atk}, ij}^2)$ and $Z_{ij}^{\text{ind}} \sim \mathcal{N}(0, \sigma_{\text{ind}, ij}^2)$ be two Gaussian distributions. Under the equality of co-probabilities, we have:

³We use the term co-probability to refer to the probability of a keyword co-occurrence.

⁴The notation $\hat{x}$ refers to the estimation of the unknown value x .

⁵The equality of co-probabilities does not imply that the distributions are equal (i.e., $\mathcal{X}^{\text{ind}} \sim \mathcal{X}^{\text{atk}}$) because it only fixes $\frac{m(m+1)}{2}$ parameters out of their 2^{m-1} parameters.

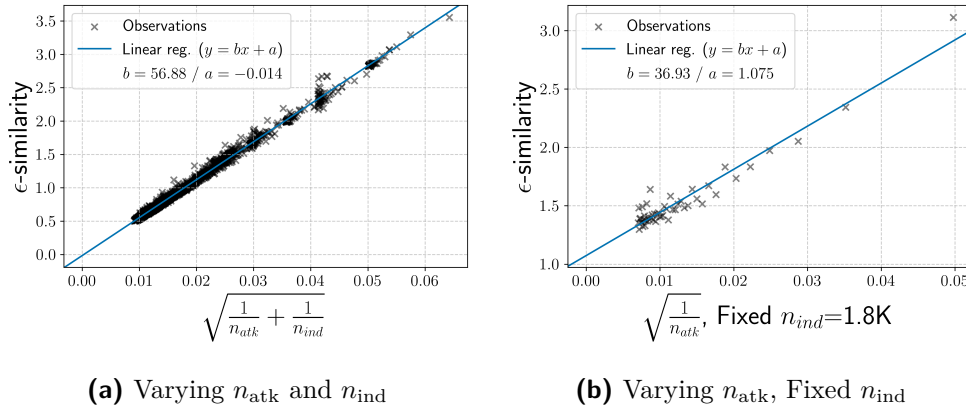


Figure 7.5 / Influence of the dataset size on the similarity (Dataset: Apache)

$$\hat{p}_{ij}^{\text{ind}} = p_{ij} + \frac{1}{\sqrt{n_{\text{ind}}}} Z_{ij}^{\text{ind}}, \hat{p}_{ij}^{\text{atk}} = p_{ij} + \frac{1}{\sqrt{n_{\text{atk}}}} Z_{ij}^{\text{atk}} \quad (7.3)$$

It implies that $(\hat{p}_{ij}^{\text{ind}} - \hat{p}_{ij}^{\text{atk}}) = \sqrt{\frac{1}{n_{\text{ind}}} + \frac{1}{n_{\text{atk}}}} Z_{ij}$, $Z_{ij} \sim \mathcal{N}(0, \sigma_{ij}^2)$. We can reuse this result in the expression of ε :

$$\begin{aligned} \varepsilon = \|\text{SimMat}\| &= \sqrt{\sum_{i,j} \left(\frac{C_{ij}^{\text{ind}}}{n_{\text{ind}}} - \frac{C_{ij}^{\text{atk}}}{n_{\text{atk}}} \right)^2} = \sqrt{\sum_{i,j} (\hat{p}_{ij}^{\text{ind}} - \hat{p}_{ij}^{\text{atk}})^2} \\ \Rightarrow \varepsilon &= \sqrt{\sum_{i,j} \left(\frac{1}{n_{\text{ind}}} + \frac{1}{n_{\text{atk}}} \right) (Z_{ij})^2} = \sqrt{\left(\frac{1}{n_{\text{ind}}} + \frac{1}{n_{\text{atk}}} \right) K}, K \text{ a random variable} \end{aligned} \quad (7.4)$$

We can draw two conclusions from Equation (7.4). First, the size of the attacker's dataset matters as much as the size of the indexed dataset. It seems evident that the more documents the attacker knows, the better the attack is. However, thinking that a bigger indexed dataset helps the attacker could seem counterintuitive. To understand this intuition, we must remember that the attacks rely on comparing two estimators (i.e., the co-occurrence matrices). The attack is unsuccessful if *any of them* is poorly estimated.

Second, fixing a dataset size creates a threshold for similarity. For example, let us fix n_{ind} and observe the effect on Equation (7.4): we have $\lim_{n_{\text{atk}} \rightarrow \infty} \varepsilon = \sqrt{\frac{1}{n_{\text{ind}}}} K$. Hence, even with an infinite-sized attacker's dataset, having an ε -similarity below a certain threshold (proportional to $\sqrt{\frac{1}{n_{\text{ind}}}}$) is unlikely.

Experimental confirmation Fig. 7.5a illustrates this theoretical result. To generate results over a wide range of similarities, we vary the attacker's and indexed dataset sizes vary between 5% and 95% of the Apache dataset. This plot comprises 2500 points with varying n_{ind} and n_{atk} . On the one hand, the linear relationship between $\sqrt{\frac{1}{n_{\text{ind}}} + \frac{1}{n_{\text{atk}}}}$ and ε confirms the symmetric influence of these

variables. On the other hand, Fig. 7.5b shows the similarity threshold created when n_{ind} is fixed. Indeed, the linear relationship in Fig. 7.5a has an intercept tending to zero, while the linear relationship in Fig. 7.5b has an intercept equal to 1. This non-zero intercept highlights the existence of a similarity threshold since, even with an infinite-sized attacker's dataset, the average ε -similarity would be 1. This threshold is *specific to the Apache dataset*; each has a different threshold influenced by the data distribution.

7.3 Attack analysis from a data-similarity point of view

Existing works [DHP21, DST23] highlighted an evident link between similarity and attack accuracy: the more similar the (indexed and attacker) datasets are, the more accurate the attacks are. This section goes beyond this simple intuition to fully explore the role of similarity in attack success. We propose a robust statistical method to analyze the accuracy (of a given attack) using the similarity metric. We then apply this method on three existing and three different datasets.

On the one hand, we use this method to understand whether similarity is the only parameter influencing the accuracy of an attack. On the other hand, we use the method to build a more consistent attack comparison methodology.

7.3.1 How to analyze attack success using a similarity metric?

We want to estimate a function $\hat{f}_{\text{Acc}}$ modelling the average attack accuracy in function of the ε -similarity; i.e., $\hat{f}_{\text{Acc}}(\varepsilon) = \mathbb{E}(\text{Acc})^6$. A continuous function provides a more detailed understanding of the attack's strengths and weaknesses. Such functions enable extrapolating the experimental results and precisely identify gaps in the literature. Attack papers usually represent the accuracy on a finite set of points, which gives a poor understanding of the complete attack behavior. Sections 7.3.2 and 7.3.3 will show how we can use these functions to provide novel insights about the existing attacks.

Estimating the accuracy function Before estimating the attack accuracy function, we need attack results. Following the results from Section 7.2.3, we vary the size of the datasets to obtain attacks results with varying ε -similarity.

Estimating a function based on this experimental data corresponds to a regression problem. First, we can point out that the function cannot be linear since the accuracy is in $[0, 1]$. Hence, we cannot directly use linear regression on the raw data. To circumvent this difficulty, we estimate a function $\hat{f}'_{\text{Acc}}(\varepsilon) = \text{logit}(\text{Acc})$. The logit function is defined as $\text{logit}(p) = \log(\frac{p}{1-p})$ with $p \in (0, 1)$ and has a

⁶ $\mathbb{E}[X]$ is the expected value of the random variable X .

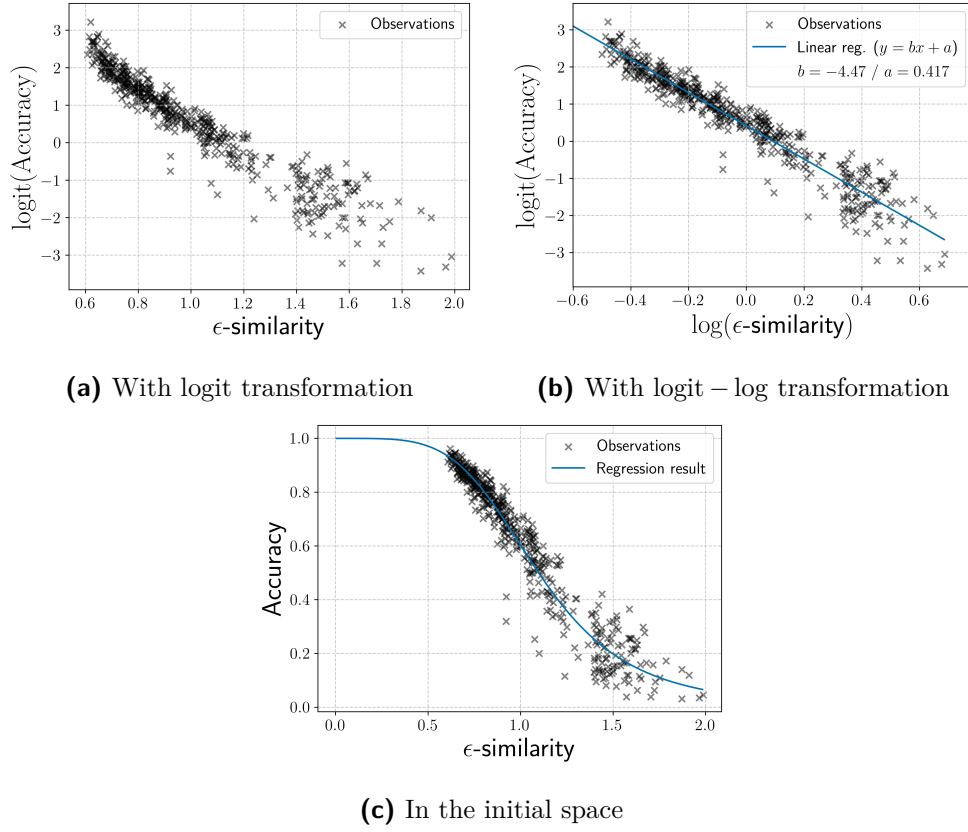


Figure 7.6 / Analysis of the refined score attack on Enron dataset ($m = 1K$)

known inverse function expit . This logit function maps a $(0, 1)$ space into the real number space. This logarithmic transformation is common in statistics.

Fig. 7.6a represents the simulation results with the logit transformation. We observe two problems in this figure: we do not have apparent linearity between the variables, and there is a “heteroscedasticity of the noises”. The first problem is simple: the distribution of the points seems flattened when ε grows. The second issue concerns the assumption made when computing a linear regression. A fundamental assumption in linear regression is that the noise distribution is the same at each point of the space. Then, the data should have the same noise distribution for small and high ε , which is not the case in Fig. 7.6a. A recurrent solution in ML to the linearity problem is to apply a logarithm transformation on the x axis. Fig. 7.6b presents the results of a linear regression between $\text{logit}(\text{Accuracy})$ and $\log(\varepsilon)$. Combining these two logarithmic transformations makes the linear relationship more apparent. Moreover, the logarithmic transformation slightly corrected the heteroscedasticity. Scaling methods (where a data point is modified by a scaling factor depending on its ε) exist to perfectly fix heteroscedasticity. However, they add more complexity for equivalent results.

To sum up, we map the attack simulation results in a logit – log space to compute a linear regression (i.e., $\text{logit}(\mathbb{E}(\text{Acc})) = b \log(\varepsilon) + a$). We then deduce the average accuracy function $\hat{f}_{\text{Acc}}(\varepsilon) = \text{expit}(b \cdot \log(\varepsilon) + a)$ with (b, a) the regression

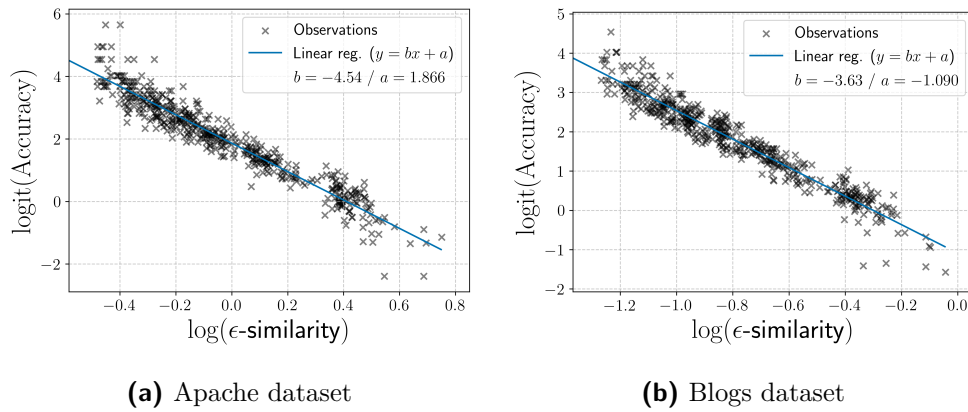


Figure 7.7 / Regression results on others datasets ($m = 1K$)

parameters.

Fig. 7.6 represents the resulting average accuracy function for the refined score attack (obtained on the Enron dataset). This estimation protocol can be executed on any passive attack; e.g., Fig. 7.8 compares the average accuracy function obtained on three different attacks.

Alternative regression methods Our regression model is built on classical statistical approaches (e.g., the logit function) and produces convincing results, but we cannot formally prove that it is the best model to represent the relationship between accuracy and similarity. This situation is perfectly summarized by an aphorism attributed to the statistician George Box [Box76]: “all models are wrong, but some are useful.” In other words, it is commonly accepted that a statistical model cannot capture all the complexities of the real world. However, this relative imperfection does not prevent from using statistical models.

The ML literature has described many other regression models, but our method is simple and provides interpretable results despite an initial non-linear problem. A model is “interpretable” if an observer can understand the cause of a decision [Mil19] (e.g., predict the impact of an input variation). This interpretability is absent in many popular ML models, such as neural networks and random forests. These models can solve non-linear problems efficiently, but are often referred to as “black boxes” due to their lack of interpretability. Our model is interpretable and efficient because it combines linear models with simple logarithmic transformations. Interpretability is a crucial property in our case to foresee the security limits... making our model particularly useful.

7.3.2 The role of similarity in attack success

We can now use our average accuracy functions to understand whether similarity is the only factor influencing the attack success. We can reformulate it as follows: for each attack, is there a unique accuracy function $\hat{f}_{\text{Acc}}(\varepsilon)$ valid for all datasets?

Fig. 7.7 presents the average accuracy function on two other datasets: Apache and Blogs. The linear regression parameters are $(-4.54, 1.87)$ for the Blogs dataset, $(-3.63, -1.09)$ for Apache, and $(-4.47, 0.417)$ for Enron. These parameters significantly differ, so no unique function links the similarity to the accuracy. Thus, **similarity is not the only parameter influencing the accuracy** of the refined score attack.

We can explain this phenomenon using keyword distribution. Consider a hypothetical dataset with m keywords: all keywords appear in all documents. Even with perfect knowledge of the probabilities (i.e., $\varepsilon = 0$), we cannot distinguish the keywords with an accuracy better than a random guess (i.e., $\mathbb{E}(\text{Acc}) = \frac{1}{m}$), because all keywords have the same co-frequencies. Hence, each dataset has a keyword distribution, resulting in more or less distinguishable keywords.

Revisiting the notion of indistinguishability Indistinguishability is not a new notion in SSE [BF17, CGK006, XDZ⁺21]. Attack mitigations usually aim at some keyword/query indistinguishability. The intuition is to have all queries leaking the same information, so the leakage induced by SSE schemes becomes useless to an attacker.

For example, mitigations based on false-positive results [BF17, CGPR15, XYW⁺19] (used in SSE implementations such as ShieldDB [VYS⁺23]) produces so-called “indistinguishable” queries by “smoothing” the keyword frequencies. We argue that these countermeasures do not produce indistinguishability but simply noisy statistics (i.e., noisy co-occurrence estimators). Indeed, the frequencies might be indistinguishable, but the co-frequencies do not automatically inherit this property. To produce an indistinguishable leakage, we would need to smooth all the statistics, from the keyword frequencies to the co-frequencies of m keywords (i.e., the $2^m - 1$ parameters of the random binary vectors defined in Section 7.2.2).

However, the good performances of the mitigations show that noisy statistics are enough to prevent attacks. We do not need indistinguishable leakages; we must only ensure they are realistically unusable. For example, two queries can leak distinct co-frequency information; the attacks will be unsuccessful if this information leakage is too noisy. Hence, it is unnecessary to perfectly protect the queries by enforcing them to have the same co-frequency leakage. To reach a satisfying noise level, Section 7.4 introduces a maximum index size estimation (guaranteeing a low-attack accuracy).

7.3.3 Comparison framework

Finally, we can use the average accuracy functions to design a new attack comparison methodology: (1) generate attack results for each attack, (2) estimate the average accuracy functions, (3) compare the average accuracy functions.

Fig. 7.8 compares the accuracy functions of three recent attacks (on Apache dataset): Score, Refined score, and IHOP attacks. The IHOP and refined score

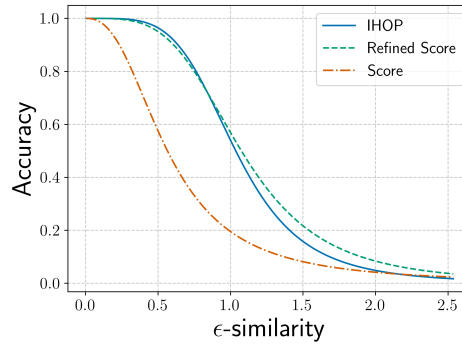


Figure 7.8 / Comparison of the average accuracy functions of the Score, Refined Score, and IHOP attacks ($m = 500$)

attacks outperform the score attack. IHOP obtains slightly better results for smaller ϵ while for higher ϵ , the refined score attack has a small advantage.

This comparison framework simplifies the identification of game-changing attacks. The existing attack papers typically compare attacks over a set of parameters, emphasizing a clear accuracy difference between a novel attack and the state-of-the-art. While this approach identifies improvements, it does not give a full picture of the situation. In our case, a traditional attack comparison between IHOP and Refined Score attack would “zoom in” between $\epsilon = 0.5$ and $\epsilon = 1.0$ to focus on the highest accuracy gains. This “zoom” ignores subtle phenomena, such as the slight advantage of the Refined Score attack on highly dissimilar datasets. Systematically comparing accuracy functions would guarantee attack papers thoroughly analyze the attack behaviors and avoid focusing on a convenient set of parameters. Moreover, the linear regression estimates an average distribution, smoothing the noise over the curve. Noisy experimental results are a recurrent concern that questions whether an accuracy difference is significant.

Nevertheless, linear regression remains a statistical estimation, so some uncertainty remains. We can model this uncertainty using confidence intervals. As for individual experimental results, we can run simple statistical tests to prove a difference to be statistically significant. While performing statistical tests on a large set of individual experimental results is tedious, performing a statistical test on linear regression parameters [CPH95] is much easier. In other words, we can statistically prove that an estimated function is significantly higher than another function instead of proving a significant difference on individual points. Hence, this simplicity should *encourage future attack papers to rigorously prove accuracy improvements using statistical tests on accuracy functions*.

Attack mitigation efficiency Finally, the comparison framework can be extended to attack mitigation comparison. The idea would be to compare accuracy functions obtained with and without countermeasures. The impact of the mitigation on the accuracy function would quantify its efficiency.

7.4 Making the “similar-data” assumption unlikely

Since attack papers require the attacker to have a “sufficiently similar” dataset, one may expect that only the attacker has an influence on the “similar-data” assumption. Indeed, this formulation may imply that the assumption only depends on the quality of the dataset found by the attacker. However, this intuition is untrue: some settings in the SSE deployment can make the “similar-data” assumption unlikely. In particular, this section shows how to set a maximum index size such that it is highly unlikely to find a “sufficiently similar” dataset.

7.4.1 Dataset size, an essential condition to attack success

Section 7.2.3 have explored the influences of dataset sizes on the similarity. A key take-away of this analysis is the similarity threshold induced by a fixed dataset size. Indeed, to reach perfect similarity (with two independent datasets), one needs both an attacker's and an indexed dataset of infinite size. If the indexed dataset has bounded size, the similarity is unlikely to go below a certain threshold (*even with an infinite-sized attacker's dataset*).

Since the accuracy is correlated to the similarity (see Section 7.3), we can deduce that: if the indexed dataset has bounded size, the attack accuracy is unlikely to go below a certain threshold. Based on this intuition, we propose the following (almost) obvious theorem linking dataset sizes and attack accuracy:

Theorem 7.9. For any distribution $\mathcal{D}$ (with D_{ind} and D_{atk} drawn from $\mathcal{D}$), for any keyword universe $\mathcal{W}$, there exists a maximum index size $n_{\text{max}} \in \mathbb{N}$ such that:

For any $n_{\text{atk}} \in \mathbb{N}$, if $n_{\text{ind}} \leq n_{\text{max}}$, the average accuracy of any passive query-recovery attack is lower than or equal to $\frac{1}{m}$.

Proof. We show that with $n_{\text{max}} = 0$, the statement in the theorem is always satisfied: the index contains no document, so all the queries return an empty set. Hence, an attacker can only randomly guess the queried keyword. The average accuracy is then $\frac{1}{m}$ (with m the number of keywords in $\mathcal{W}$). $\square$

Our proof relies on a trivial case (i.e., $n_{\text{max}} = 0$). Many data distributions could verify the theorem with a strictly positive n_{max} . However, some distributions cannot. Let us consider the keyword universe $\mathcal{W} = \{w_1, w_2\}$ and a distribution $\mathcal{D}$ such that $\mathbb{P}_{\mathcal{D}}(w_1) = 1$ and $\mathbb{P}_{\mathcal{D}}(w_2) = 0$. If D_{ind} contains even a single document, the query leakage would be sufficient to recover the queries perfectly: if the result is empty, it is w_2 ; otherwise, it is w_1 .

Moreover, there exist other distributions for which Theorem 7.9 holds for all $n_{\text{max}} \in \mathbb{N}$. For example, if each indexed document contains all the keywords from $\mathcal{W}$, each query returns all the documents, so the leakage is useless (i.e., no attack is better than a random guess). Even with an infinite-sized indexed dataset, the

best average attack accuracy would be $\frac{1}{m}$.

In practice, real-world distributions would have an $n_{\max}$ between these two edge cases. Our goal is now to estimate this value for these distributions.

7.4.2 Estimating the maximum index size

To estimate this maximum index size, we want an upper bound on the attack accuracy and then use it to deduce the maximum index size. Let $f_{\text{MaxAcc}}(n_{\text{ind}}, n_{\text{atk}})$ be the accuracy upper bound in function of the dataset sizes $(n_{\text{ind}}, n_{\text{atk}})$. The maximum index size $n_{\max}$ is such that $\lim_{n_{\text{atk}} \rightarrow \infty} f_{\text{MaxAcc}}(n_{\max}, n_{\text{atk}}) \leq \frac{1}{m}$. We can generalize this formula and consider a generic maximum accuracy threshold $\beta_{\max}$ (in $[\frac{1}{m}, 1]$) instead of $\frac{1}{m}$.

Ideally, we could deduce analytically an accuracy upper bound for all attacks. Unfortunately, a theoretical bound could be non-informative. For example, Section 7.4.1 highlights a distribution for which the attack accuracy is always perfect. Hence, a purely theoretical bound could be too loose due to these edge cases.

To avoid these issues, we will compute this accuracy upper bound using statistical estimation. This estimated upper bound is not absolute, but simply statistical: it is not impossible to be above this threshold; it is only unlikely. This statistical approach can provide tight and informative bounds *for a given attack and a given use case* (represented by a sample dataset).

Estimation protocol Section 7.3 presents the estimation of an average accuracy function. We can extend this approach to estimate our maximum accuracy function f_{MaxAcc} , but we must make two changes to the estimation process.

First, we replace ε by the value $\sqrt{\frac{1}{n_{\text{ind}}} + \frac{1}{n_{\text{atk}}}}$ in the estimation process. This replacement is motivated by the linear relationship between $\sqrt{\frac{1}{n_{\text{ind}}} + \frac{1}{n_{\text{atk}}}}$ and the similarity metric ε discovered in Section 7.2.3. Thanks to this linear relationship, we can replace ε from the average accuracy estimation to build a regression model linking the accuracy to $(n_{\text{ind}}, n_{\text{atk}})$ (instead of the accuracy to ε).

Second, we want an accuracy upper bound (not an average accuracy), so we cannot use linear regression. Instead, our estimation protocol relies on quantile regression [HN07, KB78, KH01]. In a quantile regression, the resulting estimated function describes the quantile of a data distribution⁷ instead of the average case (as linear regression does). A quantile regression computes the parameters (b, a) such that $Q_Y(\alpha) = b \cdot X + a$, for (X, Y) two data distributions and α a quantile level. We refer to [HN07, KB78, KH01] for details about the computation. This quantile regression is ideal for representing a maximum accuracy bound with high probability.

⁷The quantile of level α for distribution Y is defined as follows: $Q_Y(\alpha) = \inf\{y : F_Y(y) \geq \alpha\}$, with F_Y the cumulative distribution function of Y .

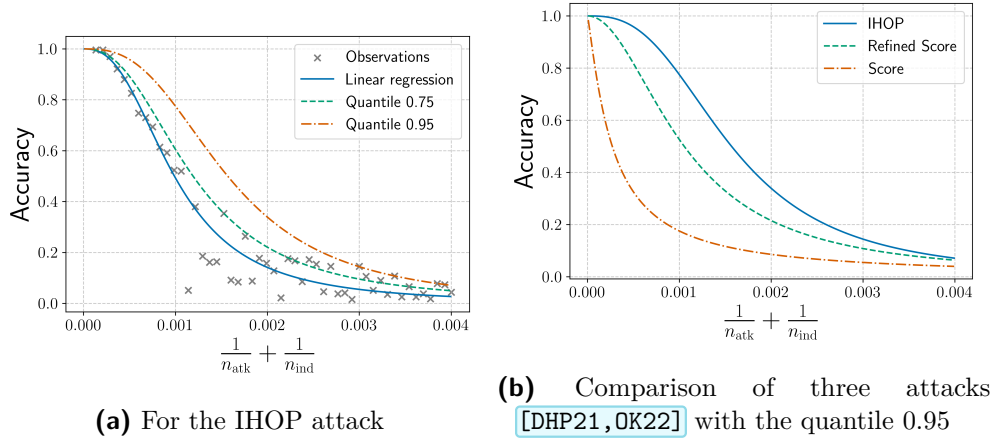


Figure 7.10 / Estimated accuracy upper bounds (500 keywords / Dataset: Enron)

To sum up, our quantile regression estimates (for a high quantile α) the parameter (a, b) such that $Q_{\logit(\text{Acc})}(\alpha) = b \log(\sqrt{\frac{1}{n_{\text{ind}}} + \frac{1}{n_{\text{atk}}}}) + a$. Then, we deduce our estimated upper bound $\hat{f}_{\text{MaxAcc}} = \text{expit}(b \cdot \log(\sqrt{\frac{1}{n_{\text{ind}}} + \frac{1}{n_{\text{atk}}}}) + a)$. Now that we have a satisfying estimation protocol, we only need to generate simulation results (using a sample dataset representative of the use case) with varying dataset sizes and compute the quantile regression.

For the quantile choice, the thresholds 0.1%, 1%, and 5% are recurrent in statistics. The smaller the quantile is, the more conservative the estimation is.

Limitations This statistical approach provides a tight upper bound, but it has a drawback: the upper bound holds for a specific attack on a specific dataset. Thus, the maximum index size must be recalculated if a new attack is proposed, or if the use case changes.

While it is limiting, this observation holds for many attack mitigations. For example, index padding [CGPR15, XYW⁺19] consists in adding false-positive results to the index. These works do not present a theoretical manner to set the padding parameters. Hence, like our maximum index size evaluation, these mitigations rely on cryptanalysis.

Anyway, our goal is not to present a perfect mitigation. We want to show that practitioners can limit the similar-data assumption using a simple technique.

A conservative estimation As our statistical approach provides an attack mitigation usable by practitioners (i.e., setting a maximum index size), we build our estimation protocol to be highly conservative: a real-world attacker is unlikely to reach the accuracy bound. The quantile regression is the main component contributing to this conservative estimation since it provides an upper bound contrary to the average function of the linear regression. Moreover, we also obtain attack results assuming advantageous attack conditions: (1) the attacker knows

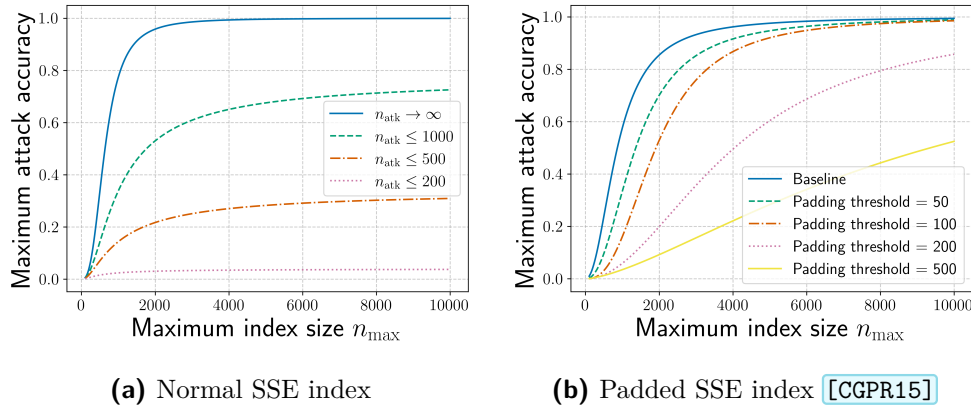


Figure 7.11 / Maximum attack accuracy in function of the maximum index size $n_{\max}$ (quantile regression with $\alpha = 0.95$ / Dataset: Enron / Attack: IHOP [OK22])

the whole keyword universe, (2) the attacker observed all possible queries, (3) we use uniform splitting for dataset generation. *In short, real-world attackers cannot benefit from better conditions.*

Divergence from Theorem 7.9 Our (conservative) statistical estimation bounds the accuracy with “high probability”, while Theorem 7.9 bounds the average case. Our bound is then stronger than the one described in Theorem 7.9. The average case considered in Theorem 7.9 provides a shorter formulation, but we could prove a variant providing an upper bound with high probability using a similar proof.

7.4.3 Example on Enron dataset

Fig. 7.10a shows the estimated accuracy upper bound for the IHOP attack on the Enron dataset. We observe a smooth and coherent upper bound (with regard to the attack results).

We reproduced this estimation protocol for the score and refined score attacks [DHP21] and plot the 0.95 quantiles of all three attacks in Fig. 7.10b. Surprisingly, Fig. 7.10b shows that the IHOP accuracy upper bound is always better than the Refined Score attack. Indeed, Fig. 7.8 of Section 7.3.3 showed that the Refined Score attack was **in average** better than the Refined Score attack on poorly similar datasets. These observations mean that IHOP occasionally reaches very high accuracies, while refined score results have a smaller variance and higher mean. Despite these minor differences, both attacks are accurate in similar scenarios, but IHOP is better because it requires no known queries.

As IHOP has the highest accuracy upper bound, we use this attack to deduce the maximum index size. Fig. 7.11a shows the maximum index size for varying maximum accuracies $\beta_{\max}$. The top curve (in blue) depicts the most conservative setup: adversary with infinite-sized attacker's dataset.

For example, a maximum accuracy of 5% requires a maximum size of 218 docu-

ments per index. While this constraint could be too strict for some applications, some measures could allow satisfying it in practice. Let us assume we store emails (like those from Enron) using SSE, we could (1) enforce a regular cleaning of mailboxes or (2) instantiate a separate secure index for each folder.

To increase this maximum size, we propose two solutions. On the one hand, we could relax our assumptions and consider a finite-sized attacker’s dataset. The three bottom curves of Fig. 7.11a shows the estimation with three different bounds on the attacker’s dataset size: 1000, 500, 200. On the other hand, we could apply other attack mitigations such as padding [CGPR15,XYW⁺19] and compute the maximum index size for a padded index (see Fig. 7.11b).

Our results shows that SSE practitioners can have a direct influence on the “similar-data” assumption. While this opens new directions in SSE attack mitigation (i.e., via maximum index size), further theoretical analysis is necessary to build a complete security framework supporting such statistical approaches.

7.5 Do data breaches provide “sufficiently similar” data?

Attack papers [BKM20,DHP21] often motivate the existence of an attacker’s dataset based on data breaches. SSE attackers could build their dataset using previously leaked documents, but do these data breaches provide “sufficiently similar” data?

Simulating a data breach Inference-attack papers [DHP21,OK22] generate indexed and attacker’s datasets by splitting a real-world dataset (e.g., Apache emails) into two disjoint sets. By default, all papers split the dataset *uniformly at random*. We propose to compare this splitting method to a more realistic alternative: the temporal split.

The temporal split consists in splitting the dataset based on the document timestamp. With email datasets such as Apache, the attacker knows all the emails before a given date, and the index contains all emails sent after this date. This splitting simulates nicely a data breach: the split timestamp corresponds to the breach date; the attacker has access to all data before the breach.

Accuracy discrepancy Table 7.12a presents the accuracy (of the refined score attack [DHP21]) and the ε -similarity obtained using the temporal split with four different splitting dates. Table 7.12b presents the accuracy and the ε -similarity obtained over 100 repetitions of the uniform split. With the temporal split, the attack accuracy is always below 5%, while it is always above 94% with the uniform split. The ε -similarity results confirm this observation since the ε is much higher with the temporal split. The low accuracy with the temporal split is not due to the dataset size since the split on the year 2007 provides sizes equivalent to those used for the uniform split (i.e., around 25K documents per dataset). Therefore,

Year	2003	2005	2007	2009
ε -similarity	4.90	4.07	4.52	4.62
Attack acc. (%)	0.70	2.81	2.81	1.75

(a) Temporal split

	Average	Min.	Max.
ε -similarity	0.62	0.59	0.71
Attack acc. (%)	98.20	94.39	99.65

(b) Random split (100 repetitions)

Table 7.12 / Accuracy of the refined score attack [DHP21] and ε -similarity on the Apache dataset (with $|\mathcal{W}| = 1K$)

the uniform split used in all attack papers can simulate an overly powerful attacker compared to a more realistic method.

The temporal split decreases the similarity (and the accuracy) compared to the uniform split because the distribution of the Apache dataset shifts over time. Note that if the Apache dataset had no distribution shift over time, the uniform and the temporal split should provide the same results.

Other datasets (*without* a distribution shift) would have no accuracy difference between the temporal and uniform split. However, distribution shifts (also referred to as “concept drift” in the literature) are common in real-world datasets; making them a recurrent concern in ML research [CGS⁺21, Tsy04, ŽPG16]. **Hence, this phenomenon could hinder attackers relying on data breaches from building similar datasets.** This observation weakens the recurrent claim of attack papers [BKM20, DHP21, IKK12] motivating the existence of a similar dataset thanks to possible data breaches. The existence of a “sufficiently similar” attacker’s dataset could be a stronger assumption than expected.

What is the correct method for research works? Even if the uniform split simulates an overly powerful attacker, researchers should keep using it to obtain a conservative attack analysis: a real-world attacker would not have better conditions since the uniform splitting simulates a best-case scenario for the attacker (as detailed in appendix of the full version of this chapter).

Attack papers could also provide results based on the temporal split to evaluate the attacks in more realistic setups. Such analysis would incentivize novel attacks leveraging a potential distribution shift to improve their accuracy.

Conclusion

This chapter provided a novel understanding of the “similar-data” assumption used in inference attacks against SSE. From the factors influencing the attack

accuracy to a mechanism to make the assumption unlikely, we raised several concerns regarding the practicality of this assumption in existing attacks. Overall, our results all points at the same direction: the similar-data assumption can be hard to fulfil. In particular, we designed multiple statistical tools usable on any future attack to analyze their performances and their sensitivity to data similarity.

Our results raised new questions about the security of SSE schemes. While the literature focused on leakage minimization to guarantee security, our results (especially the maximum index size) opens the way to new approaches combining statistics with cryptanalysis to estimate real-world risks.

Future works We focused on the most attacked SSE setting, but our statistical approach can be extended to any SSE attack assuming an auxiliary data knowledge (e.g., attacks using query frequency). Such extensions require additional works to adapt our mathematical model and statistical tools.

Part III

MPC protocols for sparse matrix multiplications

This third part presents MPC protocols for sparse matrix multiplication that avoid the access pattern leakage whose risks were demonstrated in the previous part. We focus on matrix multiplication because it is a central operation in many ML algorithms, making it crucial for PPML applications on sparse data. For example, sparse matrix multiplications are essential to implement efficient recommender systems. Our contribution builds upon existing work in secure sparse vector aggregation. Specifically, we extend their approach based on oblivious sorting to perform sparse matrix multiplication efficiently over secret-shared sparse data.

8 Secure sparse matrix multiplications and their applications to privacy-preserving Machine Learning

A few existing works (discussed in Section 3.2) have proposed secure sparse multiplications, but none of them is compatible with the outsourced MPC setting that has become popular in the PPML literature. This chapter introduces protocols to multiply secret-shared sparse matrices, compatible with the outsourced setting. Our protocols rely on oblivious sorting and provide communication cost reductions up to $\times 1000$ on realistic problem sizes compared to secure dense matrix multiplications. Independently of the performance gains, our secure sparse multiplications support applications that are impractical with existing dense solutions.

This chapter is drawn from [DHPR25d].

8.1 Introduction

Several MPC frameworks enable training ML models on secret-shared data [KPPS21, LZLR23, MR18, MZ17, PS20, WGC19, WTB⁺21]. These algorithms make use of existing MPC frameworks such as MP-SPDZ [Ke120]. However, **none of these frameworks provide algorithms optimized for sparse data**. Note that this chapter puts an emphasis on PPML applications, but our results apply to any application involving private sparse data.

The scalability limitations (especially memory issues) of dense plaintext algorithms transfer naturally to dense MPC algorithms. Thus, **MPC frameworks need dedicated sparse algorithms** for ML applications like recommender systems. These frameworks are not fundamentally incompatible with sparse data; there is simply no suitable secure sparse algorithm in the literature.

As detailed in Section 3.2, there exist protocols to sum sparse vectors securely, including sorting-based [AHI⁺22, BGG⁺22], private histograms [BGG⁺22], or private-union-based [BAT21]. Moreover, a few papers [CZW⁺21, CCL⁺21, SGRP19] studied secure sparse multiplications in a two-party setting. However, these sparse multiplication protocols require the data owners to actively participate because one of the computation parties must *know the plaintext sparse data*. Hence, they do not support more than two data owners simultaneously, which is too limiting, as modern ML applications involve thousands of data owners.

Typical ML applications require a more generic MPC setup: outsourced training. In this setting (see Fig. 3.4), the data owners share their secret data with a group of computation servers and disconnect. This distinction between input parties

and computation parties enables supporting an arbitrary number of data owners, contrary to existing works [CZW⁺21, CCL⁺21, SGRP19]. However, the adaptation of these works either results in an inefficient construction or is not straightforward because they require a shareholder to know the plaintext data. The secure multiplication of sparse matrices in the outsourced setting is then **a significant open problem**. *This chapter will focus on this outsourced setting; providing sparse multiplications compatible with any MPC framework.*

Our contributions We present **two secure algorithms** to compute matrix multiplications on secret-shared sparse data. Our algorithms **avoid the memory issues** of dense matrix multiplications on sparse data (e.g., in an experiment from Section 8.3: 19TB using dense multiplications vs. 60GB using ours). Moreover, our algorithms **reduce the communication costs** up to a factor 1000, and we implement two ML applications that are impractical with existing secure algorithms. Third, as Chapter 4 argued for the need of volume pattern leakage, we leverage the properties of real-world sparse data to **mitigate this mandatory leakage** of our sparse multiplications.

8.2 Secure sparse multiplications

This section presents two secure sparse matrix multiplications: matrix-vector and matrix-matrix. Existing secure sparse multiplications are incompatible with the outsourced setting, so we can only compare to the dense baselines. We assume the bit length and number of computation parties are $O(1)$, so our asymptotic complexities are only in function of the matrix size.

Our algorithms rely on secure additions, multiplications, comparisons, oblivious shuffling, and oblivious sorting [HICT14]. On the one hand, sorting-network-based oblivious sorting only requires secure additions, multiplications, and comparisons. On the other hand, we can implement oblivious shuffling using oblivious sorting: generate a random secret-shared value for each element in the input list and obviously sort the list based on these random values. Our algorithms are then **compatible with all secret-sharing schemes** supporting basic arithmetic and comparison operations. However, for optimal performances, one should implement our algorithms using honest-majority sorting/shuffling [HICT14, LWZ11].

8.2.1 Sparse data representation

We use the **tuple representation** [BGS11]: each sparse vector is a list of non-zero tuples: $t = (i, v_i)$ with i the coordinate in the vector and v_i the non-zero value. Let $\text{coord}(t) = i$ denote the coordinate of the non-zero and $\text{val}(t) = v_i$ its value. In other words, the sparse vectors are shared as a list of shares: $\llbracket x \rrbracket = [(\llbracket i_1 \rrbracket, \llbracket v_{i_1} \rrbracket), \dots, (\llbracket i_k \rrbracket, \llbracket v_{i_k} \rrbracket)]$, with $k = \text{nnz}(x)$.

Algorithm 5 Secure sparse vector multiplication**Input:** $\llbracket x \rrbracket$ and $\llbracket y \rrbracket$ are two secret-shared sparse vectors**Leakage:** Number of non-zero elements per vector

```

1: function SPARSEVECTORMULT( $\llbracket x \rrbracket, \llbracket y \rrbracket$ )
2:   Let  $\llbracket z \rrbracket \leftarrow \text{Concatenate}(\llbracket x \rrbracket, \llbracket y \rrbracket)$ 
3:   Let  $\llbracket z \rrbracket \leftarrow \llbracket \text{sort}(z) \rrbracket$  ▷ Asc. sort on the non-zero coordinate.
4:   Let  $\llbracket s \rrbracket \leftarrow 0$ 
5:   for  $i \leftarrow 1$  to  $\text{nnz}(x) + \text{nnz}(y) - 1$  do
6:      $\llbracket \text{obliv. if } \llbracket \text{coord}(z_i) \rrbracket = \llbracket \text{coord}(z_{i+1}) \rrbracket$ 
7:       Let  $\llbracket s \rrbracket \leftarrow \llbracket s \rrbracket + \llbracket \text{val}(z_i) \rrbracket \times \llbracket \text{val}(z_{i+1}) \rrbracket$ 
8:   return  $\llbracket s \rrbracket$  ▷ a scalar

```

8.2.2 Toy protocol: vector-vector

We propose to start with a simple vector multiplication (i.e, inner product). Since dense vector multiplications have an $O(1)$ communication cost (under honest majority), we know in advance that it is unbeatable. Modern ML applications essentially involve matrix-vector and matrix-matrix multiplications, so vector multiplication is not a relevant focus. However, this *toy protocol* provides simple intuitions to understand how to build sorting-based sparse multiplications.

We want to multiply two n -dimensional sparse vectors x and y , represented as tuple lists. Algorithm 5 presents our solution. First, we concatenate the two tuple lists. Note that this concatenation is trivial (and) secure because the sparse vectors x and y are lists of shares. Thus, the concatenation requires no communication; the shareholders simply concatenate locally their list of shares. Then, we sort (*obliviously*) the resulting list by coordinate. If two consecutive tuples have equal coordinates, we multiply their values and sum all the multiplication results to build the vector multiplication value.

We use the notation $\llbracket \text{obliv. if} \rrbracket$ to describe conditional structures on secret-shared values. The presence of conditional branches may seem contrary to the obliviousness goal. Indeed, revealing the executed branch would leak information about the secret-shared value. However, conditional structures on secret-shared values are implemented so that the executed branch remains secret. All the $\llbracket \text{obliv. if} \rrbracket$ instructions are implemented via a combination of arithmetic and comparison operations. For example, we implement the lines 6 to 7 of Algorithm 5 using the following expression: $\llbracket s \rrbracket \leftarrow \llbracket s \rrbracket + (\llbracket \text{coord}(z_i) \rrbracket = \llbracket \text{coord}(z_{i+1}) \rrbracket) \times \llbracket \text{val}(z_i) \rrbracket \times \llbracket \text{val}(z_{i+1}) \rrbracket$; with the secure comparison $\llbracket \text{coord}(z_i) \rrbracket = \llbracket \text{coord}(z_{i+1}) \rrbracket$ outputting $\llbracket 1 \rrbracket$ or $\llbracket 0 \rrbracket$. Such instructions are less readable, so we keep conditional structures in our algorithms with a dedicated notation. Such conditional statements are recurrent in MPC literature: e.g., [AHI⁺22] used conditional statements in their heavy hitters protocol.

Algorithm 5 costs $O((\text{nnz}(x) + \text{nnz}(y)) \log(\text{nnz}(x) + \text{nnz}(y)))$ communications and

computations in $O(\log(\text{nnz}(x) + \text{nnz}(y)))$ rounds. The sorting step induces the superlinear complexity. The multiplication loop is parallelizable, so it only costs $O(\text{nnz}(x) + \text{nnz}(y))$ in $O(1)$ rounds. In comparison, the dense multiplication has $O(1)$ communication costs under honest majority, and an $O(n \log n)$ cost under dishonest majority. Moreover, our sparse algorithm significantly reduces the memory footprint under both threat models.

8.2.3 Matrix-vector

We want to multiply an $n \times m$ matrix X to an m -dimensional vector y . This chapter assumes the leakage motivated in Chapter 4: per-row sparsity. Thus, we can adapt the sparse matrix representation and group the non-zero elements by rows: a sparse matrix would be a list of sparse vectors, one per row.

An intuitive solution is to compute a vector-vector multiplication on each matrix row. This approach is adequate for dense multiplications, but inefficient for sparse matrices. Indeed, this naive extension induces a linear dependency on the number of rows: $O((\text{nnz}(X) + n \cdot \text{nnz}(y)) \log(\text{nnz}(X) + n \cdot \text{nnz}(y)))$ communications and computations. The term $n \cdot \text{nnz}(y)$ comes from the “replication” of y for each row of X . The vector y is sorted independently for each row, which induces this inefficiency. To sum up, we need a dedicated sparse matrix-vector multiplication because the naive extension is inefficient.

Algorithm 6 presents a more optimized solution avoiding the linear dependency on n . The intuition behind this optimization is to build a tuple list sorted as follows $[y_j, x_{i_1,j}, x_{i_2,j}, \dots, y_{j'}, x_{i_3,j'} \dots]$. In this list, we group the elements related to the same column, starting with the element from vector y . We then fix y_j , iterate over all $x_{*,j}$, and multiply it with y_j until we reach the next $y_{j'}$.

Once we finish the vector component multiplications, we sort the multiplication results according to the first coordinate to aggregate all the results with equal coordinates and build the result. Every time two values are aggregated, the first value is replaced by a placeholder. Algorithm 7 details this aggregation step.

After the aggregation (Line 17 of Algorithm 6), we remove the placeholders using Algorithm 8. To avoid revealing information about the non-zero coordinates, Algorithm 8 uses the “shuffle-and-reveal” trick frequently used in secure sparse sum [AHI⁺22, BGG⁺22].

Algorithm 6 costs $O((\text{nnz}(X) + \text{nnz}(y)) \log(\text{nnz}(X) + \text{nnz}(y)))$ communications and computations, but it has a linear round complexity due to the multiplication loop (lines 7 to 14 of Algorithm 6) and the aggregation step (Algorithm 7). In the full version of this contribution, we present recursive algorithms taking inspiration from the logarithmic-round secure maximum (notably implemented in MPyC [Sch18]) in order to build a multiplication loop and an aggregation step with logarithmic round complexity.

Table 8.1 compares the complexities of our sparse algorithm to the dense baselines.

Algorithm 6 Secure sparse matrix-vector multiplication**Input:** $\llbracket X \rrbracket$ a secret-shared sparse matrix, $\llbracket y \rrbracket$ a secret-shared sparse vector**Leakage:** Number of non-zeros per matrix/vector**Notation:** the function RowCoord() (resp. ColCoord()) returns the first/row (resp. second/column) coordinate of a non-zero tuple.

```

1: function SPARSEMATVECTMULT( $\llbracket X \rrbracket$ ,  $\llbracket y \rrbracket$ )
2:   Let  $\llbracket z \rrbracket \leftarrow \llbracket X \rrbracket$ 
3:   for ( $\llbracket j \rrbracket$ ,  $\llbracket v \rrbracket$ ) in  $\llbracket y \rrbracket$  do
4:     Append tuple (coord : ( $\llbracket \perp \rrbracket$ ,  $\llbracket j \rrbracket$ ), val :  $\llbracket v \rrbracket$ ) to  $\llbracket z \rrbracket$ 
5:   Let  $\llbracket z \rrbracket \leftarrow \llbracket \text{sort}(z) \rrbracket \triangleright$  Asc. sort on the 2nd and then 1st coordinates (with
       $\perp < 1$ )

6:   Let prevY  $\leftarrow$  (coord : ( $\llbracket \perp \rrbracket$ ,  $\llbracket \perp \rrbracket$ ), val :  $\llbracket \perp \rrbracket$ )
7:   for  $k \leftarrow 1$  to ( $\text{nnz}(X) + \text{nnz}(y)$ ) do
8:     obliv. if  $\llbracket \text{ColCoord}(\text{prevY}) \rrbracket = \llbracket \text{ColCoord}(z_k) \rrbracket$ 
9:       Let  $\llbracket \text{val}(z_k) \rrbracket \leftarrow \llbracket \text{val}(z_k) \rrbracket \times \llbracket \text{val}(\text{prevY}) \rrbracket$ 
10:    else
11:      Let  $\llbracket \text{val}(z_k) \rrbracket \leftarrow \llbracket 0 \rrbracket$ 
12:    obliv. if  $\llbracket \text{RowCoord}(z_k) \rrbracket = \llbracket \perp \rrbracket$ 
13:      Let  $\llbracket \text{prevY} \rrbracket \leftarrow \llbracket z_k \rrbracket$ 
14:      Let  $\llbracket \text{val}(z_k) \rrbracket \leftarrow \llbracket \perp \rrbracket$ 

15:   Drop the second coordinate from all tuples in  $\llbracket z \rrbracket$ 
16:   Let  $\llbracket z \rrbracket \leftarrow \llbracket \text{sort}(z) \rrbracket \triangleright$  Asc. sort on the remaining coord.
17:   AGGEQUALCOORD( $\llbracket z \rrbracket$ )
18:   PLACEHOLDERREMOVAL( $\llbracket z \rrbracket$ )
19:   return  $\llbracket z \rrbracket \triangleright$  sparse vector stored as a list of non-zeros

```

On the one hand, honest-majority dense multiplication is no longer constant in communications, so our algorithm is advantageous for highly sparse matrices. On the other hand, the memory footprint is more critical in this operation. Several real-world sparse matrices do not fit in memory using a dense format (see example application in Section 8.3), even in plaintext! **Our sparse algorithm is then mandatory when memory overflowing prevents from using dense algorithms.**

8.2.4 Matrix-matrix

We want to multiply two sparse matrices. Such multiplications are used in ML algorithms for specific computations, especially correlation matrices (i.e., $X^\top X$). Other ML algorithms also require sparse-dense matrix multiplication (e.g., to multiply a sparse input matrix with a dense parameter matrix). Even in plaintext, sparse-dense multiplications require specific algorithms differing from those of

Algorithm 7 Aggregation of tuples with equal coordinates**Input:** $\llbracket Z \rrbracket$, a sorted list of secret shared tuples

```

1: procedure AGGEQUALCOORD( $\llbracket Z \rrbracket$ )
2:   for  $k \leftarrow 1$  to  $\text{length}(\llbracket Z \rrbracket) - 1$  do
3:     obliv. if  $\llbracket \text{coord}(Z_k) \rrbracket = \llbracket \text{coord}(Z_{k+1}) \rrbracket$ 
4:       Let  $\llbracket \text{coord}(Z_k) \rrbracket \leftarrow \llbracket \perp \rrbracket$ 
5:       Let  $\llbracket \text{val}(Z_{k+1}) \rrbracket \leftarrow \llbracket \text{val}(Z_{k+1}) \rrbracket + \llbracket \text{val}(Z_k) \rrbracket$ 

```

Algorithm 8 Placeholder removal**Input:** $\llbracket Z \rrbracket$, a list of secret shared tuples

```

1: procedure PLACEHOLDERREMOVAL( $\llbracket Z \rrbracket$ )
2:   Let  $\llbracket Z \rrbracket \leftarrow \llbracket \text{shuffle}(Z) \rrbracket$ 
3:   for  $k \leftarrow 1$  to  $\text{length}(\llbracket Z \rrbracket)$  do
4:     Let  $\text{is\_placeholder} \leftarrow \text{REVEAL}(\llbracket \text{coord}(Z_k) \rrbracket = \perp)$ 
5:     if  $\text{is\_placeholder}$ 
6:       Remove the  $k$ -th tuple from  $\llbracket Z \rrbracket$ 

```

sparse-sparse multiplications (i.e., the focus of this chapter). Thus, we leave sparse-dense multiplications for future works. This subsection then emphasizes the computation $X^\top X$, which is a key sparse-sparse operation in ML.

To compute $X^\top X$, we multiply an $m \times n$ sparse matrix to an $n \times m$ matrix. With our leakage, we know the per-column sparsity of the first matrix (because X is transposed), and per-row sparsity of the second. Algorithm 9 generalizes this problem: multiplying an $n \times m$ sparse matrix X to an $m \times p$ sparse matrix Y , with per-column sparsity knowledge for X and a per-row knowledge for Y . We use the notation $X^{(k)}$ (resp. $Y^{\{k\}}$) to refer to the k -th column (resp. row) of X (resp. Y) stored as a sparse vector.

When implementing a matrix-matrix multiplication naively, it is common to design a triple nested loop multiplying: each row i of X is multiplied to each column j of Y . The third loop is the vector multiplication on each k -th element of the vectors. Linear algebra libraries usually leverage a simple optimization to speed up the dense matrix multiplication: swapping the loops on i and k . This optimization significantly speeds up the implementation by reducing the memory access costs [Lei18].

Our sparse algorithm reuses a similar intuition to design a fast algorithm and avoid performing a series of vector multiplications. Our algorithms have two main steps: (1) compute all the individual scalar multiplications, (2) aggregate them using oblivious sorting. In matrix-matrix multiplications, all values in the k -th column of X must be multiplied by all values in the k -th row of Y . Our leakage provides direct access to the k -th column of X and the k -th row of Y , so we can compute the individual scalar multiplications by iterating on the non-zeros of $X^{(k)}$

Majority	Algorithm	Comm.	Comp.	Storage
Dishonest	Dense [Naive alg.], [CKR ⁺ 20, MZ17, MG23]	$O(nm)$		$O(nm)$
	Sparse (ours : Algo- rithm 6)	$O(\text{nnz}_* \cdot \log(\text{nnz}_*))$		$O(\text{nnz}_*)$
Honest	Dense [Hoo12, KPPS21] [LZLR23, MR18, WTB ⁺ 21]	$O(n)$	$O(nm)$	$O(nm)$
	Sparse (ours : Algo- rithm 6)	$O(\text{nnz}_* \cdot \log(\text{nnz}_*))$		$O(\text{nnz}_*)$

Table 8.1 / Complexities of the sparse and dense matrix-vector multiplications

Notations: $\text{nnz}_* = \text{nnz}(X) + \text{nnz}(y)$, $\text{nnz}(X) \ll n \times m$, $\text{nnz}(y) \ll m$

Majority	Algorithm	Comm.	Comp.	Storage
Honest	Dense [CKR ⁺ 20, MZ17, MG23]	$O(nm + mp + np)$	$O(nmp)$	$O(nm + mp + np)$
	Sparse (ours : Algo- rithm 9)	$O(\text{MinMult} \cdot \log(\text{MinMult}))$		$O(\text{MinMult})$
Dishonest	Dense [Hoo12, LZLR23] [MR18, WTB ⁺ 21]	$O(np)$	$O(nmp)$	$O(nm + mp + np)$
	Sparse (ours : Algo- rithm 9)	$O(\text{MinMult} \cdot \log(\text{MinMult}))$		$O(\text{MinMult})$

Table 8.2 / Complexities of the sparse and dense matrix-matrix multiplications

Notations: $\text{MinMult} = \sum_{k=0}^m \text{nnz}(X^{(k)}) \cdot \text{nnz}(Y^{\{k\}})$, $\text{nnz}(X) \ll n \times m$, $\text{nnz}(Y) \ll m \times p$

and $Y^{\{k\}}$. Algorithm 9 reuses Algorithm 7 and Algorithm 8 for aggregation and placeholder removal.

Table 8.2 compares our sparse algorithm to the dense algorithms. The notation $\text{MinMult} = \sum_k \text{nnz}(X^{(k)}) \cdot \text{nnz}(Y^{\{k\}})$ denotes the number of non-zero scalar multiplications required by the matrix multiplication. It represents an intuitive lower bound: the best plaintext algorithm is $O(\text{MinMult})$, with no optimality proof. Our algorithm *only adds a logarithmic factor* compared to the plaintext sparse algorithm [BGS11].

8.2.5 Security proof sketch

This subsection sketches the security proof of our algorithms. We provide proof sketches in the honest-but-curious model using the real-ideal paradigm [EKR18].

We can define the ideal functionality for the sparse vector multiplication $\mathcal{F}_{\text{VectMult}}^\eta$ as follows:

- Input: $\llbracket x_a \rrbracket = \{(\llbracket i_{1,a} \rrbracket, \llbracket v_{1,a} \rrbracket), \dots, (\llbracket i_{\eta,a} \rrbracket, \llbracket v_{\eta,a} \rrbracket)\}$ and $\llbracket x_b \rrbracket$ (defined similarly)
- Output: a secret-shared scalar $\llbracket s \rrbracket$ such that: $s = \sum_{j,k \in \{1 \dots \eta\}} v_{j,a} \cdot v_{k,b} \cdot \delta(i_{j,a}, i_{k,b})$ (with δ the Kroenecker delta).

Algorithm 9 Secure sparse matrix multiplication**Input:** $\llbracket X \rrbracket$ and $\llbracket Y \rrbracket$ are two secret-shared sparse matrices**Leakage:** Number of non-zeros per column in X (with $X^{(k)}$ a column), number of non-zeros per row in Y (with $Y^{\{k\}}$ a row)

```

1: function SPARSEMATMULT( $\llbracket X \rrbracket, \llbracket Y \rrbracket$ )
2:   Let  $\llbracket Z \rrbracket \leftarrow \emptyset$ 
3:   for  $k \leftarrow 1$  to  $m$  do
4:     for  $i \leftarrow 1$  to  $\text{length}(\llbracket X^{(k)} \rrbracket)$  do
5:       for  $j \leftarrow 1$  to  $\text{length}(\llbracket Y^{\{k\}} \rrbracket)$  do
6:         Let  $\llbracket r_i \rrbracket \leftarrow \llbracket \text{coord}(X_i^{(k)}) \rrbracket$   $\triangleright$  Row coordinate
7:         Let  $\llbracket r_j \rrbracket \leftarrow \llbracket \text{coord}(Y_j^{\{k\}}) \rrbracket$   $\triangleright$  Column coordinate
8:         Let  $\llbracket r_v \rrbracket \leftarrow \llbracket \text{val}(X_i^{(k)}) \rrbracket \times \llbracket \text{val}(Y_j^{\{k\}}) \rrbracket$ 
9:         Append a tuple  $((\llbracket r_i \rrbracket, \llbracket r_j \rrbracket), \llbracket r_v \rrbracket)$  to  $\llbracket Z \rrbracket$ 
10:        // Non-zero tuple: coordinates then value

11:   Let  $\llbracket Z \rrbracket \leftarrow \llbracket \text{sort}(Z) \rrbracket$   $\triangleright$  Asc. sort on the coordinates.
12:   AGGEQUALCOORD( $\llbracket Z \rrbracket$ )
13:   PLACEHOLDERREMOVAL( $\llbracket Z \rrbracket$ )
14:   return  $\llbracket Z \rrbracket$   $\triangleright$  sparse matrix stored as a list of non-zeros

```

This functionality has a property η . This property represents the protocol leakage in our functionality. As motivated in Chapter 4, leakage is necessary to have efficient secure sparse multiplications.

We sketch a proof showing that Algorithm 5 securely realizes the ideal functionality $\mathcal{F}_{\text{VectMult}}^\eta$. Our proof relies essentially on the composition theorem for the honest-but-curious model (Theorem 7.3.3 of [Go109]). Certain subprotocols we use have been proven to be secure and correct in the honest-but-curious model. By using the composition theorem, we can rely on their security and only prove the security of our overall protocol in which these subprotocols are embedded.

The correctness of Algorithm 5 can be verified by an easy arithmetic exercise.

Security of the sub-protocols The first statement in Algorithm 5 is a concatenation that is performed locally with no communication and hence does not need to be simulated. The second statement is an oblivious sort. The proof depends on the sub-protocol chosen for instantiation. This choice is highly influenced by the threat model, so we consider the Batcher's sort for dishonest majority and the oblivious radix sort for honest majority (i.e., the two threat models studied in this chapter). On the one hand, the Batcher's sort only requires comparisons and arithmetic operations. It is then secure because the arithmetic and comparison operations on secret-shared values are trivially secure in the honest-but-curious model [Hoo12]. On the other hand, the oblivious radix sort is proven secure in [HICT14]. After the sorting statement, we have a loop. This loop statement reveals no information because it uses η , a property of $\mathcal{F}_{\text{VectMult}}^\eta$ (i.e., a public

value). Finally, the loop contains a conditional statement. As highlighted in Section 8.2, we use the notation $\llbracket \text{obliv. if} \rrbracket$ to maximize the readability. This conditional statement would be implemented as follows: $\llbracket s \rrbracket \leftarrow \llbracket s \rrbracket + (\llbracket \text{coord}(z_i) \rrbracket = \llbracket \text{coord}(z_{i+1}) \rrbracket) \times \llbracket \text{val}(z_i) \rrbracket \times \llbracket \text{val}(z_{i+1}) \rrbracket$, with $\llbracket \text{coord}(z_i) \rrbracket = \llbracket \text{coord}(z_{i+1}) \rrbracket$ outputting $\llbracket 1 \rrbracket$ or $\llbracket 0 \rrbracket$. The arithmetic operations are trivially secure in the honest-but-curious model. For the secure equality, we can use the secure protocol proposed by Toft [Tof07]. Hence, this oblivious conditional statement is secure because it relies on secure comparisons and arithmetic operations. Note that such oblivious conditional statements are recurrent in MPC protocols (e.g., in private heavy hitters [AHI⁺22]).

Security of the overall protocol Since all the sub-protocols are individually secure, according to the Composition Theorem, Algorithm 5 preserves the input secrecy in the honest-but-curious model.

Matrix-vector and matrix-matrix We do not develop the proof sketch for our sparse matrix-vector (Algorithm 6) and matrix-matrix (Algorithm 9) multiplications, but the proof is very similar. As our sparse vector multiplication, these two algorithms use oblivious sorting, comparisons, arithmetic operations, and oblivious conditional structures. For all these operations, the security would be proven as sketched for the vector multiplication. However, they also use two additional sub-protocols: AggEqualCoord and PlaceholderRemoval.

The AggEqualCoord relies on an oblivious conditional structure and arithmetic operations. The proof is then trivial, because we can express the conditional structure using a combination of comparisons and arithmetic operators.

The PlaceholderRemoval uses a “shuffle-and-reveal” trick: shuffle a list of secret-shared values and reveal which of them are placeholders. Other security papers [HICT14] used similar “shuffle-and-reveal” tricks in protocols with proven security. In our algorithms, revealing the number of placeholders indirectly reveals the number of non-zero elements in the output matrix. However, this information is public by definition because it is contained in the output.

Extension to malicious security Our security proofs hold in the honest-but-curious model. Malicious security would require a verifiable secret-sharing scheme and dedicated proofs for the security in the malicious model. We remind that our main sub-protocols have known maliciously secure variant (especially oblivious sorting [AHI⁺22, HICT14] and oblivious shuffling [AHI⁺22, LWZ11]) that would be the basis of a malicious security proof. We leave this extension for future work.

8.3 Experimental analysis

This section compares the dense algorithms to our sparse algorithms under three sparsity levels: 99%, 99.9%, 99.99% of zero values. These thresholds are recurrent in sparse data applications (see real-world datasets mentioned in Section 3.1). The

dense algorithm is, by definition, insensitive to the sparsity level, so we provide one curve for the dense algorithm and one per sparsity level for our sparse algorithm.

Our comparison focuses on communication costs and memory usage. The communication costs corresponds to the total communication exchanged between the servers during the protocol. The memory usage is represented thanks to the memory overflows: we scale up our experiments until the algorithms cannot be executed due to a lack of memory. We use red crosses to represent these memory overflows on the plots.

For each multiplication type (i.e., matrix-vector and matrix-matrix), we also implement an example use case built upon it. Each example use case relies on a real-world dataset to demonstrate the impracticality of dense algorithms on simple ML applications involving sparse data.

8.3.1 Experimental setup

We execute our experiments on a server with 188GB of RAM and an Intel Xeon CPU. We use the MPyC framework [Sch18] to simulate 3-party protocols.

We perform our experiments under an honest-majority assumption because this setting provides the **best dense baselines** (see Tables 8.1 and 8.2). More specifically, we chose the *SSS-based matrix multiplication* [Hoo12] as dense baseline instead of the other honest-majority protocols [KPPS21, LZLR23, MR18, WTB⁺21] for two reasons. First, the SSS-based matrix multiplication has smaller constant factors; e.g., it only requires the communication of a single ring element, while [KPPS21, LZLR23, MR18, WTB⁺21] require multiple communication rounds. Second, the SSS-based protocol supports an arbitrary number of parties (like our sparse multiplications), while [KPPS21, LZLR23, MR18, WTB⁺21] are 3-party protocols.

Implementation details We use 64-bit (secret-shared) fixed-point numbers, with a 32-bit fractional part. Floating-point arithmetic is still poorly scalable in MPC protocols. MPC works usually rely on fixed-point arithmetic [KRS22] to avoid performance bottlenecks. The minor precision loss due to fixed-point arithmetic is often acceptable for ML applications [KRS22].

We use the protocol by [LWZ11] for oblivious shuffling and the oblivious radix sort [HICT14] for sorting. The first step of the oblivious radix sort is a secure bit decomposition. As this operation is expensive, we accelerate the process using pre-computations. Our algorithms assume the data owners pre-compute and share a bit decomposition of their non-zero coordinates. Hence, they share a bit array instead of a single integer per non-zero coordinate.

8.3.2 Matrix-vector

Fig. 8.3a compares the dense and sparse matrix-vector multiplications for a square matrix. Note that our algorithm with 99.99% sparsity has a negligible cost when

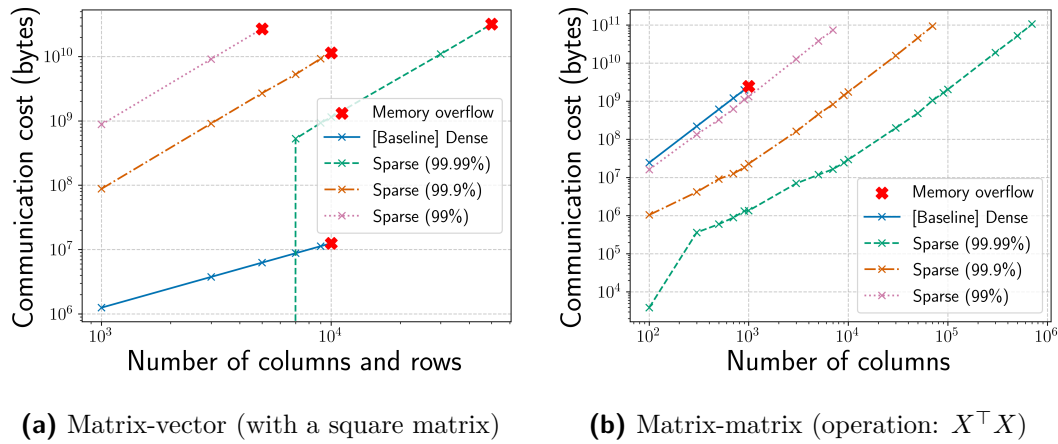


Figure 8.3 / Comparison of the secure matrix multiplications under honest majority: dense algorithm vs. *our sparse algorithm* with three sparsity levels

the number of rows is small because the high sparsity leads to empty sparse vectors. Overall, the dense algorithm is less expensive than our algorithm.

The main takeaway is the memory issue. Indeed, while the dense algorithm is less expensive, our algorithm supports larger matrices (i.e., the first motivation of sparse algorithms). For example, the dense algorithm (contrary to our sparse algorithm with 99.99% sparsity) triggers memory overflows for matrices with more than 10K columns. As developed in Section 3.1, there is no (dense) solution to avoid memory issues (19TB of memory would be needed).

One may argue that our algorithm also has some memory overflows. To analyze these results, we must remind the results from Section 3.1: larger datasets become sparser. Understanding this phenomenon is essential to interpret Fig. 8.3a: the memory overflows illustrated in this figure should not impact most real-world datasets because of their correlation between sparsity and matrix sizes.

Our figures represent the costs for fixed sparsity rates, but large matrices are unlikely to have a low sparsity rate. For example, real-world datasets with 10K columns [LLW17] can have much more than 99% sparsity; e.g., with 10K columns, Yahoo Movie dataset already has a 99.7% sparsity! Thus, using real-world sparse data, our algorithm should not be affected by such memory issues, because larger datasets are sparser.

More generally, any algorithm (even sparse) has memory issues when the data is large enough. Sparse algorithms simply support larger sparse matrices than dense algorithms. In practice, the correlation between sparsity and matrix size even amplifies the gains provided by sparse algorithms.

Example use case: Recommender system Matrix-vector is a core component of some recommender systems. This example use case builds a simple recommender system in which a client sends a (secret-shared) item identifier and receives a list of similar items. We use the **Bookcrossing dataset**: contains ratings from

279K users on 340K books, resulting in 99.998% zeros in the dataset (1K users as training dataset). Each row corresponds to a user, and each column to a book.

Our recommender system relies on a **nearest neighbors** algorithm [Agg16]. This model has no training step: the inference is computed on the training dataset, and requires two matrix-vector multiplications. This algorithm computes the similarity between the book the client chose and all the other books in the database. Then, the system returns the identifiers of the k most similar books.

The dense algorithm triggers a memory overflow because it cannot store the training data using a dense matrix. The sparse algorithm takes, in average, 48 minutes. **The dense algorithm is impractical while our sparse algorithms support this application.** Our runtime could be further reduced using MPC optimizations (e.g., precomputed comparisons [SM25]).

8.3.3 Matrix-matrix

Fig. 8.3b shows the communication costs for the multiplication $X^T X$. This experiment considers a varying number of columns and a fixed number of rows (100 rows). The sparse algorithm brings a significant communication reductions compared to the dense multiplication: $\times 100$ gain for 99.9% sparsity and a $\times 1000$ gain for 99.99% sparsity. The cost reduction is much higher for matrix-matrix multiplication than for matrix-vector because the matrix-matrix multiplication complexity has a quadratic dependency on sparsity.

Besides the performance gain, **our algorithm supports much larger matrices**: the dense algorithm triggers a memory overflow for matrices with more than 1K, while our sparse algorithm scales up to 1M columns. Fig. 8.3b shows no memory overflow for our algorithm because we stopped experiments exceeding 10 hours.

Example use case: Access control This multiplication can be used to build an ML-based access control system. This example use case trains an ML model deciding whether an access request is suspicious. Access data can be particularly privacy-sensitive, e.g., the access log of medical databases can indirectly leak patient information. As medical access logs are sensitive, there is no publicly available dataset, so we use the **Amazon access control dataset**¹; that should have similar properties as more sensitive datasets.

The Amazon dataset contains 32.7K samples, each containing the user's properties, the service properties, and the access decision (granted or not). Each property is a categorical value that we transform into binary arrays using the one-hot encoding. This encoded dataset has 15K features with 99.95% sparsity.

Our access control relies on **Linear Discriminant Analysis**. The training requires computing: (1) the proportion of each class (i.e., access decision), (2) the

¹<https://github.com/pyduan/amazonaccess/tree/master/data>

mean vector of each class, and (3) the covariance matrix of the whole dataset. The most expensive operation is, by far, the covariance matrix estimation.

Note that the covariance matrix is necessarily dense. As $Cov(X) = \frac{1}{n} X^\top X - \bar{x}^\top \bar{x}$, we store the covariance in two parts: the sparse product $X^\top X$ and the dense mean vector $\bar{x}$. This trick provides an efficient storage of the covariance matrix. This covariance matrix is used at inference time via a matrix-vector multiplication (e.g., $Cov(X) \cdot y = \frac{1}{n} X^\top X \cdot y - \bar{x}^\top \bar{x} \cdot y$). Thus, this compact representation does not change the model inference algorithm and avoids a potential memory overflow due to the high-dimensional dense matrix.

We implement this application using the dense and sparse matrix multiplications. We use the first 10K samples as training dataset. As expected from our previous benchmark, the dense algorithm triggers a memory overflow during the covariance computation. The sparse algorithm completes the training algorithm in 5 hours. Hence, *our sparse algorithm provides a satisfying solution* to this problem, while the **dense algorithm is impractical**.

8.4 Mitigating the leakage

Chapter 4 highlighted the need for sparsity-related leakage in secure sparse algorithms. While existing works [CCL⁺21,SGRP19] do not discuss the leakage, this section presents three techniques to mitigate it.

8.4.1 Row anonymization

By default, our algorithms assume the per-row sparsity to be public. This information means that the number of non-zeros from each data owner is public (but the position and value of the non-zeros remain secret). To enhance their privacy, the data owner can submit their shares via an anonymization layer (e.g., Tor network). Machine learning algorithms typically assume that training instances are independently and identically drawn and don't extract information from the order of the instances (rows). Permuting the rows of the input matrix hence does not substantially change the input.

Thanks to the anonymization layer, the servers would be unable to link a data owner to a number of non-zeros. This privacy enhancement also reduces the leakage: instead of the per-row sparsity, the distribution of the per-row sparsity is public.

This difference is subtle, but it is important for privacy: instead of individual information, a “collective” information is public. Indeed, this leakage provides information about the whole population, but not about an individual.

ML practitioners usually need this sparsity distribution information, before performing any ML training. Sparsity is essential information to choose an appropriate

ML model. Thus, it is realistic to assume that the distribution of the per-row sparsity is public.

8.4.2 Max-row padding

As motivated above, the per-row sparsity distribution is acceptable in practical ML applications. However, if needed, we can further reduce the leakage and only reveal an upper bound on the per-row sparsity.

To only assume this upper bound as leakage, the input sparse matrix must be padded to this maximum per-row sparsity. To pad a row, the data owner represents some zero elements as non-zeros anyway (i.e., creating dummy non-zeros), such that all rows have the pre-specified number of non-zeros. Computing the maximum per-row sparsity over all rows in MPC requires $O(n \log n)$ communications (in $O(\log n)$ rounds) for a matrix of n rows.

8.4.3 Matrix templating

The main drawback of max-row padding is its sensitivity to the maximum number of non-zeros per row. Sparse datasets usually contain only a few rows with many non-zeros. Most rows contain much fewer non-zeros than the maximum. For example, Fig. 8.4 shows the per-row sparsity distribution on the Bookcrossing dataset. While the maximum number of non-zeros per row is nearly 10^4 , 99% of the rows have less than 100 non-zeros and 90% of the rows have less than 10! Thus, padding the rows until a global maximum number of non-zeros would add many unnecessary dummy non-zeros to the dataset.

Up until now, all our techniques and algorithms were oblivious to the number of rows owned by each data owner: they support all possible settings from data owners owning a single row to settings with data owners owning a sub-matrix. Note that the input size (i.e., number of rows) of each data owner is usually public in MPC. If each data owner owns a sub-matrix, we can propose a more subtle padding technique avoiding the shortcomings of max-row padding.

Instead of padding each row to a fixed maximum, we propose to build a “matrix template”, a safe upper bound for the distribution of number of non-zeros per row, for which we are sure the real data matrix will fit in it. Let us call $\hat{F}(d) = \sum_{i=d}^{\infty} \hat{f}(i)$ the number of rows in the real dataset having d non-zeros or more. Let us describe what we call a template for a matrix Y : the template divides Y (horizontally) into K sub-matrices: $Y_1, \dots, Y_K$, with respectively $n_1, \dots, n_K$ rows. Each row in Y_k must have $\overline{\text{nnz}}_k$ non-zero elements, and we have $\overline{\text{nnz}}_1 \leq \overline{\text{nnz}}_2 \leq \dots \leq \overline{\text{nnz}}_K$. A matrix template is then defined by $T = \{(n_1, \overline{\text{nnz}}_1), \dots, (n_K, \overline{\text{nnz}}_K)\}$. A real dataset will fit in the template (after a permutation of the rows, see Section 8.4.2) if $\forall i \in [K] : \hat{F}(\overline{\text{nnz}}_i + 1) \leq \sum_{j=i+1}^K n_j$.

A data owner P owning a dataset Y can fit Y into a template T by padding Y according to T . First, they sort (increasingly) the rows of Y based on their

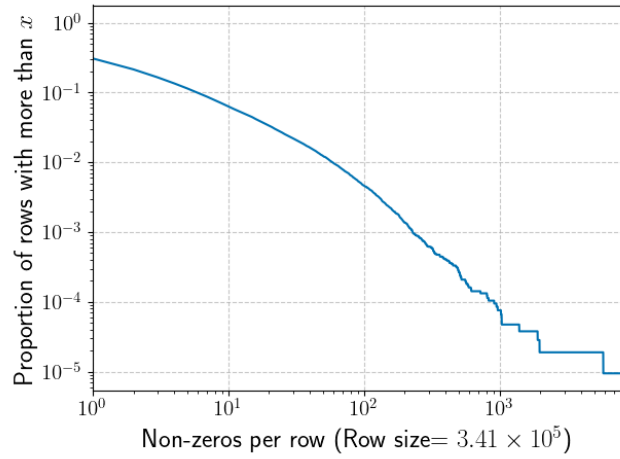


Figure 8.4 / Per-row sparsity distribution in Bookcrossing [ZMKL05]

number of non-zeros. Remember that the order of the rows does not matter in ML applications. Second, they pad the first n_1 rows of Y to have $\overline{\text{nnz}}_1$ non-zeros. They repeat this padding process for the next n_2 rows with $\overline{\text{nnz}}_2$ non-zeros, etc. until the last n_K rows with $\overline{\text{nnz}}_K$ non-zeros. Thanks to the sub-matrices, only Y_K is padded to have the maximum number of non-zeros, the rest of the rows can have much fewer non-zeros. Note that matrix templating with $K = 1$ has the same output as max-row padding. Similarly, the higher K is, the smaller the overhead because the matrix template becomes more accurate.

We envision four privacy-preserving approaches to obtain such matrix templates. First, one may know the population distribution, i.e., the probability $f(d)$ of drawing a row with d non-zeros in the process generating the data. Second, without such domain knowledge one may have seen a public dataset D from which one could draw statistics and approximate the population distribution by inferring $\hat{F}_D(d) \geq F(d)$, before one has to process the sensitive data. Third, one could start from the real data provided as input and use differential privacy to release safe upper bounds $\hat{F}_{DP}(d) \geq \hat{F}(d)$. Fourth, we can compute it using MPC-based quantile estimation. **Using matrix templating, the leakage is then reduced to a few quantiles** from the per-row sparsity distribution.

For details about the last three approaches, we refer the reader to the appendix of this chapter [DHPR25d].

8.4.4 Overhead comparison

Row padding and matrix templating induce some overhead due to the dummy non-zeros they add. Fig. 8.5 compares the memory footprint of several alternatives: dense multiplication, no knowledge minimization, row anonymization, max-row padding, and matrix templating (using quantiles as in Section 8.4.3). The memory footprint is a simple proxy to understand the impact of knowledge minimization techniques on our algorithms' related communication and computation costs

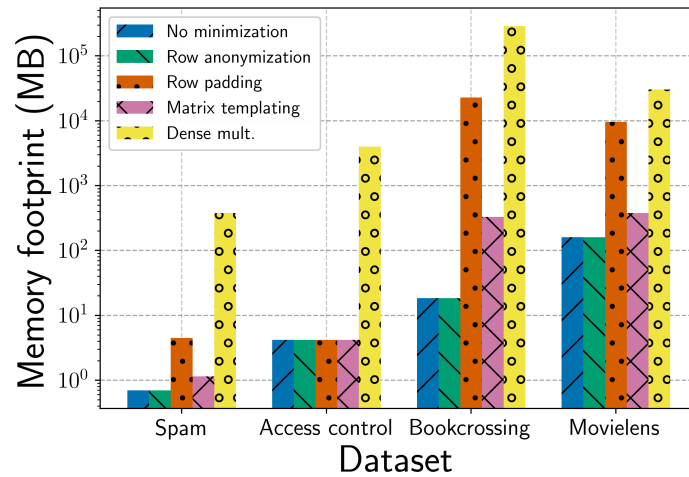


Figure 8.5 / Matrix storage costs for different techniques of leakage minimization on four real-world datasets

(see also our tables in Section 8.2). We base the comparison on four real-world datasets: SMS Spam [GBSG06], Amazon Access Control, Bookcrossing [ZMKL05], and MovieLens [HK16]. For the matrix templating, our experiments simulate the MPC-based estimation with 20 data owners.

On the one hand, (as expected) row anonymization has no effect on the memory footprint. On the other hand, padding rows to a global maximum induces a small overhead on the Spam and the Access control datasets, while it induces significant overhead on the Bookcrossing and MovieLens datasets. The overhead on Bookcrossing and MovieLens makes the sparse matrix nearly as large as the dense matrix... in such case, the sparse algorithms would have no interest.

Our matrix templating significantly reduces this overhead. For example, on MovieLens, adding dummy non-zeros via matrix templating only doubles the memory footprint (vs. nearly $\times 100$ for max-row padding). As mentioned earlier, the overhead could be even further reduced if we reveal more than 5 quantiles.

Finally, on the Access Control dataset, all knowledge minimization techniques have an equal memory footprint. In this dataset, all rows have *by design* the same number of non-zeros (i.e., naturally padded), so max-row padding and matrix templating does not need to add any dummy non-zero.

Conclusion

This chapter introduced algorithms based on oblivious sorting to multiply secret-shared sparse matrices. Our algorithms are **compatible with a more generic MPC setup** (i.e., outsourced computations) than existing works on secure sparse matrix multiplications. They **avoid memory issues** present in dense secure multiplications by leveraging the data sparsity. Our experiments show

communication cost reductions up to $\times 1000$ compared to dense matrix multiplications. We also implemented two real-world ML applications highlighting that **our sparse algorithms support applications impractical with existing protocols**. Finally, we proposed methods to minimize the leakage (mandatory in any secure sparse algorithm).

Our implementation is publicly available and open-source, so our algorithms can be transferred into MPC frameworks:

<https://github.com/MarcTOK/Secure-sparse-multiplications>

Part IV

Improving Function Secret Sharing schemes for sparse aggregation

This fourth part introduces new FSS schemes to support their applications to sparse data aggregation (detailed in Chapter 2). While existing two- and three-party FSS schemes are highly efficient, extending them to an arbitrary number of parties remains challenging. Some schemes, for example, require keys whose size grows exponentially with the number of parties. To support applications to sparse computations, we design new multi-party FSS schemes that overcome such bottlenecks.

All chapters in this part rely on the notations presented in Table 9.1.

9 Eliminating exponential key growth in PRG-Based Distributed Point Functions

While 2-party PRG-based schemes with logarithmic key sizes have been known for a decade, extending these solutions to multi-party settings has proven challenging. As a first step into multi-party FSS, this chapter optimizes the PRG-based DPF proposed by [BGI15]. This DPF based on the PRG assumption has an exponential factor q^p in its key size, making it impractical for many applications notably to statistics and ML. This chapter addresses this efficiency bottleneck by eliminating the exponential factor present in the key size of [BGI15].

Our construction is the first PRG-based multi-party DPF scheme with practical key sizes, and provides key up to $3\times$ smaller than the best known DPF. Our work demonstrates that with careful optimization, PRG-based multi-party DPFs can indeed achieve practical performances.

This chapter is drawn from [DHPR25b].

9.1 Introduction

As explained in Section 2.2.3, statistics applications (such as sparse aggregation) require efficient schemes with output in prime fields $\mathbb{F}_q$. Indeed, shares must be in prime fields so that shareholders can aggregate them locally. It is naturally possible to encode and share any l -bit integers using bit fields $((\mathbb{F}_2)^l)$, but the addition of two shared values would require communications between shareholders.

Unfortunately, the existing PRG-based multi-party DPF proposed by [BGI15] has an exponential factor q^p present in their key size. This scheme was initially designed for applications like PIR in which bit field outputs (i.e., $q = 2$) is acceptable. However, as statistics applications require larger moduli, we need to eliminate the factor q^p to have a scheme compatible notably with sparse aggregation.

Our contribution Our optimization relies on the honest-majority assumption to obtain a key size of $O(\sqrt{N \cdot \binom{p}{m+1}} \log q)$ instead of $O(\sqrt{N} q^{\frac{p-1}{2}} \log q)$ in [BGI15]. In other words, we tolerate fewer corrupted parties than [BGI15] in order to eliminate the exponential factor. Our benchmark shows that our scheme is the first multiparty PRG-based scheme with practical key sizes. We even provide keys up to $3\times$ smaller than the best performing DPF (i.e., the IT DPF by [BK022]).

Notation	Meaning
General notations	
$\mathcal{E}$	Tuple $\mathcal{E} = (\text{Gen}, \text{Eval}, \text{Decode})$ defining an FSS scheme
p	Number of parties/shareholders
m	Number of corrupted parties
1^λ	Security parameter
$\mathbb{F}_q$	Prime field
$\mathbb{G}$	Generic Abelian group
$\llbracket x \rrbracket_i$	i -th share of the secret x
$s \xleftarrow{R} S$	Uniform sampling of an element s from the set S
Function specific notations	
N	Function domain size
$\mathcal{X}$	Function domain
$\mathcal{Y}$	Function co-domain
$\mathcal{F}$	Function family
Derived notations	
ν	$= \lceil \sqrt{N} \rceil$
C	$= \binom{p}{m+1}$

Table 9.1 / Notations for the chapters on FSS

9.2 Honest-majority PRG-based scheme

9.2.1 PRG-based scheme by [BGI15]

To build their multi-party DPF scheme, [BGI15] represented the function domain $\{1, \dots, N\}$ as a grid of size $\nu \times \nu$ (with $\nu = \lceil \sqrt{N} \rceil$). Like [BGI15], we use assume the existence of a PRG $G : \{0, 1\}^\lambda \rightarrow \mathbb{G}^\nu$.

This grid is full of zeros except on the cell (γ_*, δ_*) , with $\alpha = \gamma_*\nu + \delta_*$. For each row $\gamma \in \{1, \dots, \nu\}$, the Gen algorithm samples q^{p-1} random λ -bit random seeds $s_{\gamma,1} \dots s_{\gamma,q^{p-1}}$. For each seed $s_{\gamma,j}$, the algorithm generates additive shares of a coefficient $a_{\gamma,j}$: $\llbracket a_{\gamma,j} \rrbracket_1, \dots, \llbracket a_{\gamma,j} \rrbracket_p$ (i.e., one share per DPF key). The coefficient is defined as follows $a_{\gamma,j} = 1$ if $\gamma = \gamma_*$, 0 otherwise. Finally, it sets a “correction word” $W \in (\mathbb{F}_q)^\nu$ such that $W + \sum_{i=1}^{q^{p-1}} G(s_{\gamma_*,j}) = e_{\delta_*} \cdot \beta$ (with e_δ a unit vector equal to 1 on index δ , 0 otherwise). Each key k_i contains the correction word, their share of the coefficients $\llbracket a_{\gamma,j} \rrbracket_i$ (for all rows γ and all $j \in \{1, \dots, q^{p-1}\}$), and it contains all the seeds $s_{\gamma,j}$ for which $\llbracket a_{\gamma,j} \rrbracket_i \neq 0$. This last condition (on the seed inclusion in a key) ensures that there is at least one seed unknown to an adversary composed of $p - 1$ out of p parties.

The Eval algorithm represents the input x as a tuple (γ, δ) , expands the corresponding seeds $s_{\gamma,j}$, multiplies the expanded seeds with the corresponding shared coefficient $\llbracket a_{\gamma,j} \rrbracket_i$, sums everything with the correction word W , and the share

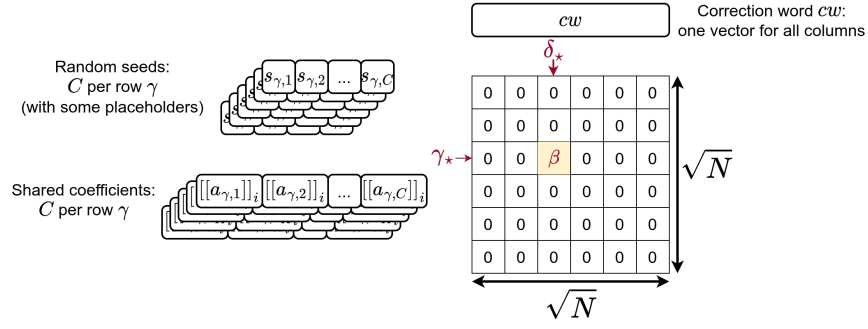


Figure 9.2 / Structure of our DPF keys

$\llbracket f(x) \rrbracket_i$ is on the δ -th index of the sum vector:

$$\llbracket f(x) \rrbracket_i = v[\delta] \text{ with } v = W + \sum_j \llbracket a_{\gamma,j} \rrbracket_i \cdot G(s_{\gamma,j})$$

An element could surprise: the number of random seeds $\nu \times q^{p-1}$. Such a large number is necessary, so an adversary cannot infer information about the secret function based on the distribution of the shares $\llbracket a_{\gamma,j} \rrbracket_i$. These shares can be structured as matrix shares A_γ with $A_\gamma[i, j] = \llbracket a_{\gamma,j} \rrbracket_i$. To preserve function privacy, each matrix must contain all combination of additive shares summing to 1 (if $\gamma = \gamma_*$) or 0 (if $\gamma \neq \gamma_*$). If the matrices do not contain all possible combinations, an adversary (owning up to $p - 1$ keys out of p) can recover γ_* based on the share distribution. Since there exists q^{p-1} combinations of p shares of 0 (resp. of 1) in $\mathbb{F}_q$, the key generator must sample q^{p-1} random seeds.

As we optimize [BGI15], Algorithm 10 (describing our scheme) follows roughly the same structure as their scheme. The *only difference* is the matrix sampling in Lines 8 and 9. We then refer to Algorithm 10 for more details.

9.2.2 Our honest-majority scheme

The main scalability bottleneck in [BGI15] lies in the size of the matrices of shares, we improve their scheme by eliminating its dependence on the field size; thanks to the honest-majority assumption ($m < p/2$). Specifically, we redesign the matrices to be smaller, resulting in more compact keys. Algorithm 10 presents our multi-party DPF scheme, and Fig. 9.2 an overview of our DPF keys.

Our matrix sampling (described in the function `MatrixOfShares` of Algorithm 10) generates a matrix A that distributes shares among all possible combinations of $m + 1$ parties out of p . For each combination S_j of $m + 1$ parties, the function samples $m + 1$ shares of the secret coefficient a (using additive secret sharing) and assigns each share to the cell of A ; $A[i, j] = \llbracket a \rrbracket_i$ if $i \in S_j$, 0 otherwise. The secret coefficient a is either set to 0 or 1 depending on whether the value x being evaluated matches α (the non-zero point).

While in [BGI15], each matrix of shares has p rows and q^{p-1} columns, our

Algorithm 10 Honest-majority DPF scheme adapted from [BGI15]

```

1: function MATRIXOFSHARES( $a$ )
2:   Initialize  $A$  an  $p \times C$  matrix with zeros and the counter  $k$  to 1.
3:   for each set of parties  $S_j$  in the set of all combinations of  $m + 1$  of  $p$  do
4:     Sample  $m + 1$  shares of the value  $a$ :  $\{\llbracket a \rrbracket_{1,i} \dots \llbracket a \rrbracket_{m+1,i}\}$ .
5:     for each  $i$  in  $S_j$  do  $A[i, j] \leftarrow \llbracket a \rrbracket_{k,i}$  and increment  $k$ .
   return  $A$ 

6: function Gen( $\alpha, \beta, p, m, 1^\lambda$ )
7:   Represent  $\alpha$  as a pair  $\alpha = (\gamma_*, \delta_*)$  with  $\gamma_*, \delta_* \in \{0 \dots \nu\}$ .
8:   Sample  $A_1, \dots, A_\nu$  s.t. for all  $\gamma \neq \gamma_*$ ,  $A_\gamma \leftarrow \text{MatrixOfShares}(0)$ .
9:   Sample  $A_{\gamma_*} \leftarrow \text{MatrixOfShares}(1)$ .
10:  Choose randomly and independently  $\nu \cdot C$  seeds  $s_{1,1}, \dots, s_{\nu,C} \in \{0, 1\}^\lambda$ .
11:  Set the correction words  $W \in \mathbb{G}^\nu$  s.t.  $W + \sum_{j=1}^C G(s_{\gamma_*,j}) = e_{\delta_*} \cdot \beta$ .
12:  for  $i \in \{1 \dots p\}, j \in \{1 \dots C\}, \gamma \in \{1 \dots \nu\}$  do
13:    if  $A_\gamma[i, j] \neq 0$   $\sigma_{i,\gamma,j} \leftarrow (s_{\gamma,j}, A_\gamma[i, j])$ .
14:    else  $\sigma_{i,\gamma,j} \leftarrow (0, 0)$ .  $\triangleright$  Receives no seed and no coefficient share
15:  Set  $\sigma_{i,\gamma} \leftarrow (\sigma_{i,\gamma,1} \parallel \dots \parallel \sigma_{i,\gamma,C})$  for all  $1 \leq i \leq p, 1 \leq \gamma \leq \nu$ .
16:  return  $(k_1, \dots, k_p)$  with  $k_i = (\sigma_{i,1} \parallel \dots \parallel \sigma_{i,\nu} \parallel W)$  for all  $1 \leq i \leq p$ .

17: function Eval( $k_i, x$ )
18:  Represent  $x$  as a pair  $x = (\gamma, \delta)$  with  $\gamma, \delta \in \{0 \dots \nu\}$ .
19:  Parse  $k_i = ((s_{1,1}, A_1[i, 1]) \parallel \dots \parallel (s_{1,C}, A_1[i, C]) \parallel \dots \parallel (s_{\nu,C}, A_\nu[i, C]) \parallel W)$ .
20:  return  $y_i[\delta]$  with  $y_i \leftarrow A_\gamma[i, 1] \cdot W + \sum_{j=1}^C A_\gamma[i, j] \cdot G(s_{\gamma,j})$ .

21: function Decode( $\llbracket y \rrbracket_1, \dots, \llbracket y \rrbracket_p$ ) return  $\sum_{i=1}^p \llbracket y \rrbracket_i$ .
```

construction produces matrices with p rows and $\binom{p}{m+1}$ columns. As in [BGI15], we sample one random seed per column. The i -th key contains the i -th row of the matrix and includes the j -th seed if the corresponding cell $A[i, j]$ is not null.

As in [BGI15], it is necessary that (for any given γ) at least one seed $s_{\gamma,j}$ remains unknown to the adversary. With all the seeds, they could unmask the correction word W and recover β . Our scheme provides each seed to $m + 1$ parties, so under honest majority, at least one combination of $m + 1$ out of p parties contains only honest agents. Thus, there is at least one seed unknown to an adversary.

Our optimization cannot be extended to dishonest majority. Indeed, with $m > p/2$, the adversary would know all the seeds because there would be at least one corrupted party in each combination of $m + 1$ out of p parties.

Based on these intuitions, we can consider the following security theorem:

Theorem 9.3. Let $\lambda \in \mathbb{N}$, $N, p \in \mathbb{N}$, then the tuple (Gen, Eval, Decode) as described in Algorithm 10 is an FSS scheme for the family of all point functions with $\alpha \in \{1, \dots, N\}$ and any $\beta \in \mathbb{F}_q$.

Assuming that there exists a secure PRG, then this scheme is correct and private against at most m semi-honest parties with $m < p/2$.

Proof. As we modified only slightly the scheme of [BGI15] (i.e., we only updated the definition of the matrix of shares based on the honest-majority assumption), our proof follows the same structure as theirs. Thus, we only provide a proof sketch, and refer to [BGI15] for more details.

The correctness can be verified by an easy arithmetic exercise considering successively three cases: (1) $\gamma \neq \gamma_*$, (2) $\gamma = \gamma_*$ and $\delta \neq \delta_*$, and (3) $\gamma = \gamma_*$ and $\delta = \delta_*$. Using the $\sqrt{N} \times \sqrt{N}$ grid (illustrated in Fig. 9.2), we represent any input x as (γ, δ) and α as (γ_*, δ_*) .

For privacy, we must show that there exists a simulator that outputs samples from a distribution that is computationally indistinguishable from the distribution of the real DPF keys. We propose to study separately: the random seeds $s_{\gamma,j}$, the correction word W , and the coefficient shares $A_\gamma[i, j]$.

The simulation is straightforward: for each (simulated) seed, sample a random seed $\widetilde{s}_{\gamma,j}$; for the correction word, sample a random vector $\widetilde{W}$; for the coefficients, for each S_j combination of $m+1$ parties of p , for each i in S_j , sample a random value and store it in $\widetilde{A}_\gamma[i, j]$ (the rest of $\widetilde{A}_\gamma$ is null). The simulator can return “simulated” keys based on these elements.

Since both the real and simulated seeds are randomly sampled, the simulator’s output distribution $(\widetilde{s}_{\gamma,j})$ is computationally indistinguishable from that induced by the distribution of a single output of Gen.

The correction word W is a secret vector (i.e., $e_{\delta_*} \cdot \beta$) masked with the output of $\binom{p}{m+1}$ seeded PRGs. Remember that a key k_i contains a seed $s_{\gamma,j}$ only if the i is part of the j -th combination of m out of p parties. So, under honest majority, there is always at least one seed unknown to the adversary controlling m out of p parties. Hence, the correction word W is computationally indistinguishable from the randomly sampled $\widetilde{W}$, because it is masked with the output of (at least) one PRG seeded with a seed unknown to the adversary [CBM15].

Finally, each coefficient is shared between $m+1$ parties, so an adversary controlling m parties cannot distinguish real shares from random values $\widetilde{A}_\gamma[i, j]$ provided by the simulator. $\square$

Key size optimization: Instead of using a $\sqrt{N} \times \sqrt{N}$ grid, we should use a grid with $\sqrt{N}(C)^{-1}$ rows and $\sqrt{N \cdot C}$ columns. This “non-square” grid leverages the fact that each row requires C seeds, and a unique vector W for all columns. Thanks to this trick, we obtain a better key size: $O(\sqrt{N \cdot C} \log q)$.

Extension to comparison functions [BGI15] presented a simple adaptation of their DPF to support comparison functions; functions such that $f(x) = \beta$ when $x \leq \alpha$, 0 otherwise. These schemes have applications notably in machine learning [BGI19, KPM24]. We can naively reuse our optimization on this other scheme.

9.3 Key size benchmark

This section compares our optimized scheme to existing schemes in order to identify asymptotic and practical key size reductions.

We focus our comparison on schemes supporting all possible DPF applications; excluding schemes based on elliptic curves that support a limited number of applications due to their non-linear decoding [KPM24] (e.g., do not support PIR).

Asymptotic Our scheme provides a key size of $O(\sqrt{N \cdot \binom{p}{m+1}} \log q)$, which is clearly better than the key size of [BGI15] (i.e., $O(\sqrt{N} q^{\frac{p-1}{2}} \log q)$).

As [BK022] built an honest-majority scheme with $O(\sqrt[4]{N})$ key size upon [BGI15], we can comment how we distinguish from them. Their paper does not modify [BGI15], but combine it with replicated secret sharing to reduce the dependency on N (i.e., $O(\sqrt[4]{N})$ instead of $O(\sqrt{N})$). However, their approach worsened the exponential factors already in [BGI15]; as shown in our benchmark below. On the contrary, we leverage the honest-majority scheme to avoid exponential factors present in [BGI15], but we maintained the same dependence on N .

The IT DPF by [BK022] has a key size comparable to ours: $O(\sqrt{N} \cdot \binom{p}{m+1} \log q)$. However, our approach saves a factor $\sqrt{\binom{p}{m+1}}$, yielding significant key size reductions in practice.

While these asymptotic comparisons are informative, they are often insufficient to assess practical performance. [BK022] exemplify this problem: although they substantially reduced the dependence on the domain size N , they kept exponential factors without providing any concrete efficiency analysis. Therefore, we present a detailed comparison based on exact key sizes to offer a more accurate assessment.

Exact As we aimed to remove the exponential factors q^p (for outputs in $\mathbb{F}_q$), we start by studying the dependency on q . Fig. 9.4 compares key sizes for varying prime moduli. Our benchmark includes a curve “Trivial scheme” corresponding to the most trivial DPF: sharing the function truth table (i.e., $O(N)$ key size). This curve serves as baseline to identify impractical solutions. The PRG-based solutions [BGI15, BK022] have keys orders of magnitude larger than those of this trivial solution, while the IT DPF of [BK022] and ours are below.

Recently, [BCGI22] showed that, for a composite modulus $m = q_1 q_2 \dots q_l$, we can use the Chinese Remainder Theorem (CRT) to replace q^p with $\sum q_i^p$. Fig. 9.4a

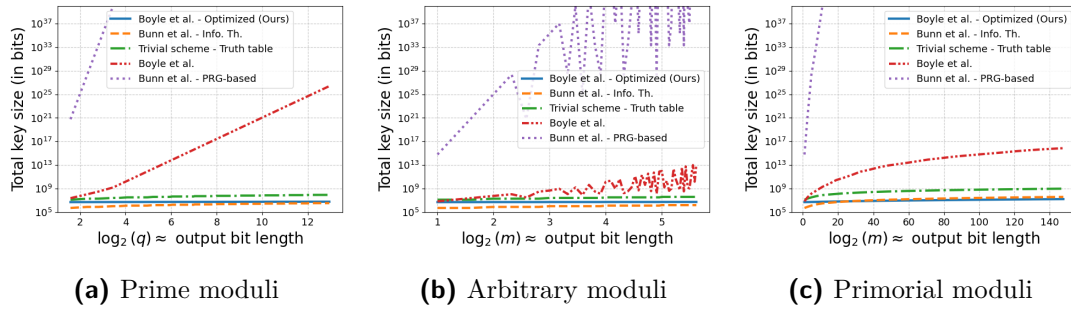


Figure 9.4 / Key size of various DPF schemes for varying moduli ($p = 7$)

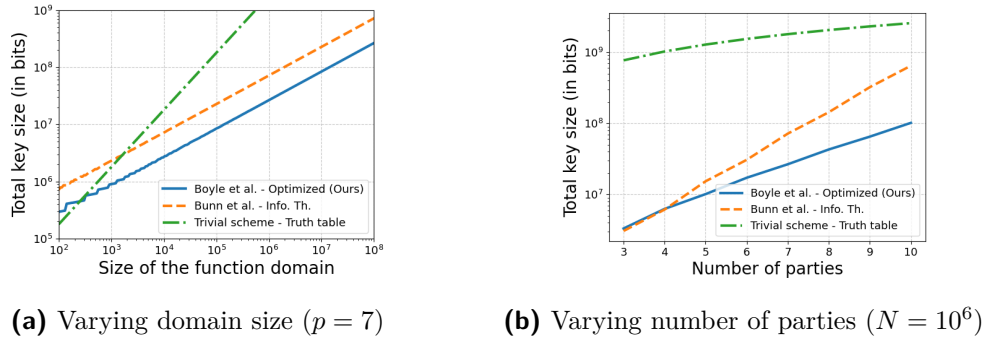


Figure 9.5 / Exact key size of the most efficient DPF schemes

compares key sizes for arbitrary moduli, but the variations make the figure poorly readable. Instead, Fig. 9.4c compares key sizes for primorial moduli; a primorial is the product of the first n primes. Primorials is the best case of this CRT trick as they provide composite moduli with the smallest primes possible.

Even with the CRT trick, the existing PRG-based schemes [BGI15, BK022] provides key sizes larger than the trivial scheme for any modulus above 210. As their key sizes are impractical, we exclude these schemes from our other figures to focus on the comparison of our scheme to practical schemes.

Fig. 9.4c also shows that the key size of the IT scheme of [BK022] grows faster with the modulus than ours. This phenomenon is explained by the fact that our key size is dominated by components conditioned by a security parameter that is independent of the modulus.

Fig. 9.5a compares the practical schemes for varying function domain sizes. **Our key size is $2.4\times$ smaller than the best existing scheme.** Moreover, Fig. 9.5b shows that **our scheme has a better scaling with the number of parties** thanks to the factor $\sqrt{\binom{p}{m+1}}$ identified in the asymptotic comparison.

Conclusion Our optimization based on the honest-majority assumption transformed an PRG-based DPF [BGI15] with impractical key sizes into the DPF with the smallest key sizes. This chapter proves that, like in two- and three-party schemes, PRG is a promising primitive to build DPF schemes with compact keys.

DDH-based schemes for multi-party Function Secret Sharing

Our first FSS contribution successfully optimized an existing scheme, resulting to the best performing multi-party DPF. However, the gap between multi-party and two-party schemes remains unchanged because this optimized scheme matches the asymptotic key size of [BK022]: $O(\sqrt{N} \binom{p}{m+1})$. Thus, further work is necessary to go beyond the $O(\sqrt{N})$ asymptotic key size.

Building on this objective, this chapter designs a DDH-based DPF with $O(\sqrt[3]{N})$ key size. Our benchmark highlights key sizes up to $10\times$ smaller (for realistic problem sizes) than state-of-the-art schemes. Finally, we extend our technique to schemes supporting comparison functions.

This chapter is drawn from [DHPR25a].

10.1 Introduction

When relying on standard hardness assumptions (which then excludes [KP25]), the existing literature leaves two choices for multi-party DPF: either an $O(\sqrt[4]{N})$ key size with exponential factors [BK022], or an $O(\sqrt{N})$ key size without exponential factors [BK022, CBM15].

Our goal is to provide an intermediary solution, a better trade-off: $O(\sqrt[3]{N})$ key size without exponential factors. Similarly, we want to extend these results to DCF, and provide the same choice range as in DPF.

We would like to emphasize the important contribution of [KP25] (published concurrently to our work) which presents a multi-party scheme with logarithmic key sizes. Unfortunately, it requires the introduction of two new hardness assumptions. Further research is still necessary to estimate the parameters under which these assumptions hold. In this chapter, on the other hand, we want to rely on standard, widely-established assumptions.

Our Contributions

- (1) We build a **multi-party DDH-based DPF with $O(\sqrt[3]{N})$ key size**. Our scheme provides keys **up to $\times 10$ smaller** than existing works.
- (2) We **extend our approach to build a DCF with $O(\sqrt[3]{N})$ key size**.
- (3) As DDH-based schemes require encoding the secret as a DDH group element, we present **two encodings, discuss their properties and applications**.

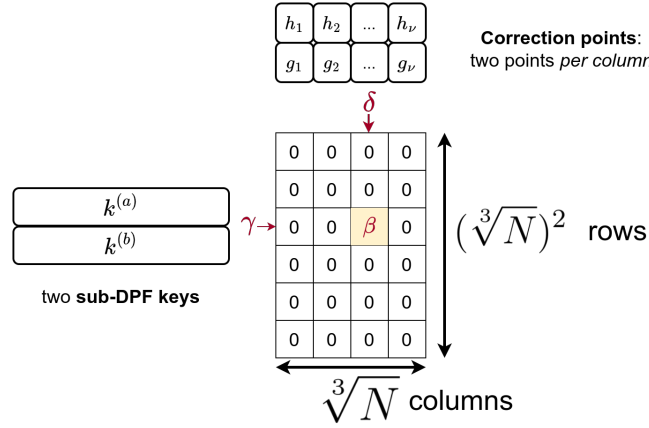


Figure 10.1 / High-level structure of our DDH-based DPF

10.2 DDH-based Distributed Point Function

This section introduces a DDH-based approach to build a scheme with $O(\sqrt[3]{N})$ key size upon the information-theoretic scheme of [BK022] which has an $O(\sqrt{N})$ key size. We call “sub-DPF” or “sub-scheme” the DPF scheme of [BK022] on which we apply our DDH-based optimization.

As this section focuses on point functions, we use the notation (α, β) to refer to the parameters of the secret function f (i.e., $f(x) = \beta$ if $x = \alpha$, 0 otherwise).

10.2.1 Scheme

We represent the domain as a grid of dimensions: $(\sqrt[3]{p})^2 \times \sqrt[3]{p}$ (see Figure 10.1). The non-zero value is in cell (γ_*, δ_*) .

Gen algorithm Let $\mathbb{G}$ be a cyclic group of prime order q_0 , with g a generator. We assume that the DDH assumption holds in $\mathbb{G}$. Let $g_\beta = \text{Encode}_{\text{DDH}}(\beta)$ be the encoding of β in $\mathbb{G}$ (possible encodings are discussed in Section 10.5). Our Gen algorithm starts by sampling one random number $r \xleftarrow{R} \mathbb{F}_{q_0}$ and compute r_{inv} its multiplicative inverse. Note that this multiplicative inverse exists because q_0 is prime (and large so sampling a 0 is unlikely).

We then use the sub-DPF to share two secret point functions f_a, f_b such that $f_a(x) = r$ and $f_b(x) = 1$, if $x = \gamma_*$, otherwise 0. This step outputs two sets of DPF keys $\{k_1^{(a)}, \dots, k_p^{(a)}\}$ and $\{k_1^{(b)}, \dots, k_p^{(b)}\}$, with each key of size $O(\sqrt[3]{N})$ thanks to the sub-DPF. For each column δ , we sample a random generator $g_\delta \xleftarrow{R} \mathbb{G}$. For all $\delta \neq \delta_*$, we set the “correction points” $h_\delta = g_\delta^{r_{\text{inv}}}$. We set the correction points for the column δ_* : $h_{\delta_*} \leftarrow g_{\delta_*}^{-r_{\text{inv}}} \cdot g_{\beta}^{r_{\text{inv}}}$. Each party i receives a key k_i containing two DPF “sub-keys” $k_i^{(a)}, k_i^{(b)}$ and all corrections points.

Eval algorithm Party i represents x as (γ', δ') and evaluates: $\llbracket s_a \rrbracket_i \leftarrow \text{Eval}_{\text{DPF}}(k_i^{(a)}, \gamma')$ and $\llbracket s_b \rrbracket_i \leftarrow \text{Eval}_{\text{DPF}}(k_i^{(b)})$. Each party obtains $\llbracket f(x) \rrbracket_i = h_{\delta'}^{\llbracket s_a \rrbracket_i} \cdot g_{\delta'}^{\llbracket s_b \rrbracket_i}$.

Algorithm 11 details our DDH-based scheme. Fig. 10.1 provides a high-level representation of our DPF keys, with an $O(\sqrt[3]{N} \cdot \binom{p-1}{m} \cdot (\lambda + \log q))$ key size.

Algorithm 11 DDH-based DPF scheme

- 1: Let $\mathbb{G}$ be a cyclic group of prime order q_0 and with g a genertor. We assume that the DDH assumption holds in $\mathbb{G}$.
 - 2: Let $(\text{Gen}_{\text{DPF}}^{(*)}, \text{Eval}_{\text{DPF}}^{(*)}, \text{Decode}_+)$ be the information-theoretic proposed by [BK022] (with output in $\mathbb{F}_{q_0}$).
 - 3: Let $\text{Encode}_{\text{DDH}} : \mathcal{Y} \rightarrow \mathbb{G}$ and $\text{Decode}_{\text{DDH}} : \mathbb{G} \rightarrow \mathcal{Y}$ be two functions such that $\text{Encode}_{\text{DDH}}(0) = g^0$ and $\text{Decode}_{\text{DDH}}(\text{Encode}_{\text{DDH}}(y)) = y$ for any $y \in \mathcal{Y}$.
 - 4: **function** $\text{Gen}_{\text{DPF}}(\alpha, \beta, p, m)$
 - 5: Let $g_\beta \leftarrow \text{Encode}_{\text{DDH}}(\beta)$.
 - 6: Let (γ_*, δ_*) be the position of α in a $(\nu)^2 \times \nu$ grid, with $\nu = \lceil \sqrt[3]{N} \rceil$
 - 7: Sample one (non-zero) $r \xleftarrow{R} \mathbb{F}_{q_0}$ and set $r_{\text{inv}} \leftarrow r^{-1}$.
 - 8: Generate two sets of DPF keys:

$$\{k_1^{(a)}, \dots, k_p^{(a)}\} \leftarrow \text{Gen}_{\text{DPF}}^{(*)}(\gamma_*, r_a, p, m)$$

$$\{k_1^{(b)}, \dots, k_p^{(b)}\} \leftarrow \text{Gen}_{\text{DPF}}^{(*)}(\gamma_*, 1, p, m).$$
 - 9: **for** $\delta \in \{1 \dots \nu\}, \delta \neq \delta_*$ **do**
 - 10: Sample a point g_δ from $\mathbb{G}$ and set $h_\delta \leftarrow g_\delta^{-r_{\text{inv}}}$.
 - 11: Sample a g_{δ_*} from $\mathbb{G}$ and set $h_{\delta_*} \leftarrow g_{\delta_*}^{-r_{\text{inv}}} \cdot g_\beta^{r_{\text{inv}}}$.
 - 12: Set $k_i = (k_i^{(a)} || k_i^{(b)} || g_1 || h_1 || \dots || g_\nu || h_\nu), \forall i \in \{1 \dots p\}$
 - 13: **return** $(k_1, \dots, k_p)$.
 - 14: **function** $\text{Eval}_{\text{DPF}}(k_i, x)$
 - 15: Let (γ', δ') be the position of x in a $(\sqrt[3]{N})^k \times \sqrt[3]{N}$ grid.
 - 16: Parse k_i as $k_i = (k_i^{(a)} || k_i^{(b)} || g_1 || h_1 || \dots || g_\nu || h_\nu)$.
 - 17: Let $\llbracket s_a \rrbracket_i \leftarrow \text{Eval}_{\text{DPF}}^{(*)}(k_i^{(a)}, \gamma'), \llbracket s_b \rrbracket_i \leftarrow \text{Eval}_{\text{DPF}}^{(*)}(k_i^{(b)}, \gamma')$.
 - 18: Let $\llbracket f(x) \rrbracket_i \leftarrow h_{\delta'}^{\llbracket s_a \rrbracket_i} \cdot g_{\delta'}^{\llbracket s_b \rrbracket_i}$.
 - 19: **return** $\llbracket f(x) \rrbracket_i$ $\triangleright \llbracket f(x) \rrbracket_i \in \mathbb{G}$.
 - 20: **function** $\text{Decode}_{\text{DPF}}(\llbracket f(x) \rrbracket_1, \dots, \llbracket f(x) \rrbracket_p)$ **return** $\text{Decode}_{\text{DDH}}(\prod_{i=1}^p \llbracket f(x) \rrbracket_i)$
-

10.2.2 Asymptotic key size

Fig. 10.1 provides a high-level representation of our DPF keys, with an $O(\sqrt[3]{N} \cdot \binom{p-1}{m-1} \cdot (\lambda + \log q))$ key size. Let us break down this asymptotic cost.

First, we have two sub-DPF keys. The previous DPFs are defined over the domain

$\{1 \dots (\sqrt[3]{p})^2\}$ to “cover all the rows”. We have $(\sqrt[3]{p})^2$ rows and use [BK022] as sub-DPF (with an $O(\sqrt{M} \binom{p-1}{m-1} \log q_0)$ key size for a domain of size M). Each sub-DPF key is then of size $O(\sqrt[3]{p} \cdot \binom{p-1}{m-1} \cdot \log q_0)$ ($M = (\sqrt[3]{p})^2$ in our case).

Second, for each column δ , we have $g_\delta, h_\delta \in \mathbb{G}$. For convenience, assume $\mathbb{G} = \mathbb{F}_q^\times$. As we have $\sqrt[3]{p}$ columns, the total size of these elements is $O(\sqrt[3]{p} \cdot \log q)$.

In our scheme, λ is implicitly present because DDH is assumed to be hard in cyclic group $\mathbb{G}$. As $\mathbb{G}$ is of order q_0 , we have $\lambda = O(\log q_0)$.

Supporting other sub-DPFs We use a grid of size $(\sqrt[3]{N})^2 \times \sqrt[3]{N}$ because the sub-DPF we use has an $O(\sqrt{M})$ key size (for a domain size of M). However, using our DDH-based approach, we can replace [BK022] with any scheme that outputs additive shares in $\mathbb{F}_{q_0}$. Specifically, for any sub-scheme with key size $O(\sqrt[k]{N})$, our method yields a DDH-based variant with key size $O(\sqrt[k+1]{N})$.

In particular, our approach can be applied to the dishonest-majority scheme proposed in [GWW25]. The resulting scheme (combining [GWW25] with our DDH-based approach) is secure against a dishonest majority and has key size $O(\sqrt[3]{N} \cdot p^3 \lambda^3 \cdot (\log q + \lambda))$. As shown in Section 10.3, the scheme of [GWW25] already has larger keys than sharing the function’s truth table. Therefore, extending our scheme using [GWW25] would result in an impractical solution, because [GWW25] is impractical.

Given this, we do not further develop this extension in our work. Instead, we focus our discussions solely on our honest-majority scheme, which, in contrast, provides substantial key size reductions for practical problem sizes.

Since we can reduce key sizes from $O(\sqrt[k]{N})$ to $O(\sqrt[k+1]{N})$, our result raises a key question: can we build a multi-party scheme with logarithmic key size via a recursive application of our scheme? Our DDH-based scheme outputs multiplicative shares (in a DDH group), so it cannot be used recursively.

10.2.3 Security

In our scheme, the DDH assumption prevents an adversary from distinguishing the correction points, and then prevents them from recovering the column δ_* containing the non-zero from the key. To prove this intuition, we propose to consider the following theorem:

Theorem 10.2. Let $\lambda \in \mathbb{N}$, $N, p \in \mathbb{N}$, then $(\text{Gen}_{\text{DPF}}, \text{Eval}_{\text{DPF}}, \text{Decode}_{\text{DPF}})$ as described in Algorithm 11 is an FSS scheme for the family of point functions.

Assuming that the DDH assumption holds in $\mathbb{G}$ and that the information-theoretic scheme $(\text{Gen}_{\text{DPF}}^{(*)}, \text{Eval}_{\text{DPF}}^{(*)}, \text{Decode}_+)$ from [BK022] is a correct and private DPF scheme, then this scheme is correct and private against at most m semi-honest parties with $m < p/2$.

Proof. Correctness: We represent any input x as (γ, δ) and α as (γ_*, δ_*) . We need to study the FSS output in three cases: (1) $\gamma \neq \gamma_*$, (2) $\gamma = \gamma_*$ and $\delta \neq \delta_*$, and (3) $\gamma = \gamma_*$ and $\delta = \delta_*$.

If $\gamma \neq \gamma_*$, each party i holds $\llbracket s_a \rrbracket_i, \llbracket s_b \rrbracket_i$ (obtained using their sub-DPF keys), such that $\sum_i \llbracket s_a \rrbracket_i = \sum_i \llbracket s_b \rrbracket_i = 0$ (because the sub-DPF is correct by assumption and outputs additive shares). For any δ :

$$\begin{aligned} \prod_i \llbracket f(x) \rrbracket_i &= \prod_i (h_\delta^{\llbracket s_a \rrbracket_i} \cdot g_\delta^{\llbracket s_b \rrbracket_i}) = h_\delta^{\sum_i \llbracket s_a \rrbracket_i} \cdot g_\delta^{\sum_i \llbracket s_b \rrbracket_i} = h_\delta^0 \cdot g_\delta^0 = g^0 \\ \Rightarrow \text{Decode}_{\text{DPF}}(\text{Eval}_{\text{DPF}}(k_1, x), \dots, \text{Eval}_{\text{DPF}}(k_p, x)) &= \text{Decode}_{\text{DDH}}(g^0) = 0 \end{aligned}$$

If $\gamma = \gamma_*$ and $\delta \neq \delta_*$, each party i holds $\llbracket s_a \rrbracket_i, \llbracket s_b \rrbracket_i$, such that $\sum_i \llbracket s_a \rrbracket_i = r$ and $\sum_i \llbracket s_b \rrbracket_i = 1$. The parties obtain:

$$\begin{aligned} \prod_i \llbracket f(x) \rrbracket_i &= \prod_i (h_\delta^{\llbracket s_a \rrbracket_i} \cdot g_\delta^{\llbracket s_b \rrbracket_i}) = h_\delta^{\sum_i \llbracket s_a \rrbracket_i} \cdot g_\delta^{\sum_i \llbracket s_b \rrbracket_i} \\ &= g_\delta^{-r_{\text{inv}} \cdot r} \cdot g_\delta = g_\delta^{-1} \cdot g_\delta = g^0 \\ \Rightarrow \text{Decode}_{\text{DPF}}(\text{Eval}_{\text{DPF}}(k_1, x), \dots, \text{Eval}_{\text{DPF}}(k_p, x)) &= \text{Decode}_{\text{DDH}}(g^0) = 0 \end{aligned}$$

If $\gamma = \gamma_*$ and $\delta = \delta_*$, we also have $\sum_i \llbracket s_a \rrbracket_i = r$ and $\sum_i \llbracket s_b \rrbracket_i = 1$, but h_δ is now equal to $(g_\delta^{-r_{\text{inv}}} \cdot g_\beta^{r_{\text{inv}}})$. It implies:

$$\begin{aligned} \prod_i \llbracket f(x) \rrbracket_i &= \prod_i (h_\delta^{\llbracket s_a \rrbracket_i} \cdot g_\delta^{\llbracket s_b \rrbracket_i}) = h_\delta^{\sum_i \llbracket s_a \rrbracket_i} \cdot g_\delta^{\sum_i \llbracket s_b \rrbracket_i} \\ &= g_\delta^{-r_{\text{inv}} \cdot r} \cdot g_\beta^{r_{\text{inv}} \cdot r} \cdot g_\delta = g_\beta \\ \Rightarrow \text{Decode}_{\text{DPF}}(\text{Eval}_{\text{DPF}}(k_1, x), \dots, \text{Eval}_{\text{DPF}}(k_p, x)) &= \text{Decode}_{\text{DDH}}(g_\beta) = \beta \end{aligned}$$

Privacy: This proof will rely on a construction also used in [CBM15] to prove the security of their DDH-based DPF: Seed-Homomorphic PseudoRandom Generators (SH-PRG). A PRG G is seed-homomorphic if it satisfies the following property: for any seeds s_1, s_2 , $G(s_1 + s_2) = G(s_1) \oplus G(s_2)$. In particular, [CBM15] rely on DDH-based SH-PRG G_{DDH} : given L randomly sampled public parameters $(g_1, \dots, g_L) \in \mathbb{G}^L$, $G_{\text{DDH}}(s) = (g_1^s, \dots, g_L^s)$ for any seed s . SH-PRGs naturally inherit all properties of a PRG, notably the fact that the output is computationally indistinguishable from true randomness if the seed is unknown to the adversary.

Recall the two assumptions of Theorem 10.2: (1) the DDH is hard, and (2) the sub-DPF scheme is private and correct. To prove the privacy (see Definition 2.5), we must show that (for every set of corrupted parties $S \subseteq \{1 \dots p\}$ of size m) there exists an efficient simulator that, for any input functions, outputs samples from a distribution that is computationally indistinguishable from the distribution of the DPF keys.

- ▲ Remind that each key k_i contains the following elements: two sub-DPF keys $(k_i^{(a)}, k_i^{(b)})$ and pairs of “correction points” $(h_1, g_1), \dots, (h_\nu, g_\nu)$

By assumption, the sub-DPF scheme $(\text{Gen}_{\text{DPF}}^{(*)}, \text{Eval}_{\text{DPF}}^{(*)}, \text{Decode}_+)$ is secure, so there exists an efficient simulator to simulate $(k_i^{(a)}, k_i^{(b)})$ (for all $i \in \{1 \dots p\}$).

To simulate the correction points, for each $\delta \in \{1 \dots \nu\}$, the simulator samples a random generator $\widetilde{g}_\delta \in \mathbb{G}$ and another random element $\widetilde{h}_\delta \in \mathbb{G}$. Note that g_δ is also sampled randomly during the key generation, so the distribution of g_δ is indistinguishable from the distribution of $\widetilde{g}_\delta$.

Finally, we have to prove that the adversary cannot distinguish h_δ from $\widetilde{h}_\delta$. To prove this statement, we will consider successively two scenarios: (1) the adversary knows (δ_*, β) and (2) the adversary does not know the pair.

Let us first assume that the adversary knows (δ_*, β) . Note that we can rewrite the tuple $(h_1, \dots, h_{\delta_*}, \dots, h_\nu)$ as $((g_1^{-1})^{r_{\text{inv}}}, \dots, (g_{\delta_*}^{-1} g_\beta)^{r_{\text{inv}}}, \dots, (g_\nu^{-1})^{r_{\text{inv}}})$. As $g_1, \dots, g_\nu$ and g_β are known to the attacker $((g_1^{-1}), \dots, (g_{\delta_*}^{-1} g_\beta), \dots, (g_\nu^{-1}))$, these values can be interpreted as the public parameters of a DDH-based SH-PRG. The only condition on the public parameters is that they must be independently sampled generators. This condition is trivially satisfied for all g_δ with $\delta \neq \delta_*$, and we can observe that g_{δ_*} is independent of g_β so $(g_{\delta_*}^{-1} g_\beta)$ is indistinguishable from random.

Let G_{DDH}^* then be a DDH-based SH-PRG with $((g_1^{-1}), \dots, (g_{\delta_*}^{-1} g_\beta), \dots, (g_\nu^{-1}))$ as public parameters. We have:

$$(h_1, \dots, h_\nu) = ((g_1^{-1})^{r_{\text{inv}}}, \dots, (g_{\delta_*}^{-1} g_\beta)^{r_{\text{inv}}}, \dots, (g_\nu^{-1})^{r_{\text{inv}}}) = G_{DDH}^*(r_{\text{inv}})$$

In other words, $(h_1, \dots, h_\nu)$ is the output of G_{DDH}^* on the seed r_{inv} . The seed r_{inv} is unknown to the adversary, so they cannot distinguish the real $(h_1, \dots, h_\nu)$ from the randomly sampled $(\widetilde{h}_1, \dots, \widetilde{h}_\nu)$ outputted by the simulator.

Let us now assume that the adversary does not know (δ_*, β) . If the adversary were able to distinguish h_δ from $\widetilde{h}_\delta$ without this knowledge, we could trivially build a distinguisher for an adversary knowing (δ_*, β) . Since the h_δ can be formulated as the output of a DDH-based SH-PRG, this hypothetical distinguisher would break the DDH assumption (upon which the SH-PRG is built). By assumption, DDH is hard, so this adversary cannot distinguish real h_δ from the simulator output $\widetilde{h}_\delta$. $\square$

10.2.4 Comparison with existing works

Table 10.3 compares existing schemes to ours. We achieve two goals: (1) having a practical scheme (i.e., avoids exponential factors contrary to [BGI15, BK022]), and (2) improving the dependency on the function domain size N (compared to other practical schemes [BK022, CBM15, GWW25]). In other words, our $O(\sqrt[3]{N})$ key size is a nice intermediary solution between an $O(\sqrt[4]{N})$ key size with exponential factors [BK022] and $O(\sqrt{N})$ key sizes without exponential factors [BK022, CBM15, GWW25].

Scheme	Majority	Assumptions	Key size
[BGI15]	Dishonest	PRG	$O(\sqrt{N} \cdot q^{\frac{p-1}{2}} \cdot (\log q + \lambda))$
[CBM15, KPM24]	Dishonest	DDH	$O(\sqrt{N} \cdot (\lambda + \log q))$
PRG-based [BK022]	Honest	PRG	$O(\sqrt[4]{N} \cdot \sqrt{q^{p^m}} \cdot \binom{p-1}{m} \cdot (\lambda + \log q))$
Info.-Th. [BK022]	Honest	None	$O(\sqrt{N} \cdot \binom{p-1}{m} \cdot \log q)$
[GWW25]	Dishonest	PRG	$O(\sqrt{N} \cdot p^3 \lambda^3 \cdot (\log q + \lambda))$
[KP25]	Dishonest	DDH + 2 new hardness assumpt.	$O(\log N \cdot (\log q + \lambda))$
Our scheme	Honest	DDH	$O(\sqrt[3]{N} \cdot \binom{p-1}{m} \cdot (\lambda + \log q))$

Table 10.3 / Comparison of the multi-party DPF schemes

The scheme of [KP25] achieves the best known asymptotic key size, namely $O(\log N)$. However, this improvement comes at the cost of relying on two new hardness assumptions whose concrete security is not yet well understood. The work of [KP25] represents an exciting theoretical advancement, but the lack of concrete security hinders its implementation *for now*. Further research is needed to determine parameter sets that achieve standard security levels (e.g., 128-bit or 256-bit). In contrast, our scheme has $O(\sqrt[3]{N})$ key sizes, but is based solely on the DDH assumption; a *standard and extensively studied cryptographic assumption*.

10.2.5 Pseudo-random secret sharing

Pseudo-random secret sharing (PRSS) [CDI05] is a protocol reducing the communication cost of secret sharing protocols thanks to pseudo-random seed expansion. As most multi-party schemes [BGI15, BK022, CBM15] (including ours) output DPF keys containing secret shares, we can reduce their size thanks to PRSS.

In particular, we rely on the simplest form of PRSS applied to additive secret shares. Additive secret sharing on a secret s normally outputs p shares $\{\llbracket s \rrbracket_1 \dots \llbracket s \rrbracket_p\}$, such that $\sum_i \llbracket s \rrbracket_i = s$. With PRSS, the secret holder generates $p - 1$ random seeds $r_1, \dots, r_{p-1}$ and a share $\llbracket s \rrbracket_p$, such that $\llbracket s \rrbracket_p + \sum_{i=1}^{p-1} G(r_i) = s$ (with G a PRG). One shareholder receives $\llbracket s \rrbracket_p$ and the other $p - 1$ parties receive a random seed r_i .

The random seeds can be reused to share multiple values. To share a vector of n values, the secret holder sends $p - 1$ seeds and a single vector of n shares (vs. p vector of n shares in classic secret sharing). If the MPC protocol allows the secret holder to be a shareholder, they can keep the vector of shares and send

only random seeds to the other shareholders.

This technique amortizes the communication costs (i.e., key size) when the protocols require sharing large vectors. Since all practical schemes [BK022,CBM15] are compatible with PRSS, Section 10.3 compares the key sizes using PRSS.

10.3 Key size benchmark

We compare the exact key size of our DPF scheme to those of existing schemes. Our experiments report the total key size instead of the individual key size. This metric takes into account the key size amortization provided by PRSS. For this benchmark, we implement our scheme using elliptic curves $E(\mathbb{F}_q)$, as they provide prime-order cyclic groups in which the DDH is known to be hard. More specifically, we use the curve P-256.

While the scheme of [KP25] has the best asymptotic key sizes, it relies on two new hardness assumptions whose concrete security is not yet well understood. As a result, it remains unclear what parameter choices would yield 128-bit or 256-bit security. Without this information, the scheme cannot yet be implemented securely, and thus remains theoretical. Therefore, we cannot include it in our benchmark. Nevertheless, because of its low asymptotic key size, the scheme from [KP25] is a promising direction for future deployments; once its concrete security has been studied more thoroughly.

Exponential factors in existing works Fig. 10.4 illustrates the exponential factor of the PRG-based multi-party DPF [BGI15,BK022]. Fig. 10.4a represents their key size in function of the output bit length (i.e., $\log_2 q$ for an output in $\mathbb{F}_q$). While this problem was barely discussed in existing works [BGI15,BK022], Fig. 10.4 shows that it makes these techniques purely unusable. For only 12-bit moduli, the DPF of [BGI15] has a key size of 10^{18} bits and the PRG-based DPF of [BK022] has a key size of 10^{38} .

Fig. 10.4 includes a curve “Trivial scheme”, corresponding to the most trivial DPF implementation: sharing the truth table of the function. This trivial baseline is much better than these PRG-based solutions for prime fields other than $\mathbb{F}_2$. While other works [BCGI22,KPM24] warned about this potential scalability issue, this chapter provides the first experimental proof of their impracticality in $\mathbb{F}_q$.

In comparison, we can barely see on Fig. 10.4a the key size difference between the rest of the DPF schemes [Ours BK022,CBM15]. Note that Fig. 10.4 includes curves for the DDH-based schemes [Ours CBM15]. Such schemes are implemented with a fixed elliptic curve P-256, so the output is of fixed size (256 bits). Thus, their key sizes are constant on this figure.

Recently, [BCGI22] proposed an optimization based on the Chinese Remainder Theorem (CRT) to optimize the key sizes of PRG-based schemes [BGI15,BK022]

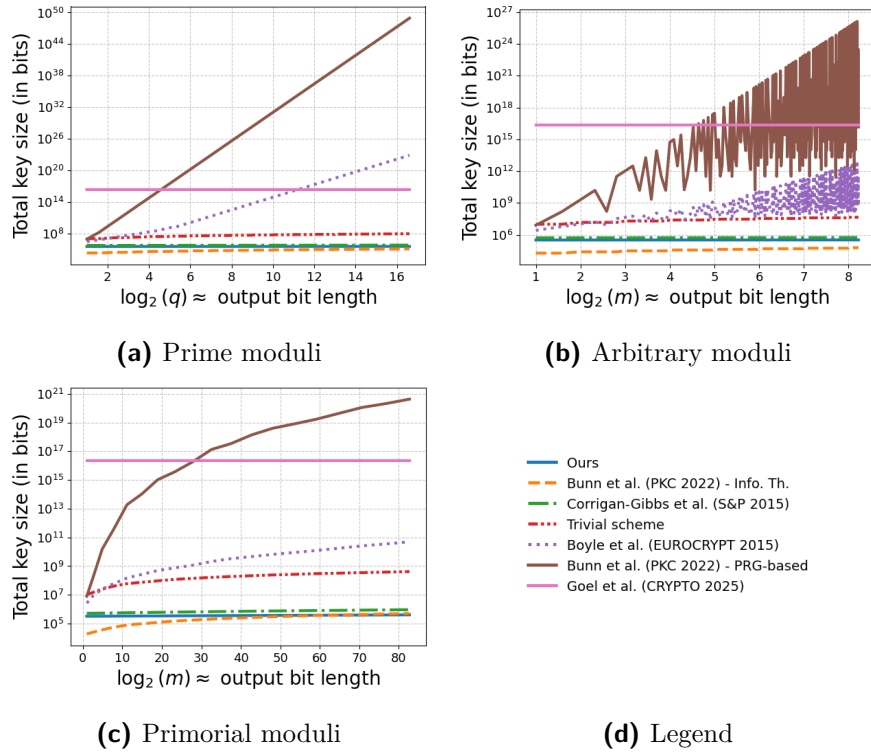


Figure 10.4 / DPF key sizes for varying moduli. Parameters: $p = 5$ parties, $N = 10^6$

for composite moduli. Thanks to this CRT trick, we can replace the exponential factor q^p with a sum of smaller factors $\sum_i q_i^p$ (with q_i the prime factors of the composite modulus).

Fig. 10.4b represents the key sizes for varying composite moduli, but the figure is hard to read, because this trick is sensitive to the prime decomposition. Two numbers m and $m + 1$ can have completely different prime factors. To provide more readable results, Fig. 10.4c represents the key sizes for varying primorial moduli. A primorial is a composite number whose prime factors are the n smallest primes. Such numbers are the best case scenario for the CRT trick because they provide the smallest factors possible. On Fig. 10.4c, we observe that the trick significantly improves the key size of the PRG-based schemes, but they remain way above the trivial scheme for any modulus above 210.

Finally, Fig. 10.4 provides a key insight into [GW25]. Note that their key size may seem constant in q on Fig. 10.4, but it is not; their key size is simply dominated by factors independent of q . While their work replaces the exponential factor present in [BG15] with a polynomial factor of $p^3 \lambda^3$, we demonstrate that, for realistic problem sizes, this polynomial term actually exceeds q^p . As a result, the asymptotic improvement offered by [GW25] does not translate into practical efficiency: the key size remains significantly larger than that of the trivial scheme. In other words, although [GW25] presents a notable theoretical advance by eliminating the exponential term of [BG15], the resulting scheme remains impractical; performing worse than the trivial DPF scheme.

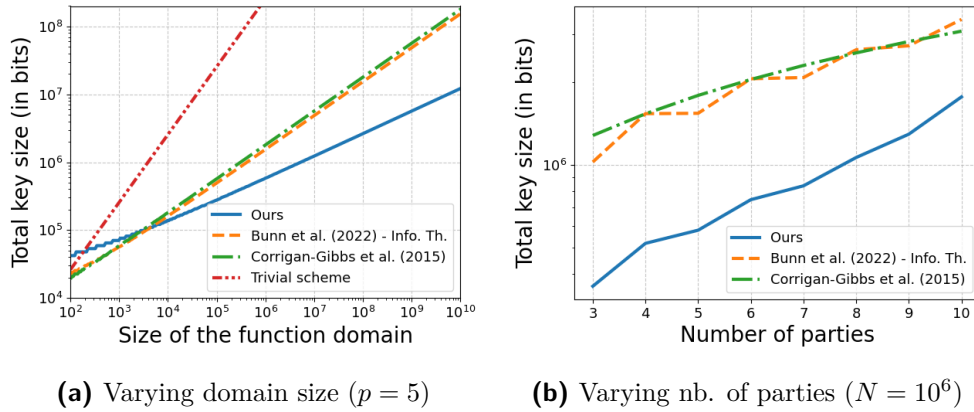


Figure 10.5 / Key sizes for varying domain sizes and number of parties

Since PRG-based schemes [BGI15, BK022, GWW25] are impractical for moduli larger than 210 (even with the CRT optimization of [BCGI22]), we exclude them from our other experiments. This allows us to observe precisely the key size improvements achieved by our scheme when compared to practical DPF schemes [BK022, CBM15].

Varying function domain Fig. 10.5a compares the key sizes in function of the domain sizes; varying from 10^2 to 10^{10} . [ZYP24] used the same problem sizes to benchmark three-party DPF schemes.

Our key size is the smallest for any domain size above 10^3 . When $N = 10^6$, our key size is 3 times smaller. When $N = 10^9$, our key size is 10 times smaller.

Our key size is higher than those of [BK022, CBM15] only when N is below 10^3 . To put this into perspective, Fig. 10.5a also shows that the trivial DPF scheme becomes the most efficient for smaller domain sizes (below 10^2). Thus, the advantage provided [BK022, CBM15] only holds for particularly small function domains and the trivial DPF could even be preferable in such cases.

Varying number of parties Fig. 10.5b compares the key sizes in function of the number of parties; varying from 3 to 10. We observe that [CBM15] has a better scaling than our scheme, but our key size remains smaller.

The growth of our key size is not as smooth as the growth of the scheme by [CBM15]. For example, the growth between 3 and 4 parties is steeper than between 4 and 5. This phenomenon is caused by the honest-majority assumption: (contrary to [CBM15]) our total key size is correlated to the number of honest parties. Under the honest-majority assumption, 4-party and 5-party setups have exactly the same number of honest parties (i.e., 3). The same phenomenon is present for [BK022].

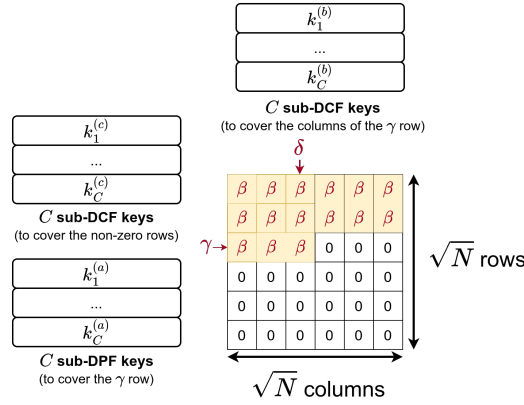


Figure 10.6 / High-level structure of our adapted DCF based on [BK022]

10.4 DDH-based Distributed Comparison Function

This section adapts our DPF scheme to comparison functions. First, we introduce an adaptation of [BK022] to comparison functions. Then, we build our DDH-based DCF upon this adapted scheme.

10.4.1 Adapting [BK022] to DCF

[BK022] presented a generic technique to build more efficient honest-majority DPF schemes from dishonest-majority schemes. They applied their technique on two existing schemes: the trivial DPF (i.e., a secret-shared truth table) and on the PRG-based scheme from [BGI15].

While their constructions provides some of the best solutions in DPF (see Table 10.3), they did not adapt them to comparison functions. This leaves the literature with only weak DCF baselines [BGI15, KPM24]. Thus, we propose to extend their work and build a DCF following the same intuition as their DPF. Moreover, our DCF scheme will also reused this adapted scheme as foundation.

Their DPF schemes To understand our adaptation, it is first important to understand their initial DPF. Their main intuition is to represent a point function f as the product of two point functions f_a and f_b (defined over small domains): $f(x) = f_a(\gamma) \times f_b(\delta)$ (with (γ, δ) the representation of x in a $(\sqrt{N} \times \sqrt{N})$ grid). Each function is shared using an existing DPF scheme. However, additive secret shares cannot be multiplied without communications.

Instead of using additive secret sharing, [BK022] used replicated secret sharing: each party receives multiple DPF keys. Under the honest-majority assumption, the shareholders can perform one offline multiplication on values shared via replicated secret sharing.

To sum up, [BK022] used replicated secret-sharing and the honest-majority assumption to build a technique reducing the key size of existing dishonest-majority

Algorithm 12 DCF adapted from the DPF of [BK022] (changes are in blue)

- 1: Let $(\text{Gen}_{\text{DPF}}^{(*)}, \text{Eval}_{\text{DPF}}^{(*)}, \text{Decode}_+)$ be a $(p-1, p)$ -secure DPF scheme with $O(\sqrt[p]{M})$ key size for a function domain of size M and $(\text{Gen}_{\text{DCF}}^{(*)}, \text{Eval}_{\text{DCF}}^{(*)}, \text{Decode}_+)$ be a $(p-1, p)$ -secure DCF scheme with $O(\sqrt[p]{M})$ key size.
 - 2: Let S be the sorted list of all possible combinations of $(p-m)$ elements from $\{1 \dots p\}$; it contains $C = \binom{p}{p-m}$ sets.
 - 3: **function** $\text{Gen}_{\text{DCF}}(\alpha, \beta, p, m)$
 - 4: Let (γ_*, δ_*) be the position of α in an $(\sqrt{N} \times \sqrt{N})$ grid.
 - 5: $\{k_1^{(a)}, \dots, k_C^{(a)}\} \leftarrow \text{Gen}_{\text{DPF}}^{(*)}(\gamma_*, \beta, C, C-1)$.
 - 6: $\{k_1^{(b)}, \dots, k_C^{(b)}\} \leftarrow \text{Gen}_{\text{DCF}}^{(*)}(\delta_*, 1, C, C-1)$
 - 7: $\{k_1^{(c)}, \dots, k_C^{(c)}\} \leftarrow \text{Gen}_{\text{DCF}}^{(*)}(\gamma_* - 1, \beta, C, C-1)$
 - 8: **for** $i \in \{1 \dots p\}$ **do** initialize k_i to an empty bit string.
 - 9: **for** $i \in \{1 \dots p\}, j \in \{1 \dots C\}$ **do**
 - 10: **if** $i \in S[j]$ **then** $k_i \leftarrow k_i || (k_j^{(a)}, k_j^{(b)}, k_j^{(c)})$.
 - 11: **else** $k_i \leftarrow k_i || (\perp, \perp, \perp)$.
 - 12: **return** $(k_1, \dots, k_p)$
 - 13: **function** $\text{Eval}_{\text{DCF}}(k_i, x)$
 - 14: Let (γ', δ') be the position of x in a $\sqrt{N} \times \sqrt{N}$ grid.
 - 15: Parse k_i as $k_i = (k_1^{(a)}, k_1^{(b)}, k_1^{(c)}) || \dots || (k_C^{(a)}, k_C^{(b)}, k_C^{(c)})$.
 - 16: Initialize $\text{res} \leftarrow 0$.
 - 17: **for** $j \in \{1 \dots C\}$ **do**
 - 18: **if** $i == \min S[j]$ $\triangleright$ Ensures that each share is only added once.
 - 19: $\text{res} \leftarrow \text{res} + \text{Eval}_{\text{DCF}}^{(*)}(k_j^{(c)}, \gamma')$
 - 20: **for** $j \in \{1 \dots C\}, j' \in \{1 \dots C\}$ **do**
 - 21: **if** $i == \min(S[j] \cap S[j'])$ $\triangleright$ Same as Line 18.
 - 22: $\text{res} \leftarrow \text{res} + \text{Eval}_{\text{DPF}}^{(*)}(k_j^{(a)}, \gamma') \times \text{Eval}_{\text{DCF}}^{(*)}(k_j^{(b)}, \delta')$.
 - 23: **return** res
 - 24: **function** $\text{Decode}_{\text{DCF}}(\llbracket f(x) \rrbracket_1, \dots, \llbracket f(x) \rrbracket_p)$ **return** $\sum_{i=1}^p \llbracket f(x) \rrbracket_i$
-

schemes from $O(\sqrt[k]{N})$ to $O(\sqrt[2k]{N})$. As reported in Table 10.3, this technique leads to a PRG-based DPF with $O(\sqrt[4]{N})$ key size and an information-theoretic DPF with $O(\sqrt{N})$ key size.

Our adapted DCF While point functions are the product of two point functions, we can decompose comparison functions using three sub-functions (defined over smaller domains): one point function f_a and two comparison functions f_b, f_c . A comparison function f can be expressed as $f(x) = f_a(\gamma) \times f_b(\delta) + f_c(\gamma)$ with

- $f(x) = \beta$ if $x \leq \alpha$, 0 otherwise; with $x \in \{1 \dots N\}$ and $\alpha = \gamma_* \times \lceil \sqrt{N} \rceil + \delta_*$.
- $f_a(\gamma) = \beta$ if $\gamma = \gamma_*$, 0 otherwise; with $\gamma \in \{1 \dots \lceil \sqrt{N} \rceil\}$.
- $f_b(\delta) = 1$ if $\delta \leq \delta_*$, 0 otherwise; with $\delta \in \{1 \dots \lceil \sqrt{N} \rceil\}$.
- $f_c(\gamma) = \beta$ if $\gamma < \gamma_*$, 0 otherwise; with $\gamma \in \{1 \dots \lceil \sqrt{N} \rceil\}$.

As illustrated in Fig. 10.6, the multiplication of f_a and f_b enables to represent the row γ_* containing a segment of β values and a segment of 0 values. Furthermore, f_c covers all the rows before γ_* full of β values.

Like in [BK022], we rely on honest-majority and replicated secret sharing to perform the multiplication of f_a and f_b . Due to space limitations, we cannot include the detailed algorithms. Algorithm 12 details the algorithms for this DCF scheme adapted from [BK022].

Similarly to [BK022], we can apply this technique either on the PRG-based schemes of [BGI15] or on the trivial FSS schemes (i.e., sharing a truth table). The PRG-based DCF has $O(\sqrt[4]{N} \cdot \sqrt{q^{p^m}} \cdot \binom{p-1}{m} \cdot (\lambda + \log q))$ key size, while the information-theoretic DCF has $O(\sqrt{N} \cdot \binom{p-1}{m} \cdot \log q)$ key size.

Since our adapted DCF only adds a sub-DCF (assumed to be secure) compared to the initial DPF, the security proofs can be easily adapted from [BK022].

10.4.2 Our Scheme

We build our DDH-based DCF upon the information-theoretic DPF of [BK022] and the DCF adapted from [BK022]. The main change compared to our DPF scheme is that we require both a sub-DCF and a sub-DPF.

To understand their respective roles, let us represent the function domain as a grid. On the one hand, the DCF covers all rows containing only non-zeros values (i.e., β). On the other hand, the DPF enables to select the row γ_* containing a segment of β and a segment of 0. To complete our extension, we update the definition of the “correction points”, so that the γ_* row contains β until the column δ_* and 0 after.

Algorithm 13 details our scheme (with the changes compared to our DPF in blue). Fig. 10.1 provides a high-level representation of our DCF key.

Algorithm 13 DDH-based DCF scheme (changes compared to our DPF in blue)

-
- 1: Let $\mathbb{G}$ be a cyclic group of prime order q_0 and with g a generator.
 - 2: Let $(\text{Gen}_{\text{DPF}}^{(*)}, \text{Eval}_{\text{DPF}}^{(*)}, \text{Decode}_+)$ be the information-theoretic DPF of [BK022] and $(\text{Gen}_{\text{DCF}}^{(*)}, \text{Eval}_{\text{DCF}}^{(*)}, \text{Decode}_+)$ be the DCF adapted from it.
 - 3: Let $\text{Encode}_{\text{DDH}} : \mathcal{Y} \rightarrow \mathbb{G}$ and $\text{Decode}_{\text{DDH}} : \mathbb{G} \rightarrow \mathcal{Y}$ be two functions such that $\text{Encode}_{\text{DDH}}(0) = g^0$ and $\text{Decode}_{\text{DDH}}(\text{Encode}_{\text{DDH}}(y)) = y$ for any $y \in \mathcal{Y}$.

 - 4: **function** $\text{Gen}_{\text{DPF}}(\alpha, \beta, p, m)$
 - 5: Let $g_\beta \leftarrow \text{Encode}_{\text{DDH}}(\beta)$.
 - 6: Let (γ_*, δ_*) be the position of α in a $(\nu)^k \times \nu$ grid, with $\nu = \lceil \sqrt[3]{N} \rceil$.

 - 7: Sample two (non-zero) $r, s \xleftarrow{R} \mathbb{F}_{q_0}$ and set $r_{\text{inv}} \leftarrow r^{-1}$ and $s_{\text{inv}} \leftarrow s^{-1}$.
 - 8: Generate two sets of DPF keys and one of DCF keys:

$$\begin{aligned} \{k_1^{(a)}, \dots, k_p^{(a)}\} &\leftarrow \text{Gen}_{\text{DPF}}^{(*)}(\gamma_*, r_a, p, m) \\ \{k_1^{(b)}, \dots, k_p^{(b)}\} &\leftarrow \text{Gen}_{\text{DPF}}^{(*)}(\gamma_*, 1, p, m) \\ \{k_1^{(c)}, \dots, k_p^{(c)}\} &\leftarrow \text{Gen}_{\text{DCF}}^{(*)}(\gamma_* - 1, r_c, p, m). \end{aligned}$$

 - 9: **for** $\delta > \delta_*$ **do** Sample a point g_δ from $\mathbb{G}$ and set $h_\delta \leftarrow g_\delta^{-r_{\text{inv}}}$.
 - 10: **for** $\delta \leq \delta_*$ **do** Sample a g_δ from $\mathbb{G}$ and set $h_\delta \leftarrow g_\delta^{-r_{\text{inv}}} \cdot g_\beta^{r_{\text{inv}}}$.
 - 11: Let $u \leftarrow g_\beta^{s_{\text{inv}}}$.
 - 12: Set $k_i = (k_i^{(a)} || k_i^{(b)} || g_1 || h_1 || \dots || g_\nu || h_\nu || u), \forall i \in \{1 \dots p\}$
 - 13: **return** $(k_1, \dots, k_p)$.

 - 14: **function** $\text{Eval}_{\text{DPF}}(k_i, x)$
 - 15: Let (γ', δ') be the position of x in a $(\sqrt[3]{N})^2 \times \sqrt[3]{N}$ grid.
 - 16: Parse $k_i = (k_i^{(a)} || k_i^{(b)} || g_1 || h_1 || \dots || g_\nu || h_\nu || u)$.
 - 17: Let $\llbracket s_a \rrbracket_i \leftarrow \text{Eval}_{\text{DPF}}^{(*)}(k_i^{(a)}, \gamma'), \llbracket s_b \rrbracket_i \leftarrow \text{Eval}_{\text{DPF}}^{(*)}(k_i^{(b)}, \gamma')$
 - 18: Let $\llbracket s_c \rrbracket_i \leftarrow \text{Eval}_{\text{DCF}}^{(*)}(k_i^{(c)}, \gamma')$
 - 19: Let $\llbracket f(x) \rrbracket_i \leftarrow h_{\delta'}^{\llbracket s_a \rrbracket_i} \cdot g_{\delta'}^{\llbracket s_b \rrbracket_i} \cdot u^{\llbracket s_c \rrbracket_i}$.
 - 20: **return** $\llbracket f(x) \rrbracket_i$ $\triangleright \llbracket f(x) \rrbracket_i \in \mathbb{G}$.

 - 21: **function** $\text{Decode}_{\text{DPF}}(\llbracket f(x) \rrbracket_1, \dots, \llbracket f(x) \rrbracket_p)$ **return** $\text{Decode}_{\text{DDH}}(\prod_{i=1}^p \llbracket f(x) \rrbracket_i)$
-

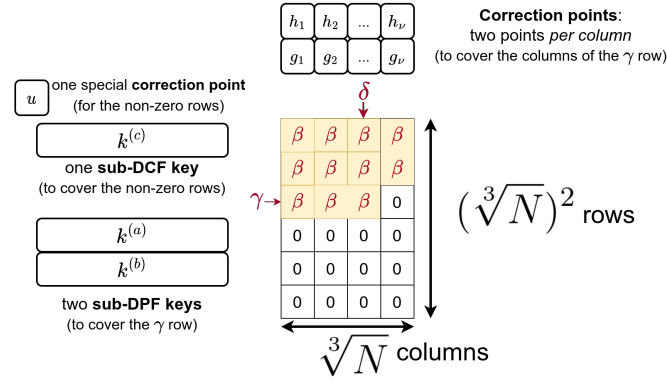


Figure 10.7 / High-level structure of our DDH-based DCF

Scheme	Majority	Assump.	Key size
[BGI15]	Dishonest	PRG	$O(\sqrt{N} \cdot q^{\frac{p-1}{2}} \cdot (\lambda + \log q))$
[KPM24]	Dishonest	DDH + Trusted decoder	$O(\sqrt{N}(2^{-\frac{p-1}{2}} \cdot \lambda + 2^{\frac{p-1}{2}} \cdot \log q))$
Our adaptation PRG-based [BK022]	Honest	PRG	$O(\sqrt[4]{N} \cdot \sqrt{q^{p^m}} \cdot \binom{p-1}{m} \cdot (\lambda + \log q))$
Our adaptation Info.-Th. [BK022]	Honest	None	$O(\sqrt{N} \cdot \binom{p-1}{m} \cdot \log q)$
Our scheme	Honest	DDH	$O(\sqrt[3]{N} \cdot \binom{p-1}{m} \cdot (\lambda + \log q))$

Table 10.8 / Comparison of the multi-party DCF schemes

This scheme is a simple adaptation of our DPF; adding a (secure) sub-DCF and updating the correction points. Thus, its security proof relies on the same arguments as our DPF scheme.

10.4.3 Comparison

Table 10.8 compares our DCF schemes to the existing schemes. The DCF schemes adapted from [BK022] have respectively $O(\sqrt[4]{N} \cdot \sqrt{q^{p^m}} \cdot \binom{p-1}{m} \cdot (\lambda + \log q))$ key size for the PRG-based and $O(\sqrt{N} \cdot \binom{p-1}{m} \cdot \log q)$ key size for the information-theoretic. Finally, our DDH-based scheme has a key size of $O(\sqrt[3]{N} \cdot \binom{p}{m+1} \cdot \log q)$.

We obtain the same conclusions as in multi-party DPF: our scheme provides an intermediary solution between efficient schemes [KPM24, BK022] with $O(\sqrt{N})$ key size and an impractical PRG-based scheme [BK022] with $O(\sqrt[4]{N} q^p)$ key size.

10.5 DDH-based secret encodings and their applications

Our schemes (as well as other DDH-based schemes) need to encode/decode the secret value β in the cyclic group $\mathbb{G}$. This encoding and decoding steps are essential as FSS applications usually involve real-valued or integer secrets that must be mapped to $\mathbb{G}$. Existing papers [CBM15, KPM24] have been vague on the exact way to encode and handle secret in DDH-based schemes.

To have a discussion as concrete as possible, this section uses elliptic curves as cyclic group; $\mathbb{G} = E(\mathbb{F}_q)$. Elliptic curves provide (DDH-hard) cyclic group with prime orders. They were notably used by [CBM15] as they provide smaller key sizes than other DDH-hard cyclic groups.

Contrary to other DPF schemes [BGI15, BK022], DDH-based schemes use multiplication to decode the secret shares. However, using elliptic curves, the multiplication in the cyclic group corresponds to an addition of elliptic curve points. Even though elliptic curves provide a form of additive decoding, this decoding does not provide the same properties as linear share decoding in $\mathbb{F}_q$.

Thus, this section presents two possible DDH-based secret encoding. For each encoding, we present its properties and use cases.

10.5.1 Share compressibility

[BGI15] identified some desirable share properties, in particular: compressibility. According to them, a scheme has compressible shares if we can combine the shares “in a meaningful way” without communication between the shareholders. They focused on the aggregation of multiple shares: $\llbracket f(x_1) \rrbracket \oplus \llbracket f(x_2) \rrbracket = \llbracket f(x_1) + f(x_2) \rrbracket$. Many FSS applications (e.g., PIR [GI14] or private histograms [BBC⁺21]) require a form of compressibility.

Works like [BGI15, BK022] satisfy this property thanks to their additive secret-sharing in $\mathbb{F}_q$. However, “DDH-based” encoding can require a non-linear share decoding, that does not necessarily guarantee compressibility. The next subsections present two DDH-based secret encodings, discuss their relation to compressibility, and list some applications.

10.5.2 Point-based secret encoding

An elliptic curve point P is defined using the coordinates (x, y) . Using the curve formula, we can recover y for any given x .

Point encoding consists in finding $P_\beta = (\beta, y_\beta)$, with $y_\beta \geq 0$. The shares are randomly sampled points $P_1, \dots, P_p \in E(\mathbb{F}_q)$ such that $\sum_i P_i = P_\beta$. The share decoding requires summing the p points and then extracting the first coordinate of the sum result.

However, this encoding *prevents a general share compressibility* because we cannot sum two encoded secrets. This constraint comes from the properties of the elliptic curve addition. Consider two secrets $\beta_1, \beta_2 \in \mathbb{F}_q$ and their respective point-based encodings P_{β_1}, P_{β_2} , let $P_* = P_{\beta_1} + P_{\beta_2} = (x_*, y_*)$. In the general case, we have $x_* \neq \beta_1 + \beta_2$.

Nevertheless, there is an exception to this impossible addition: if one of the shared values is 0. By convention, we encode zero values using the point at infinity $\mathcal{O}$. Hence, we have $P_\beta + P_0 = P_\beta + \mathcal{O} = P_\beta$. Thus, point-based encoding provides a limited compressibility: we can sum multiple function shares only if, *at most* one of them is not null.

DPF Use case Anonymous broadcasters enable several parties to broadcast a message without revealing the message origin. These systems often take the form of public bulletin board in which the writers are anonymous. In recent years, DPFs have become valuable primitives to build efficient anonymous broadcasters like in Riposte [CBM15] and Spectrum [MSD22].

Their DPF-based protocols have the following structure: (1) the servers initialize a large shared bulletin board, (2) each sender i picks at random position α_i in the bulletin board, (3) each sender i uses their message m_i to share a secret point function f_i such that $f_i(x) = m_i$ if $x = \alpha_i$, 0 otherwise, (4) for each shared function $\llbracket f_i \rrbracket$, the servers obtain a shared vector $\llbracket v_i \rrbracket$ such that $\llbracket v_i[j] \rrbracket = \llbracket f_i(j) \rrbracket$, (5) they sum all the vectors v_i with the bulletin board, and reveal the final bulletin board. In other words, the DPF is used as an oblivious write operation on a secret-shared bulletin board.

If at most one sender picks an index, the limited compressibility is sufficient. Spectrum and Riposte proposed solutions to avoid two senders from choosing the same index.

DCF Use case Many recent works [JGB⁺24, KPM24, RTPB22] considered FSS to optimize non-linear operations in privacy-preserving machine learning. For instance, [KPM24] used DCF to perform efficient and secure ReLU operations. This last protocol does not require aggregating multiple shared secrets, so compressibility is not necessary; which makes point-based encoding perfectly acceptable.

10.5.3 Exponent-based secret encoding

This second encoding represents the secret value as an exponent: $P_\beta = \beta \cdot P$, with P , a generator of the curve. The shares are randomly sampled points $P_1, \dots, P_p \in E(\mathbb{F}_q)$ such that $\sum_i P_i = P_\beta$; it requires sampling p exponents $(s_1, \dots, s_p)$ such that $\sum s_i = \beta$. The share decoding consists in summing p elliptic curve points and then computing the discrete logarithm. While this operation is hard in a general case, it is possible if we assume a bounded β (e.g., $\beta < 10^6$). This assumption is realistic in some applications such as private statistics.

This encoding provides compressibility because we can sum two encoded secrets: $P_{\beta_1} + P_{\beta_2} = \beta_1 \cdot P + \beta_2 \cdot P = (\beta_1 + \beta_2) \cdot P = P_{\beta_1 + \beta_2}$. However, it has one drawback: the secret must be small, otherwise the decoding is impractical.

DPF Use case Some works [BBC⁺21,MST24] have used two-party DPFs to build private histograms. In this application, a group of data owners wants to compute the histogram of a certain property (e.g., salary), but each individual wants to keep their value private.

DPF schemes provides a straightforward solution: (1) each individual uses their private value $a_i \in \{1 \dots N\}$ to share a secret point function f_i with $f_i(a_i) = 1$, (2) from each shared function $\llbracket f_i \rrbracket$, the servers can deduce a shared vector $\llbracket v_i \rrbracket$ such that $\llbracket v_i[j] \rrbracket = \llbracket f_i(j) \rrbracket$, (3) The servers aggregate the shared vectors $\llbracket v_i \rrbracket$ and decode the resulting histogram.

This private histogram protocol requires share compressibility because each bin of the histogram usually contains the sum of multiple inputs. Exponent-based encoding is then our only option. The main drawback of exponent-based encoding is that the secret value must be small enough to compute a discrete logarithm, but this assumption is realistic in histogram-making because histogram values number are usually bounded.

DCF Use case [BMR25] recently proposed using DCF to estimate empirical cumulative distribution functions (ECDF) on sensitive data. These statistics are essential notably to evaluate the performances of ML models or to perform some statistical tests [BMR25]. As ECDF are related to histograms, they require the same share compressibility properties. Thus, exponent-based encoding is necessary for such applications, assuming the secret is sufficiently small. Like in private histograms, such a bound usually exists in practice.

Conclusion

This chapter presented a DDH-based approach to build optimized multi-party FSS schemes from existing schemes. In particular, it results into schemes with $O(\sqrt[3]{N})$ key sizes instead of $O(\sqrt{N})$. We applied our technique both on DPF (i.e., FSS for point functions) and DCF (i.e., FSS for comparison functions). Finally, our benchmark highlighted key size reductions up to a factor 10 on realistic problem sizes.

11 Noisy Function Secret Sharing and its applications to Differential Privacy

Although several approaches have been proposed to improve multi-party FSS, a significant efficiency gap remains between the two-party and multi-party settings. The previous chapter successfully improved the DPF state-of-the-art with $O(\sqrt[3]{N})$ key sizes, but the resulting schemes require bounded secret values, which is not always possible in ML applications.

This chapter introduces noisy FSS: a relaxation of FSS preserving the standard privacy guarantees but relaxing the correctness definition by allowing a small amount of noise in the output. We formally define noisy FSS and show how the noise introduced by the scheme can be leveraged to provide differential private outputs in statistics applications.

To demonstrate the benefits of this relaxation, we adapt a scheme proposed by [CBM15]. While their scheme provides the smallest key sizes among multi-party schemes, they do not support some applications notably in statistics due to their non-linear share decoding. On the contrary, recent works such as [GWW25] have larger keys, but support all FSS applications. Our noisy adapted scheme offers the best of both worlds by matching the best key sizes, while providing the properties necessary to statistics applications.

This chapter is drawn from [DMH⁺25].

11.1 Introduction

All existing papers in multi-party DPF have managed to provide asymptotic and/or concrete performance improvements, but there remains a significant gap to obtain key sizes comparable to two- and three-party schemes. This literature overview also shows that there is no single approach providing the best schemes. Ten years after the introduction of FSS, the two most noticeable multi-party schemes are the DDH-based DPF of [CBM15] and the DPF with relaxed privacy of [GWW25]. The former provides the best key size ($O(\sqrt{N})$ for [CBM15] vs. $O(\sqrt{N}p^3)$ for [GWW25]), but has limited applications due to its non-linear decoding.

Applications to statistics DPF has been invented initially to build PIR protocols [GI14]. While DPF has found applications in various domains, recent works have proven its benefits for private statistics protocols such as histograms [BBC⁺21], statistical testing [BMR25], and sparse vector aggregation [AFK⁺25].

While such applications are simple, they cannot be supported by DDH-based schemes due to their non-linear decoding. Indeed, the FSS correctness definition requires that $\text{Decode}(\llbracket f(x) \rrbracket_1, \dots, \llbracket f(x) \rrbracket_p) = f(x)$. However, statistics applications as described above require aggregating shares locally; i.e., we need an additional property: $\text{Decode}(\sum_i \llbracket f_i(x) \rrbracket_1, \dots, \sum_i \llbracket f_i(x) \rrbracket_p) = \sum_i f_i(x)$. Unfortunately, the non-linear decoding does not provide this property, contrary to the linear share decoding used in [BGI15, GWW25].

Works in FSS [CBM15, KPM24] have rarely discussed consequences of this non-linear decoding. On the contrary, we argue that linear decoding is essential for applications that have emerged in privacy-preserving statistics. This observation puts a new light on the literature: even though DDH-based schemes provide the best key size, they are limited to specific applications (contrary to schemes like [GWW25]).

Desirable share properties This perspective based on statistics applications emphasizes an important gap: we should compare FSS schemes on more factors than just the key size. In their seminal work, [BGI15] had introduced *informally* three share properties that would be “*desirable*” to support as many FSS applications as possible. Unfortunately, no work has ever reused these terms.

A first property is “compressibility”: the ability to combine shares “in a meaningful way.” This concept directly relates to the property necessary to statistics applications, share aggregation being one method to combine shares. This property enables to pinpoint the shortcoming of DDH-based schemes: their evaluation shares are not compressible.

Second, they coined the notion of “function-private output.” While the standard FSS privacy definition prevents information leakage from the FSS keys, the function private output prevents leakage from the evaluation shares themselves. For example, let us consider a point function f for which x_0 is not the non-zero point (i.e., $f(x_0) = 0$). The set of shares $\{\llbracket f(x_0) \rrbracket_1, \dots, \llbracket f(x_0) \rrbracket_p\}$ should not reveal information about the non-zero point $f(\alpha) = \beta$. From the adversary perspective, α should be (with equal probability) any point other than x_0 . The absence of this property has an important impact in practice because it requires a trusted party to decode the share as in [KPM24], otherwise the decoding could leak information.

Finally, [BGI15] proposed the concept of “succinctness:” the evaluation shares $\llbracket f(x) \rrbracket$ should have a size “comparable” to the function output $f(x)$.

Even though they are essential to analyze the benefits and drawbacks of schemes such as [GWW25, KPM24], recent works have not reused these properties introduced 10 years ago. An element limiting their use is surely the lack of formalism: [BGI15] have only given informal definitions which prevents formal proof or analysis of these additional properties.

Our contributions To build better schemes, we propose a novel approach: relaxing the correctness definition. While [GWW25] relaxed the privacy definition to optimize [BGI15], we instead allow some noise in the output. Specifically, we introduce the notion of ψ -correctness, a new FSS correctness definition based on stochastic dominance, which bounds the output noise using a distribution ψ .

We call this relaxed variant: noisy FSS. We formalize the definitions, study its properties, and demonstrate its benefits by adapting the DPF from [CBM15] to the noisy setting. The resulting scheme combines the small key size of [CBM15] ($O(\sqrt{N} \log q)$) with the linear decoding of [GWW25]; offering a wide application range. In short, the relaxation scheme leads to a multi-party scheme with the best key size and functionalities.

Finally, we show how to leverage the FSS noise to provide Differential Privacy (DP) guarantees in statistics applications. Thus, noisy FSS provides *by design* stronger privacy guarantees than classic FSS. Indeed, the classic FSS privacy property guarantees that the function remains private during the MPC protocol, but the protocol output (e.g., a histogram) can still leak information about the input data. Thanks to the noise of our scheme, the leakage is inherently mitigated, and this privacy gain is quantified using DP.

Concretely, this chapter contains four main contributions:

- (1) **Introduce noisy Function Secret Sharing** by relaxing the standard FSS correctness definition.
- (2) Demonstrate the benefits of noisy FSS by **adapting the DPF of [CBM15] to the noisy setting**; transforming an efficient scheme incompatible with statistics applications into a noisy scheme compatible with them.
- (3) Detail how to leverage the **FSS noise to provide differential privacy guarantees** in FSS applications to privacy-preserving statistics.
- (4) Formalize the three FSS share properties informally introduced in [BGI15]: succinctness, function-private output, and compressibility.

11.2 Formalizing the desirable share properties

[BGI15] have mentioned some desirable share properties: **compressibility, function-private output, and succinctness**. These properties were only informally presented, because their presence was trivially proven thanks to the linear secret sharing used in [BGI15].

However, in noisy FSS (defined in Section 11.3), their presence is not straightforward (notably function-private output). Thus, this section formalizes these share properties, so we can formally prove them for our scheme.

11.2.1 Succinctness

This first “desirable” property was presented by [BGI15] as follows: a share output by the Eval algorithm is “succinct” if it is “comparable in size to the function output”. This property should not be confused with “compact key sizes:” while FSS works usually want to minimize key size (the Gen output), succinctness refers to the size of the function output shares (the Eval output). For example, sharing a function truth table is a trivial FSS implementation. This trivial scheme has FSS keys as large as the function domain, but the evaluation shares are succinct because they have (individually) the same size as the function output.

We propose to formalize succinctness as follows:

Definition 11.1 (Δ -succinctness). A scheme has Δ -succinct shares if $\log_2(\#\mathbb{G}) \leq \log_2(\#\mathcal{Y}) + \Delta$ with $\mathbb{G}$ the “share space” and $\mathcal{Y}$ the function output space.

The value Δ is the bit-length difference between a function output share and the function output $f(x)$. Existing schemes [BGI15, BGI16, BK022, CBM15] provide 0-succinctness.

Definition 11.1 gives an “additive” formulation to this notion of succinctness. Other formulations based on multiplicative relationships or polynomial dependencies could be possible. However, we prefer this additive formulation because it makes the added cost visible (i.e., the bit-size difference). Moreover, these additional bits have a precise role in noisy FSS: they store the noise. Thus, this additive definition resonates with Fully Homomorphic Encryption schemes [CLT14] in which low-order bits are also dedicated to noise handling.

Note that any scheme (even poorly efficient) can satisfy Δ -succinctness for a sufficiently large Δ . A scheme is considered succinct if it satisfies it for a “sufficiently” small Δ . For instance, Garbled Circuits can be used to construct a simple two-party FSS scheme [BGI15]: given a function f , the first key consists of the garbled circuit representing x , while the second key contains the information needed to map each input x to the corresponding garbled input labels. However, this scheme is not considered succinct by [BGI15] because its output shares have a size of $O(L)$ for a function with a circuit size of L .

Remark 11.2 (Quantized functions). Certain FSS applications involve real values (e.g., privacy-preserving ML [KPM24, RTPB22]), but the use of cryptographic tools in such protocols commonly require quantizing these values. Our definitions consider $\mathcal{Y}$ to be the output space of the quantized function. ┘

11.2.2 Function-Private Output (FPO)

While the FSS privacy definition (Definition 2.5) prevents information leakage from the keys, the shares obtained using Eval can also reveal unintended information. During FSS-based protocols (like Algorithm 1), parties sometimes

exchange evaluation shares, but these shares may leak information. To prevent this leakage, [BGI15] introduced the term *function-private output*. Informally, a scheme provides a *function-private output* if, given the evaluation shares $\{\llbracket f(x) \rrbracket_1, \dots, \llbracket f(x) \rrbracket_p\}$, an adversary learns no more information than the value $f(x)$.

To formalize this concept, we extend Definition 2.5. The idea is to secretly evaluate a function f at an arbitrary number of points $x_1, \dots, x_n$ in the function domain $\mathcal{X}$. The scheme has function-private output if the evaluation shares for these points can be simulated using only the $x_1, \dots, x_n$, their evaluations $f(x_1), \dots, f(x_n)$, the function family $\mathcal{F}$, and λ .

This expresses the intuition that the evaluation shares reveal no additional information about f beyond what can already be inferred from its evaluations on the chosen points $f(x_1), \dots, f(x_n)$, and the general structure of $\mathcal{F}$. In particular, they do not reveal any information about $f(x)$ for any $x \notin \{x_1, \dots, x_n\}$. It is important that the simulator has access to the function family $\mathcal{F}$: for example, if $\mathcal{F}$ is the class of point functions and $\alpha \in \{x_1, \dots, x_n\}$, then it is trivial to deduce that $f(x) = 0$ for all the remaining elements x of the domain.

Definition 11.3 (Function-Private Output). Let $\mathcal{F}$ be a set of functions with a common domain $\mathcal{X}$. An FSS scheme $\mathcal{E} = (\text{Gen}, \text{Eval}, \text{Decode})$ for $\mathcal{F}$ has function-private output (FPO) if the evaluation shares of any subset of $\mathcal{X}$ are simulatable.

More formally, given a semi-honest adversary who corrupts a subset of the parties $A \subset \{1, \dots, p\}$, there exists a PPTA $\mathcal{S}$ such that for all PPTA $\mathcal{D}$, there exists a negligible function negl , such that for all functions $f \in \mathcal{F}$ (and their corresponding descriptions $\hat{f} \in P_{\mathcal{F}}$), for all subsets $X = \{x_1, \dots, x_n\} \subset \mathcal{X}$, and for all security parameters $\lambda \in \mathbb{N}$, we execute the generation algorithm $(k_1, \dots, k_p) \leftarrow \text{Gen}(1^\lambda, f, p)$, and the evaluation and decoding for each point in $x_j \in X$, namely

$$\begin{aligned} y_{i,j} &\leftarrow \text{Eval}(k_i, x_j) \quad \forall i \in \{1, \dots, p\} \\ y_j &\leftarrow \text{Decode}(y_{1,j}, \dots, y_{p,j}) \end{aligned}$$

and require that

$$\begin{aligned} &|\Pr[\mathcal{D}((\text{realView}, \text{realOutput}), f, \{x_1, \dots, x_n\}) = 1] \\ &\quad - \Pr[\mathcal{D}((\text{simView}, \text{idealOutput}), f, \{x_1, \dots, x_n\}) = 1]| \\ &\leq \text{negl}(\lambda) \end{aligned}$$

where

$$\begin{aligned} \text{realView} &:= \{k_i\}_{i \in A}, \{y_{i,j}\}_{i \in A, j \in \{1, \dots, n\}}, \\ &\quad \{(x_1, y_1), \dots, (x_n, y_n)\} \\ \text{realOutput} &:= \{(x_1, y_1), \dots, (x_n, y_n)\} \\ \text{simView} &:= \mathcal{S}(\lambda, \mathcal{F}, \{(x_1, f(x_1)), \dots, (x_n, f(x_n))\}) \\ \text{idealOutput} &:= \{(x_1, f(x_1)), \dots, (x_n, f(x_n))\} . \end{aligned}$$

When $n = 0$, Definition 11.3 corresponds to Definition 2.5. Indeed, the classic privacy definition of Definition 2.5 only prevents the leakage at key generation time, while the function-private output of Definition 11.3 also prevents leakage during the function evaluation. Thus, function-private output can be interpreted as an “ultimate” form of privacy in FSS.

Function-private output is important in practice because FSS applications such as in privacy-preserving ML [KPM24] require revealing specific points of the function. However, the adversary should not discover more than the expected points. Recently, [KPM24] introduced an FSS scheme requiring a *trusted decoder* in order to guarantee the function privacy. Other existing schemes verify naturally this concept, notably thanks to the use of linear secret sharing [BGI15].

11.2.3 Compressibility

Finally, [BGI15] mentioned a last share property: compressibility. They defined it as the possibility to combine shares “in a meaningful way.” This informal definition captures many possible scenarios from the simple aggregation of multiple shares to complicated non-linear functions of the shares. However, this informality complicates the analysis and comparison of FSS schemes.

We propose to specify this compressibility definition. Our formal definition focuses on “linear compressibility” because all existing works required summing function output shares. We can then formalize this property as follows:

Definition 11.4 (Linear Compressibility). Let $\mathcal{E} = (\text{Gen}, \text{Eval}, \text{Decode})$ be an FSS scheme for the function family $\mathcal{F}$. The scheme $\mathcal{E}$ provides linear compressibility if for any number of functions $f_1, \dots, f_m \in \mathcal{F}$ (and their corresponding descriptions $\hat{f}_1, \dots, \hat{f}_m \in P_{\mathcal{F}}$), corresponding number of elements $x^{(1)}, \dots, x^{(n_f)} \in \mathcal{X}$, and $\lambda \in \mathbb{N}$, the following condition holds

$$\Pr \left[\text{Decode} \left(\sum_{j=1}^{n_f} \text{Eval}(k_1^{(j)}, x^{(j)}), \dots, \sum_{j=1}^{n_f} \text{Eval}(k_p^{(j)}, x^{(j)}) \right) = \sum_{j=1}^{n_f} f_j(x^{(j)}) \right] = 1$$

where $(k_1^{(j)}, \dots, k_p^{(j)}) \leftarrow \text{Gen}(\hat{f}_j, 1^\lambda)$ for all $j \in \{1, \dots, n_f\}$.

Informally, if $\llbracket f(x) \rrbracket_i = \text{Eval}(k_i^{(f)}, x)$ is a share of $f(x)$ and $\llbracket g(x') \rrbracket_i = \text{Eval}(k_i^{(g)}, x')$, we have $\llbracket f(x) \rrbracket_i + \llbracket g(x') \rrbracket_i = \llbracket f(x) + g(x') \rrbracket_i$ (i.e., a share of $f(x) + g(x')$).

Moreover, we can comment a few details from Definition 11.4. First, this definition assumes that the functions $f_1, \dots, f_m$ come from the same function family $\mathcal{F}$. This definition could be generalized to multiple function families (possibly with different domains) *if the FSS schemes use the same decoding algorithms*. For example, with additive secret shares, we can trivially aggregate additive shares $y_i^{(f)}$ from a point function f with additive shares $y_i^{(g)}$ from a comparison function g . Second, two functions $f_j, f_{j'}$ from $f_1, \dots, f_m$ can be equal. Such equality would cover applications in which we want to aggregate points from the same function. For example, DPF-based PIR [GI14] usually evaluates a function on its whole function domain and then aggregate all the shares.

Schemes based on linear secret sharing naturally satisfy this property [BGI15]. However, DDH-based schemes only have a constrained homomorphism as explained in Section 10.5.

11.3 Noisy Function Secret Sharing

11.3.1 Core noisy definitions

Noisy FSS schemes have exactly the same algorithms as classic FSS schemes: Gen, Eval, Decode. The privacy definition remains also unchanged: the adversary should not be able to extract information from m out of p keys. The only difference is the correctness definition: Decode outputs $f(x) + \text{err}$ (with err a noise) instead of $f(x)$.

Noisy definitions in cryptography Noisy FSS is not the first noisy cryptographic primitive [BLMR13, ZAS24]. For instance, [BLMR13] introduced the notion of “almost” seed-homomorphic Pseudo-Random Generators (PRG):

Definition 11.5. A PRG G is γ -almost seed-homomorphic if

$$G(s_1) + G(s_2) = G(s_1 + s_2) + \text{err} \text{ with } \text{err} \in [-\gamma, \gamma] \text{ a small bounded error.}$$

Towards general definitions Fixing bounds like in [BLMR13] provides an easily understandable definition, but it is limiting. For example, Gaussian noises are unbounded. For our definitions, we want to adopt a more generic approach.

To build our noisy correctness definition, we then rely on “stochastic dominance” [HR69, SS07], a partial order between random variables. Let $\succsim$ refer to this partial order:

Definition 11.6 (Stochastic dominance). Given two random variables A, B ,

we say that A has first-order stochastic dominance over B if

$$\Pr[A \geq t] \geq \Pr[B \geq t] \quad \forall t$$

and we denote it by $A \succcurlyeq B$ or equivalently $B \preccurlyeq A$.

To better understand this stochastic order $\succcurlyeq$, we can observe a consequence of stochastic dominance: $A \succcurlyeq B \Rightarrow \mathbb{E}[f(A)] \geq \mathbb{E}[f(B)]$, for all monotone function f . In particular, if f is the identity function, $A \succcurlyeq B \Rightarrow \mathbb{E}[A] \geq \mathbb{E}[B]$; i.e., A takes *in average* higher values than B .

Based on this order, we propose the following “noisy correctness” definition:

Definition 11.7 (ψ -noisy correctness). Let $\mathcal{E} = (\text{Gen}, \text{Eval}, \text{Decode})$ be an FSS scheme for the function family $\mathcal{F}$, with common domain $\mathcal{X}$. For any function $f \in \mathcal{F}$ (and its corresponding description $\hat{f} \in P_{\mathcal{F}}$), $x \in \mathcal{X}$, and $\lambda \in \mathbb{N}$, we define the random variable

$$\begin{aligned} \mathcal{D}_{f,x}^\lambda &:= \text{Decode}(\text{Eval}(k_1, x), \dots, \text{Eval}(k_p, x)) \\ &\text{with } (k_1, \dots, k_p) \leftarrow \text{Gen}(\hat{f}, 1^\lambda). \end{aligned}$$

Let ψ be an error distribution. We say that $\mathcal{E}$ satisfies ψ -noisy correctness (or simply is ψ -noisy) if for all $f \in \mathcal{F}$, $x \in \mathcal{X}$, and $\lambda \in \mathbb{N}$:

$$(\mathcal{D}_{f,x}^\lambda - f(x))^2 \preccurlyeq \psi^2.$$

Thanks to the square exponent, this definition bounds *symmetrically* the distribution of $\mathcal{D}_{f,x}^\lambda$ based on an error distribution ψ . We could rephrase this definition as follows: the distribution $\mathcal{D}_{f,x}^\lambda$ is more “concentrated” (i.e., less noisy) than the error distribution ψ .

The stochastic dominance used in Definition 11.7 provides a more holistic definition than fixed bounds because we bound the error based on the error distribution ψ . On the one hand, if the error distribution is naturally bounded, we can naively consider these bounds for our error term. On the other hand, if the error distribution is unbounded (or loosely bounded), we can deduce bounds holding with high probability. For example, a Gaussian noise is unbounded, but we can set bounds that a Gaussian noise will not exceed (with high probability).

Moreover, we can consider the influence of the number of parties, p , on the noise distribution ψ . Ideally, one may like to make this dependence explicit; such as by including a linear factor in p directly in the Definition 11.7. However, we use a more generic formulation to ensure compatibility with a broad range of noisy schemes. While the noise distribution is typically influenced by p , the relationship is not necessarily linear. For instance, in our construction (described in Section 11.4),

the noise variance includes a factor of $(p + 1)$. Defining ψ generically provides greater flexibility, enabling researchers to develop schemes that rely on a wide variety of (potentially unforeseen) noise distributions.

Remark 11.8 (Fixed distribution). An FSS scheme $\mathcal{E}$ for the function family $\mathcal{F}$ (with domain $\mathcal{X}$) is ψ -noisy correct if

$$(\mathcal{D}_{f,x}^\lambda - f(x)) \sim \psi \quad (\forall f \in \mathcal{F}, \forall x \in \mathcal{X}). \quad \lrcorner$$

This remark provides an intuitive understanding of noisy correctness: the decoding output follows a reference probability distribution. While this version is easier to understand, it is not as general as Definition 11.7. Definition 11.7 only bounds the noise distribution, so the noise distribution can be “smaller” than ψ on some specific points $x \in \mathcal{X}$ or functions $f \in \mathcal{F}$; which is the case in our schemes. Remark 11.8 is more rigid and imposes the noise distribution to be exactly the same for all functions and points.

For any arbitrary algorithm, we could find a random variable ψ such that the algorithm satisfies Definition 11.7. This “tautological” satisfaction was also present in the γ -almost seed-homomorphic PRGs [BLMR13] as their correctness definition could be satisfied by any algorithm if γ is large. However, the goal of such noisy definitions is to be satisfied for “useful” noise distributions. In our case, we can notably deduce a remark for Gaussian error distributions:

Remark 11.9 (Gaussian noise). An FSS scheme $\mathcal{E}$ for the function family $\mathcal{F}$ (with domain $\mathcal{X}$) is ψ -noisy with $\psi \sim \mathcal{N}(0, \sigma^2)$ if

$$\Pr[(\mathcal{D}_{f,x}^\lambda - f(x))^2 \geq t] \leq 2 \cdot \Phi\left(\frac{\sqrt{t}}{\sigma}\right) - 1, \quad \forall t \geq 0$$

where Φ is the Gaussian cumulative distribution function. $\lrcorner$

Remark 11.8 leads us to another interesting observation:

Remark 11.10. “Classic” FSS schemes satisfy ψ -noisy FSS schemes for any error distribution ψ . $\lrcorner$

Indeed, FSS schemes satisfying Definition 2.3 also satisfy Definition 11.7 because their corresponding variable “ $(\mathcal{D}_{f,x}^\lambda - f(x))^2$ ” follows a Dirac distribution on 0. In other words, noisy correctness generalizes the correctness definition commonly used in existing works.

Noisy secret sharing Based on our noisy FSS definitions, we can naturally define the notion of noisy secret sharing. However, we have not identified any specific applications or efficiency gain provided by noisy secret sharing, contrary to noisy FSS for which this chapter presents schemes improving the state of the art.

Talwar [Tal22] has proposed “differential” secret shares enabling to share secret real numbers based on differential privacy noise. Even though Talwar uses noise in their secret sharing scheme, our noisy definitions take a completely orthogonal

approach. Talwar’s “differential” shares rely on a relaxed privacy definition compared to traditional secret shares (and maintain the correctness definition). On the contrary, our noisy secret sharing relax the correctness definition and keep the same privacy definition as classic FSS.

To frame our difference with [Tal22], we must look at the motivations behind each construction. On the one hand, differential secret sharing generalizes the notion of secret sharing to real values. On the other hand, noisy secret sharing tolerates noise to scale existing (function) secret sharing schemes. Like classic secret sharing schemes, our definitions rely on finite fields. Thus, noisy FSS is a new concept in the literature.

11.3.2 Noisy share properties

Noisy FSS schemes do not require relaxing the succinctness definition. However, we need to generalize the notion of function-private output (given in Definition 11.3). In the “classic” FPO definition, the ideal functionality is deterministic: it returns exact evaluations of the function f at the points $x_1, \dots, x_n$. This deterministic functionality is incompatible with noisy FSS, so we transform it into a probabilistic functionality: it now outputs random variables whose distributions are centered around the expected value $f(x) + e$, where $e \sim \psi$ for some error distribution ψ :

Definition 11.11 (ψ -Noisy Function-Private Output). Let $\mathcal{F}$ be a set of functions with a common domain $\mathcal{X}$ and $\psi = \{\psi_n\}_{n \in \mathbb{N}}$ be a probability ensemble, where each ψ_n has zero mean. An FSS scheme $\mathcal{E} = (\text{Gen}, \text{Eval}, \text{Decode})$ for $\mathcal{F}$ has ψ -noisy FPO if the evaluation shares of any subset of $\mathcal{X}$ are simulatable, up to an error ψ_λ (depending on the security parameter λ).

The formalization is the same as in Definition 11.3, but now

$$\text{idealOutput} := \{(x_1, f(x_1) + e_1), \dots, (x_n, f(x_n) + e_n)\}$$

with $e_1, \dots, e_n \sim \psi_\lambda$.

Remark 11.12 (Definition 11.3 $\Rightarrow$ Definition 11.11). The deterministic FPO definition is an instantiation of the noisy FPO definition for $\psi \sim \delta_0$, the Dirac distribution on 0. $\dashv$

While classic FSS schemes based on linear secret sharing inherently satisfy the FPO property [BGI15], noisy schemes do not necessarily offer the same guarantee. If the noise distribution is correlated with the secret function, the output may leak information. Section 11.4 shows how to use “noise flooding” [DDE⁺23] to guarantee FPO in our scheme adapted from [CBM15].

Finally, the noisy correctness requires relaxing the linear compressibility. To do so, we can combine noisy correctness (Definition 11.7) with the linear compressibility (Definition 11.4):

Definition 11.13 (ψ_* -noisy linear compressibility). Let $\mathcal{E} = (\text{Gen}, \text{Eval}, \text{Decode})$ be an FSS scheme for the function family $\mathcal{F}$, with common domain $\mathcal{X}$. For $f_1, \dots, f_{n_f} \in \mathcal{F}$ (and their descriptions $\hat{f} \in P_{\mathcal{F}}$), $x^{(1)}, \dots, x^{(n_f)} \in \mathcal{X}$, and $\lambda \in \mathbb{N}$, we define the random variable

$$\mathcal{D}_{f_1, x^{(1)}, \dots, f_{n_f}, x^{(n_f)}}^\lambda := \text{Decode} \left(\sum_{j=1}^{n_f} \text{Eval}(k_1^{(j)}, x^{(j)}), \dots, \sum_{j=1}^{n_f} \text{Eval}(k_p^{(j)}, x^{(j)}) \right)$$

where $(k_1^{(j)}, \dots, k_p^{(j)}) \leftarrow \text{Gen}(\hat{f}_j, 1^\lambda)$ for all $j \in \{1, \dots, n_f\}$.

Let ψ be an error distribution over $\mathcal{X}$. We say that $\mathcal{E}$ provides ψ_* -noisy linear compressibility if for any number of functions $f_1, \dots, f_{n_f} \in \mathcal{F}$, corresponding number of elements $x^{(1)}, \dots, x^{(n_f)} \in \mathcal{X}$, and $\lambda \in \mathbb{N}$, the following condition holds

$$\left(\mathcal{D}_{f_1, x^{(1)}, \dots, f_{n_f}, x^{(n_f)}}^\lambda - \sum_{j=1}^{n_f} f_j(x^{(j)}) \right)^2 \preceq \psi_*^2 .$$

We can also prove that noisy correctness implies noisy linear compressibility if a scheme has a linear share decoding:

Theorem 11.14. (Linear decoding + noisy correctness $\Rightarrow$ noisy compressibility)

Let $\mathcal{E} = (\text{Gen}, \text{Eval}, \text{Decode})$ be an FSS scheme for the function family $\mathcal{F}$.

If $\text{Decode}(y_1, \dots, y_p) = \sum_{i=1}^p y_i$ and $\mathcal{E}$ is ψ -noisy correct, then $\mathcal{E}$ provides ψ_* -noisy linear compressibility,

with ψ_* s.t. $\Pr[\psi_*^2 \geq t] = \min\{1, n_f \Pr[\psi^2 \geq t/n_f^2]\}, \forall t$.

Proof. First, observe that: $\mathcal{D}_{f_1, x^{(1)}, \dots, f_{n_f}, x^{(n_f)}}^\lambda - \sum_{j=1}^{n_f} f_j(x^{(j)})$

$$\begin{aligned}
 &= \text{Decode} \left(\sum_{j=1}^{n_f} \text{Eval}(k_1^{(j)}, x^{(j)}), \dots, \sum_{j=1}^{n_f} \text{Eval}(k_p^{(j)}, x^{(j)}) \right) \\
 &\quad - \sum_{j=1}^{n_f} f_j(x^{(j)}) \\
 &= \sum_{i=0}^p \left(\sum_{j=1}^{n_f} \text{Eval}(k_i^{(j)}, x^{(j)}) \right) - \sum_{j=1}^{n_f} f_j(x^{(j)}) \\
 &= \sum_{j=1}^{n_f} \left(\sum_{i=0}^p (\text{Eval}(k_i^{(j)}, x^{(j)}) - f_j(x^{(j)})) \right)
 \end{aligned}$$

Reusing the notation from Definition 11.7:

$$\begin{aligned}
 \mathcal{D}_{f_j, x^{(j)}}^\lambda &:= \sum_{i=1}^p \text{Eval}(k_i^{(j)}, x^{(j)}) \\
 \Rightarrow \mathcal{D}_{f_1, x^{(1)}, \dots, f_{n_f}, x^{(n_f)}}^\lambda - \sum_{j=1}^{n_f} f_j(x^{(j)}) \\
 &= \sum_{j=1}^{n_f} \left(\mathcal{D}_{f_j, x^{(j)}}^\lambda - f_j(x^{(j)}) \right)
 \end{aligned}$$

For convenience, we define the random variable $\mathcal{D}_j^- = \left(\mathcal{D}_{f_j, x^{(j)}}^\lambda - f_j(x^{(j)}) \right)$. Thus, we want to prove that $\left(\sum_{j=1}^{n_f} \mathcal{D}_j^- \right)^2 \preceq (\psi_*)^2$.

Note that we have:

$$\begin{aligned}
 &\text{For any sequence } \{d_1, \dots, d_{n_f}\}, \\
 &\left(\sum_{j=1}^{n_f} d_j \right)^2 \leq \left(n_f \cdot \max_j |d_j| \right)^2 = n_f^2 \max_j \left(d_j \right)^2 \\
 &\Rightarrow \left\{ \left(\sum_{j=1}^{n_f} \mathcal{D}_j^- \right)^2 \geq t \right\} \subseteq \left\{ \max_j \left(\mathcal{D}_j^- \right)^2 \geq t/n_f^2 \right\} \\
 &\quad \subseteq \bigcup_{j=1}^{n_f} \left\{ \left(\mathcal{D}_j^- \right)^2 \geq t/n_f^2 \right\}
 \end{aligned}$$

By the union bound:

$$\Pr \left[\left(\sum_{j=1}^{n_f} \mathcal{D}_j^- \right)^2 \geq t \right] \leq \sum_{j=1}^{n_f} \Pr \left[\left(\mathcal{D}_j^- \right)^2 \geq t/n_f^2 \right] \quad (11.15)$$

By assumption, the scheme is ψ -noisy

$$\Rightarrow \forall j, \left(\mathcal{D}_j^- \right)^2 \preceq (\psi)^2 \quad (11.16)$$

$$\iff \forall j, \forall t, \Pr \left[\left(\mathcal{D}_j^- \right)^2 \geq t \right] \leq \Pr[(\psi)^2 \geq t] \quad (11.17)$$

By combining Equations (11.15) and (11.17):

$$\Pr \left[\left(\sum_{j=1}^{n_f} \mathcal{D}_j^- \right)^2 \geq t \right] \leq n_f \Pr[(\psi)^2 \geq t/n_f^2] \quad (11.18)$$

By definition, ψ_* is such that $\Pr[\psi_*^2 \geq t] = \min\{1, n_f \Pr[\psi^2 \geq t/n_f^2]\}, \forall t$. By injecting this into Equation (11.18), we obtain:

$$\begin{aligned} \Pr \left[\left(\sum_{j=1}^{n_f} \mathcal{D}_j^- \right)^2 \geq t \right] &\leq \Pr[\psi_*^2 \geq t] \\ \iff \left(\sum_{j=1}^{n_f} \mathcal{D}_j^- \right)^2 &\preceq (\psi_*)^2 \end{aligned}$$

The scheme $\mathcal{E}$ is then ψ_* -noisy linear compressible. $\square$

Theorem 11.14 illustrates that the noise amplitude naturally increases when we aggregate multiple noisy shares. The theorem provides a first bound for the noise distribution induced by the aggregation of multiple noisy function shares. Tighter bounds could be possible by taking into account the property of ψ (for a given ψ -correct scheme). We leave the analysis of tighter bounds for future work.

Similarly to our noisy correctness definition, we use a generic noise distribution ψ_* in Definition 11.13, without explicitly specifying the influence of p and n_f . As in the case of noisy correctness, these parameters naturally affect the noise, but their impact is not necessarily linear or immediately apparent. Theorem 11.14 illustrates this well, presenting a non-linear relationship. Therefore, we once again choose a generic form for ψ_* to preserve flexibility for potential future schemes.

11.3.3 Handling the noise in statistics applications

This relaxed definition of FSS naturally raises the question: how to handle the noise in FSS applications? We identify two main strategies for handling the noise: mitigating its impact through a *scale-and-truncate* approach, or leveraging it as a source of *differential privacy*.

Scale-and-truncate Noise truncation/rounding is a straightforward solution: we scale the secret function such that $f(x) \gg \text{err}$, so the noise can be truncated from the final result. In other words, we reserve the low-order bits to the error (which is classic in applications of noisy cryptographic primitives [BLMR13]).

While this approach is unusable on historical FSS applications such as PIR (because the noise order would be proportional to the database size), it is compatible with the applications to statistics. Such applications usually require the aggregation of n functions $f_1, \dots, f_n$, so we would encode these functions such that $f_i(x) \gg n \cdot \text{err}$.

This approach has one main advantage: the output will not be noised (i.e., same correctness as classic FSS). However, the values must be scaled significantly to ensure that the noise remains much smaller than the secrets. This scaling will degrade the evaluation share succinctness because the bit size must be increased to enable the scaling.

Differential privacy (DP) The second option is to keep the noise and use it to provide DP guarantees. The FSS privacy definitions (Definitions 2.5 and 11.3) only protect the information leakage from the Eval and Gen outputs. However, the output of Decode can still leak information about the inputs of an FSS-based protocol (e.g., a private histogram-making). DP has emerged to mitigate such information leakage from public outputs.

By keeping the noise, we provide better privacy guarantees than classic FSS: the public output of FSS-based protocols also offers significant privacy guarantees. Such privacy guarantees are particularly valuable in statistics applications for which DP has become an essential privacy-enhancing technique [SU21]. Thus, Section 11.5 shows how to integrate the FSS noise into the DP analysis of such applications.

11.4 Noisy DPF scheme

This section adapts the DDH-based scheme described in [CBM15] to the noisy setting. This adaption relies on the Learning with Error (LWE) hardness assumption in order to replace the DDH-based components present in [CBM15]. This adapted schemes provides the state-of-the-art key size of [CBM15], while also offering the compressibility property absent from [CBM15] but necessary to statistics applications.

11.4.1 LWE-based pseudo-random generator (PRG)

First, let us introduce the LWE (decision) problem [Reg09]:

Definition 11.19 (Decision Learning with Errors). Let $q \geq 2$ be a modulus, n_{LWE} be the dimension, m_{LWE} be the number of samples, and χ be an error

distribution (typically Gaussian) over $\mathbb{F}_q$. Let $A \in \mathbb{F}_q^{n_{\text{LWE}} \times m_{\text{LWE}}}$ be a random public matrix. Let $s \in \mathbb{F}_q^{m_{\text{LWE}}}$ be a secret vector, and $e \xleftarrow{R} \chi^{m_{\text{LWE}}}$ be an error vector.

An adversary solves the Decision-LWE problem if they can distinguish with non-negligible probability $(A, As + e)$ from (A, u) , with u a vector uniformly sampled from $\mathbb{F}_q^{m_{\text{LWE}}}$.

From LWE, we can build a PRG $G_{\text{LWE}} : \mathbb{F}_q^{m_{\text{LWE}}} \rightarrow \mathbb{F}_q^{n_{\text{LWE}}}$ s.t. $G_{\text{LWE}}(s) = As + e$, for any seed $s \in \mathbb{F}_q^{m_{\text{LWE}}}$, and A a public matrix (randomly sampled). Note that this LWE-based PRG [BLMR13] is “almost” seed-homomorphic because $G_{\text{LWE}}(s_1) + G_{\text{LWE}}(s_2) = G_{\text{LWE}}(s_1 + s_2) + \text{err}$, with err an error term.

For simplicity, our scheme relies solely on the basic LWE-based PRG G_{LWE} , but other almost seed-homomorphic PRGs are also good foundations to build noisy FSS schemes.

11.4.2 Scheme

To build a first noisy DPF, we modify the scheme of [CBM15]. As their scheme is built upon a DDH-based PRG (i.e., another form of seed-homomorphic PRG), the main change is to replace this PRG with an LWE-based PRG. We present our adapted scheme in Algorithm 14.

Note that the LWE-based PRG G_{LWE} is an essential component of our scheme. On the one hand, it provides the pseudorandomness necessary to prove the scheme privacy (Theorem 11.20). On the other hand, it provides the seed homomorphism necessary to the scheme correctness (Theorem 11.22).

To build their scheme, [CBM15] relied on elliptic curves because they provide the only known *exact* seed-homomorphic PRG. The other *known* seed-homomorphic PRGs are all based on LWE (or LWE variants) [BLMR13], but they are *almost* seed-homomorphic (i.e., noisy). In [CBM15], the authors could not consider noisy PRGs because they were no definitions or applications allowing for noise in FSS schemes. As this chapter formalizes noisy FSS and provides guidelines for its use in concrete applications, the adaptation of [CBM15] to the noisy setting is now possible.

Besides the LWE-based PRG, our scheme has another key difference compared to [CBM15]: Line 20 of Algorithm 14. At the end of the Eval_χ , the evaluator adds a noise drawn from the distribution χ . This noise is sampled using fresh randomness at each evaluation (i.e., not deterministically from a PRG with a fixed seed contained in the key). Thus, the evaluation has two noise sources: the LWE-based PRG, and the χ noise.

At this stage, this additional noise could seem artificial and useless. Indeed, one may think that the LWE-based PRG would be sufficient to satisfy all privacy

Algorithm 14 Dishonest-majority noisy DPF scheme

-
- 1: Let $\nu \leftarrow \lceil \sqrt{N} \rceil$
 - 2: Represent the function domain $\{1 \dots N\}$ as a $(\nu \times \nu)$ grid.
 - 3: Let $G_{LWE} : \mathbb{F}_q^{m_{LWE}} \rightarrow \mathbb{F}_q^\nu$ be an LWE-based PRG using the Gaussian distribution $\backslash_{LWE}(0, \sigma_{LWE}^2)$ as error distribution.

▷ NB: $m_{LWE} = O(\lambda)$
 - 4: **function** Gen($\lambda, (\alpha, \beta), p$)
 - 5: Parse α as (α_l, α_r) (i.e., position in the $\nu \times \nu$ grid)
 - 6: Sample random vectors $b_1, \dots, b_{p-1}$ from $(\mathbb{F}_q)^\nu$
 - 7: Sample vectors of seeds $s_1, \dots, s_{p-1}$ from $(\mathbb{F}_q^{m_{LWE}})^\nu$
 - 8: Sample a single seed s_* from $\mathbb{F}_q^{m_{LWE}}$
 - 9: Set $b_p \in (\mathbb{F}_q)^\nu$ such that $\sum_{i=1}^p b_i = e_{\alpha_l}$

▷ e_x is a vector equal to 1 on index x and 0 otherwise
 - 10: Set $s_p \in (\mathbb{F}_q^{m_{LWE}})^\nu$ such that $\sum_{i=1}^p s_i = s_* \cdot e_{\alpha_r}$
 - 11: Let $v \leftarrow \beta \cdot e_{\alpha_r} - G_{LWE}(s_*)$
 - 12: Build the DPF keys $k_1, \dots, k_p$ as follows:

$k_i = (b_i, s_i, v)$
 - 13: **return** $(k_1, \dots, k_p)$
 - 14: **function** Eval $_\chi(k_i, x)$
 - 15: Parse k_i as (b_i, s_i, v)
 - 16: Parse x as (x_l, x_r) (i.e., position in the $\nu \times \nu$ grid)
 - 17: Expand the x_l -th seed: $g_i \leftarrow G_{LWE}(s_i[x_l])$
 - 18: Let $y_i \leftarrow g_i[x_r] + b_i[x_l]v[x_r]$
 - 19: Sample an error $\text{err} \in \mathbb{F}_q$ using the distribution χ

▷ The error err is sampled using fresh randomness.
 - 20: Let $y_i \leftarrow y_i + \text{err}$ ▷ To have a function-private output
 - 21: **return** y_i
 - 22: **function** Decode($y_1, \dots, y_p$)
 - 23: **return** $\sum_{i=1}^p y_i$
-

properties, but this is untrue. The LWE noise is sufficient to satisfy the standard FSS privacy. However, the supplementary χ noise is necessary to also satisfy function-private output (see Theorem 11.24).

Our key size is $O(\sqrt{N}(2 + \lambda) \log q)$. Algorithm 14 represents the function domain $\{1, \dots, N\}$ as a $\sqrt{N} \times \sqrt{N}$ grid, but we can slightly optimize the key size with an asymmetric grid $N_1 \times N_2$, with $N_1 = \sqrt{N}(1 + \lambda)^{-1/2}$ and $N_2 = \sqrt{N}(1 + \lambda)^{1/2}$. The resulting key size is $\sqrt{N(1 + \lambda)} 2 \log q$. [BGI15] used a similar approach to optimize their key size.

Setting the LWE parameter Our key size is proportional to the security parameter λ . However, this parameter is not directly visible in the LWE definition (Definition 11.19). LWE security depends on $n_{\text{LWE}}, m_{\text{LWE}}, q, \chi$. The parameters n_{LWE} and q are fixed by the application: n_{LWE} is the size of the PRG output ($\lceil \sqrt{N} \rceil$ in our case), and $\mathbb{F}_q$ is the function output space.

We must tune m_{LWE} and χ to ensure security. However, we commonly pick the error distribution χ minimizing the error amplitude, so the noise can be handled by the downstream applications. In this chapter, we use Gaussian distributions as error distributions, with σ_{LWE}^2 , their variance.

Thus, our security is conditioned by m_{LWE} : the larger m_{LWE} , the harder LWE is. The parameter m_{LWE} is then correlated to λ .

To find exact values for the LWE parameters, we refer to existing parameter selection tools [APS15, BKMR24].

11.4.3 Standard security properties

Theorem 11.20 (Privacy). Let $\lambda \in \mathbb{N}$, $N, p \in \mathbb{N}$. For any noise distribution χ , the scheme $\mathcal{E} = (\text{Gen}, \text{Eval}_\chi, \text{Decode})$ as described in Algorithm 14 is an FSS scheme for the family of all point functions with $\alpha \in \{1, \dots, N\}$ and $\beta \in \mathbb{F}_q$.

Assuming that the LWE assumption holds, then the scheme $\mathcal{E}$ is private against at most m semi-honest parties with $m < p$.

Proof. As our noisy DPF is a variant of the DPF proposed by [CBM15], we adapt their proof easily.

To prove the privacy (Definition 2.5), we must show that (for every set of m corrupted parties) there exists an efficient simulator that, for any input functions, outputs samples from a distribution that is computationally indistinguishable from the distribution of the DPF keys.

The simulator Sim returns the following: $\tilde{b}_i \xleftarrow{R} (\mathbb{F}_q)^\nu$, $\tilde{s}_i \xleftarrow{R} \mathbb{F}_q^{m_{\text{LWE}}}$, $\tilde{v} \xleftarrow{R} \mathbb{F}_q^\nu$.

Since the vectors b_i and s_i vectors outputted by Gen are also random, the simulator's output distributions $\tilde{b}_i$ and $\tilde{s}_i$ are computationally indistinguishable from

that induced by the distribution of a single output of Gen.

By assumption, G_{LWE} is a PRG because LWE is hard. The v vector output by Gen is computationally indistinguishable from random, since it is masked with the output of the PRG G_{LWE} seeded with a seed unknown to the adversary (i.e., s_*). As highlighted by [CBM15], an efficient algorithm to distinguish v from random can also distinguish the PRG output from random. $\square$

Remark 11.21. The privacy of the noisy DPF described in Algorithm 14 holds for any distribution χ used in the Eval_χ ; including the Dirac distribution on 0 (i.e., no added noise). $\lrcorner$

While the privacy hold for any χ , Section 11.4.4 shows that the function-private output only holds for some distributions.

Theorem 11.22 (Correctness). Let $\mathcal{E} = (\text{Gen}, \text{Eval}_\chi, \text{Decode})$ be the DPF scheme described in Algorithm 14, with $\chi \sim \mathcal{N}(0, \sigma_{\text{flood}}^2)$. Let the noise used for the LWE-based PRG and for flooding have Gaussian distributions with standard deviation σ_{LWE} and σ_{flood} , respectively. Then $\mathcal{E}$ is ψ -noisy correct with $\psi \sim \mathcal{N}(0, (p+1)\sigma_{\text{LWE}}^2 + p\sigma_{\text{flood}}^2)$.

Proof. Given the key $k_i = (b_i, s_i, v)$ of party i and a point $x \in \mathcal{X}$, the corresponding share is

$$\text{Eval}((b_i, s_i, v), x) = g_i[x_r] + b_i[x_l]v[x_r] + \text{err}_i$$

where $g_i \leftarrow G_{\text{LWE}}(s_i[x_l])$, $\text{err}_i \leftarrow \mathcal{N}(0, \sigma_{\text{flood}}^2)$ and (x_l, x_r) is the representation of x in the $\nu \times \nu$ grid. Direct calculation shows that

$$\begin{aligned} \mathcal{D}_{f,x} &= \text{Decode}(\text{Eval}(k_1, x), \dots, \text{Eval}(k_p, x)) \\ &= \sum_{i=1}^p \left(G_{\text{LWE}}(s_i[x_l])[x_r] + b_i[x_l]v[x_r] + \text{err}_i \right) \\ &= \left(\sum_{i=1}^p G_{\text{LWE}}(s_i[x_l]) \right) [x_r] + v[x_r] \sum_{i=1}^p b_i[x_l] + \sum_{i=1}^p \text{err}_i \\ &= G_{\text{LWE}}(s_* \cdot e_{\alpha_l}[x_l])[x_r] + v[x_r] \cdot e_{\alpha_l}[x_l] + \sum_{i=1}^p \text{err}_i + \sum_{i=1}^p \text{err}'_i \end{aligned}$$

where $\text{err}'_i \leftarrow \mathcal{N}(0, \sigma_{\text{LWE}}^2)$ captures the noise from $G_{\text{LWE}}(s_i[x_l])$. Now, by cases:

- If $x_l = \alpha_l$, namely we are in the row encoding α , recalling that $v = \beta e_{\alpha_r} - \blacktriangledown$

$G_{\text{LWE}}(s_*)$, we have

$$\begin{aligned} \mathcal{D}_{f,x} &= G_{\text{LWE}}(s_*)[x_r] + \beta e_{\alpha_r}[x_r] - G_{\text{LWE}}(s_*)[x_r] + \sum_{i=1}^p \text{err}_i + \sum_{i=1}^p \text{err}'_i \\ &= \beta e_{\alpha_r}[x_r] + \sum_{i=1}^p \text{err}_i + \sum_{i=1}^p \text{err}'_i + \text{err}'_* , \\ &\quad \text{with } \text{err}'_* \leftarrow \mathcal{N}(0, \sigma_{\text{LWE}}^2) . \end{aligned}$$

- Else, we have $\mathcal{D}_{f,x} = \sum_{i=1}^p \text{err}_i + \sum_{i=1}^p \text{err}'_i$.

So we have that $\mathcal{D}_{f,x} - f(x) = \sum_{i=1}^p \text{err}_i + \sum_{i=1}^p \text{err}'_i + \text{err}'_* \cdot e_{\alpha_l}[x_l]$, which behaves as $\mathcal{N}(0, (p+1)\sigma_{\text{LWE}}^2 + p\sigma_{\text{flood}}^2)$ when $x_l = \alpha_l$ and as $\mathcal{N}(0, p\sigma_{\text{LWE}}^2 + p\sigma_{\text{flood}}^2)$ otherwise. So, $\mathcal{D}_{f,x} - f(x)$ is stochastically dominated in absolute value by the distribution with the largest variance, that is $\psi \sim \mathcal{N}(0, (p+1)\sigma_{\text{LWE}}^2 + p\sigma_{\text{flood}}^2)$, hence the scheme is ψ -noisy. $\square$

In the description and proofs, we chose to use the (continuous) Gaussian distribution to simplify the analysis and keep it more readable. In practice, since computations are carried out over finite fields, a discrete version should be used instead. The proofs would carry over with minimal changes, with only some care needed since the sum of two discrete Gaussians is not itself a discrete Gaussian, existing works [CKS22, KLS21] provided bounds on the approximation error. Observe that the approximation error induced by the use of continuous Gaussians is tiny [KLS21]. For example, for two discrete Gaussians $X \sim \mathcal{N}_{\mathbb{Z}}(0, \sigma^2)$ and $Y \sim \mathcal{N}_{\mathbb{Z}}(0, \tau^2)$, the Rényi divergence between $X + Y$ and $W \sim \mathcal{N}_{\mathbb{Z}}(0, \tau^2 + \sigma^2)$ is only 10^{-12} when $\sigma^2 = \tau^2 = 3$ [KLS21].

This theorem highlights a quite intuitive results: the scheme is ψ -noisy with ψ being the convolution of the different noise distributions added during the generation and evaluation algorithms. As we use Gaussian distributions, we are particularly interested in the variance of $\text{Var}(\psi) = (p+1)\sigma_{\text{LWE}}^2 + p\sigma_{\text{flood}}^2$.

Let us dissect this variance. Remember that this variance corresponds to the variance of the decoded output $\text{Decode}(\text{Eval}(k_1, x), \dots, \text{Eval}(k_p, x))$. Observe that each evaluation $\text{Eval}(k_i, x)$ induces an LWE noise and a flooding noise. These evaluations are then responsible for $p(\sigma_{\text{LWE}}^2 + \sigma_{\text{flood}}^2)$ (i.e., $\sigma_{\text{LWE}}^2 + \sigma_{\text{flood}}^2$ per evaluation). After considering these evaluation-induced noises, there remains an extra unexplained σ_{LWE}^2 in $\text{Var}(\psi)$. This last term comes from the LWE-based PRG evaluation performed during the generation. Indeed, the vector v contained in each FSS key is masked using an LWE-based PRG. This “noisy masked vector” v is then responsible for this part of the scheme noise. For a detailed analysis, we refer to the proof of 11.22.

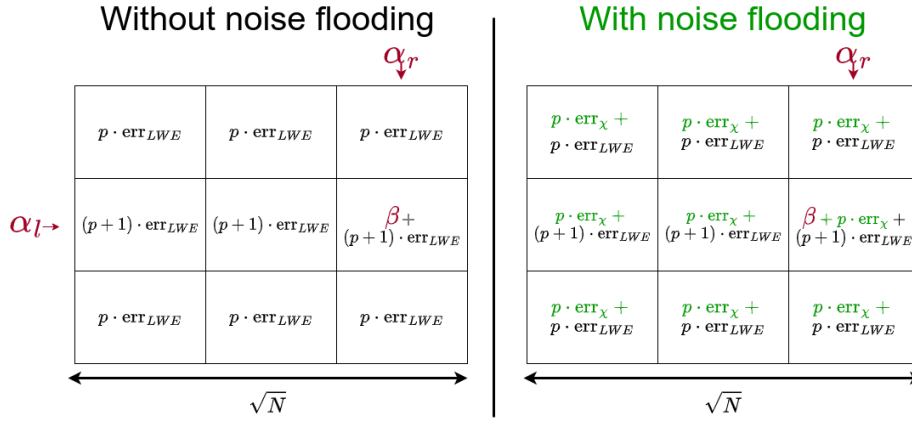


Figure 11.23 / Output of $\text{Decode}(\llbracket f(x) \rrbracket_1, \dots, \llbracket f(x) \rrbracket_p)$ (for all x in the domain represented as a grid) using our noisy scheme with and without noise flooding

11.4.4 Share properties

Succinctness As the decoding is linear, we can share any secret function with output in $\mathbb{F}_q$ with 0-succinct shares as the shares outputted by Eval are in $\mathbb{F}_q$.

However, we may reserve the low-order to the noise (and scale the secret accordingly). Thus, our scheme is theoretically 0-succinct, but practitioners may pick slightly larger prime fields in order to accommodate the noise. The secret scaling based on the noise is dependent on the application (and the expected accuracy), so we refer to Section 11.5 that explains how to leverage the FSS noise in DP computations.

Function-private output As mentioned before, while the addition of χ might seem arbitrary and useless, it is mandatory to preserve function-private output. To understand it, let us consider Algorithm 14 without the χ noise addition.

Let us analyze the protocol output for a point function f (with $f(\alpha) = \beta$) and an evaluation of $f(x)$ with $x \neq \alpha$. Without this χ noise, $\text{Decode}(\llbracket f(x) \rrbracket_1, \dots, \llbracket f(x) \rrbracket_p)$ would include p LWE noises if $x_l \neq \alpha_l$; i.e., $\text{Decode}(\llbracket f(x) \rrbracket_1, \dots, \llbracket f(x) \rrbracket_p) = p \cdot \text{err}_{LWE}$. On the contrary, if $x_l = \alpha_l$, we would have $\text{Decode}(\llbracket f(x) \rrbracket_1, \dots, \llbracket f(x) \rrbracket_p) = (p+1) \cdot \text{err}_{LWE}$. The extra err_{LWE} in the second case come from the vector v because it was masked during Gen using the noisy PRG G_{LWE} . The proof of Theorem 11.22 further details the phenomenon.

Using this difference in noise distribution, an adversary can partially learn the value α from the evaluation and decoding of $f(x)$, even if we do not evaluate $f(\alpha)$. This leakage is exactly what function-private output aims to prevent!

To avoid this leakage, Eval_χ adds a noise term $\text{err}_{\text{flood}} \leftarrow \chi$ to mask the LWE noise, a technique known as noise flooding [DDE⁺23], also used in Fully Homomorphic Encryption. This χ noise (sampled on Line 20 of Algorithm 14) “floods” the LWE noise and ensure that the noise distribution is the same on all points (from a PPT adversary perspective). Fig. 11.23 sketches this solution. However,

function privacy is only preserved if χ is chosen such that a PPT adversary cannot distinguish $\text{err}_{\text{flood}} + \text{err}_{\text{LWE}}$ from $\text{err}_{\text{flood}}$ alone. We formalize this requirement in the following theorem, which specifies the necessary conditions on χ to ensure FPO in our scheme:

Theorem 11.24 (Function-private output). Let $\mathcal{E}$ be the FSS scheme described in Algorithm 14. Let the noise used for the PRG and for flooding have Gaussian distribution with standard deviation σ_{LWE} and σ_{flood} , respectively. If $\sigma_{\text{flood}} \geq \sqrt{2^{\lambda-1.5}/(p-m)-1} \sigma_{\text{LWE}}$, with m being the number of corrupted parties, then $\mathcal{E}$ has ψ -noisy FPO with $\psi \sim \mathcal{N}(0, (p+1)\sigma_{\text{LWE}}^2 + p\sigma_{\text{flood}}^2)$.

Before proving this theorem, we recall some results on statistical indistinguishability:

Lemma 11.25. Let A, B be two centered Gaussian random variables $A \sim \mathcal{N}(0, \sigma_A^2)$ and $B \sim \mathcal{N}(0, \sigma_B^2)$. The advantage of any binary distinguisher $\mathcal{D}$ (not necessarily polynomially-bounded) is bounded by:

$$|\Pr[\mathcal{D}(A) = 1] - \Pr[\mathcal{D}(B) = 1]| \leq \frac{1}{2} \sqrt{\frac{\sigma_A^2}{\sigma_B^2} - \ln \frac{\sigma_A^2}{\sigma_B^2} - 1} .$$

Proof. For any two random variables A, B , we have that

$$|\Pr[\mathcal{D}(A) = 1] - \Pr[\mathcal{D}(B) = 1]| \leq \Delta(A, B)$$

where $\Delta(A, B)$ is the statistical distance between A and B , defined as

$$\Delta(A, B) := \frac{1}{2} \sum_x |\Pr[A = x] - \Pr[B = x]| .$$

By the Pinsker's inequality we know that

$$\Delta(A, B) \leq \sqrt{\frac{1}{2} D_{\text{KL}}(A \parallel B)}$$

where $D_{\text{KL}}(A \parallel B)$ is the Kullback-Leibler (KL) divergence of A and B , defined as

$$D_{\text{KL}}(A \parallel B) := \sum_x \Pr[A = x] \cdot \ln \frac{\Pr[A = x]}{\Pr[B = x]} .$$

For $A \sim \mathcal{N}(0, \sigma_A^2)$ and $B \sim \mathcal{N}(0, \sigma_B^2)$ direct calculations show that their KL divergence is

$$D_{\text{KL}}(A \parallel B) = \frac{1}{2} \left(\frac{\sigma_A^2}{\sigma_B^2} - \ln \frac{\sigma_A^2}{\sigma_B^2} - 1 \right) .$$

A distinguisher advantage is then upper-bounded by

$$|\Pr[\mathcal{D}(A) = 1] - \Pr[\mathcal{D}(B) = 1]| \leq \frac{1}{2} \sqrt{\frac{\sigma_A^2}{\sigma_B^2} - \ln \frac{\sigma_A^2}{\sigma_B^2} - 1} .$$

□

Using this result, we can prove Theorem 11.24:

Proof. Without loss of generality, let us assume that the corrupted subset of parties is $A = \{1, \dots, m\}$. We build a simulator $\mathcal{S}$ for Definition 11.11

$$\mathcal{S} = \mathcal{S}(\lambda, \mathcal{F}, \{(x_1, y_1), \dots, (x_n, y_n)\})$$

as follows. The simulator samples random quantities $\tilde{b}_1, \dots, \tilde{b}_m \in \mathbb{F}_q^\nu$, $\tilde{s}_1, \dots, \tilde{s}_m \in (\mathbb{F}_q^{m_{\text{LWE}}})^\nu$, and $v \in \mathbb{F}_q^\nu$. The shares of the corrupted parties can be computed following the protocol: $\tilde{y}_{i,j} = G_{\text{LWE}}(\tilde{s}_i[x_l^{(j)}])[x_r^{(j)}] + \tilde{b}_i[x_l^{(j)}]v[x_r^{(j)}] + \text{err}_{i,j}$ where $\text{err}_{i,j} \sim \mathcal{N}(0, \sigma_{\text{flood}}^2)$. The shares of the honest parties, which still belong to the view of the adversary, are all sampled uniformly at random, but the last share, which is computed as

$$\tilde{y}_{p,j} = -(\tilde{y}_{1,j} + \dots + \tilde{y}_{p-1,j}) + y_j + \sum_{i=1}^p \text{ERR}_{i,j}$$

where $\text{ERR}_{i,j}$ is the sum of the previously sampled flooding and LWE errors for $i \leq m$, while it is freshly sampled from $\mathcal{N}(0, \sigma_{\text{LWE}}^2 + \sigma_{\text{flood}}^2)$ for $i > m$. The simulated view is then

$$(\{(\tilde{b}_i, \tilde{s}_i, \tilde{v})\}_{i \in A}, \{\tilde{y}_{1,1}, \dots, \tilde{y}_{p,n}\}, \{(x_1, y_1), \dots, (x_n, y_n)\}).$$

Note that even if we do not mention it explicitly (for the sake of having lighter notation), all randomness of the corrupted parties is also part of the adversary's view. The keys $(\tilde{b}_i, \tilde{s}_i, \tilde{v})$ are clearly indistinguishable for the same reason as in the proof of Theorem 11.20. For any $j \in \{1, \dots, n\}$, each share $\tilde{y}_{i,j}$ is individually indistinguishable from its real equivalent due to the first term $G_{\text{LWE}}(\tilde{s}_i[x_l^{(j)}])[x_r^{(j)}]$ acting as a uniform mask. Moreover, it can be shown that no subset of these shares provide any useful information to the distinguisher, but when all shares for a specific x_j are considered, which is the only interesting case. In this case, their sum leads to

$$\tilde{y}_{1,j} + \dots + \tilde{y}_{p,j} = y_j + \text{ERR}_j$$

with $\text{ERR}_j := \sum_{i=1}^p \text{ERR}_{i,j} \sim \mathcal{N}(0, p(\sigma_{\text{LWE}}^2 + \sigma_{\text{flood}}^2))$. As we have seen in the proof of Theorem 11.22, the decoding of the real shares leads to two slightly different error distributions depending on whether we are on the row encoding α or not. In particular, we either get a variance $\sigma_A^2 = (p+1)\sigma_{\text{LWE}}^2 + p\sigma_{\text{flood}}^2$ or $\sigma_B^2 = p\sigma_{\text{LWE}}^2 + p\sigma_{\text{flood}}^2$.

The distinguisher can potentially remove the randomness of the dishonest parties to make its job easier. We hence consider the worst case of

$$\begin{aligned} \text{ERR}_j &\sim \mathcal{N}(0, (p-m)(\sigma_{\text{LWE}}^2 + \sigma_{\text{flood}}^2)) \\ \sigma_A^2 &= (p-m+1)\sigma_{\text{LWE}}^2 + (p-m)\sigma_{\text{flood}}^2 \\ \sigma_B^2 &= (p-m)\sigma_{\text{LWE}}^2 + (p-m)\sigma_{\text{flood}}^2 \end{aligned}$$

▼

and show that if $\sigma_{\text{flood}} \geq \sqrt{2^{\lambda-1.5}/(p-m)} - 1 \sigma_{\text{LWE}}$, it is not possible to distinguish the simulated shares' decoding in the first case σ_A better than $\leq 2^{-\lambda}$ (the second case σ_B is trivial as the error distribution coincides with the one of err_j). Let $x = \sigma_A^2/\sigma_B^2$. Thanks to Lemma 11.25, we just need to prove that $1/2\sqrt{x - \ln x - 1} \leq 2^{-\lambda}$. Since $\sigma_A^2 > \sigma_B^2$ we have that $x > 1$. By taking a Taylor expansion of order 1 of $x - \ln x - 1$, we get $(x-1)^2/2$, in particular we have that $x - \ln x - 1 < (x-1)^2/2$ for all $x > 1$. Hence,

$$\frac{1}{2} \sqrt{x - \ln x - 1} < \frac{1}{2} \sqrt{(x-1)^2/2} = \frac{x-1}{2\sqrt{2}} .$$

Now,

$$\begin{aligned} x = \frac{\sigma_A^2}{\sigma_B^2} &= \frac{(p-m+1)\sigma_{\text{LWE}}^2 + (p-m)\sigma_{\text{flood}}^2}{(p-m)\sigma_{\text{LWE}}^2 + (p-m)\sigma_{\text{flood}}^2} \\ &= \frac{\sigma_{\text{LWE}}^2}{(p-m)\sigma_{\text{LWE}}^2 + (p-m)\sigma_{\text{flood}}^2} + 1 . \end{aligned}$$

So,

$$\begin{aligned} \frac{x-1}{2\sqrt{2}} \leq 2^{-\lambda} &\Leftrightarrow \frac{\sigma_{\text{LWE}}^2}{(p-m)(\sigma_{\text{LWE}}^2 + \sigma_{\text{flood}}^2)} \leq 2^{1.5-\lambda} \\ &\Leftrightarrow \sigma_{\text{flood}}^2 \geq \left(\frac{2^{\lambda-1.5}}{p-m} - 1 \right) \sigma_{\text{LWE}}^2 . \end{aligned}$$

□

Compressibility As our scheme provides noisy correctness and has a linear share decoding, according to Theorem 11.14, our scheme also provides noisy linear compressibility.

11.4.5 Comparison with state-of-the-art

Table 11.26 compares our noisy DPF to existing schemes, based on their key size and share compressibility. We also include the “trivial scheme”: sharing the secret function truth table.

Based on key sizes, we can categorize existing schemes in three categories. First, the PRG-based schemes with an exponential factor [BGI15, BK022], that makes them impractical. Second, the DDH-based schemes [CBM15, KPM24] with no dependency on the number of parties: $O(\sqrt{N}(\lambda + \log q))$ key size. Third, the schemes with a polynomial dependency on p : $O(\sqrt{N} \cdot \binom{p-1}{m} \cdot \log q)$ for the information-theoretic scheme of [BK022] and $O(\sqrt{N} \cdot p^3 \lambda^3 \cdot (\log q + \lambda))$ for the PRG-based of [GWW25].

While DDH-based schemes [CBM15, KPM24] have the best key size, they lack an essential property: compressibility. Thus, contrary to other schemes, they cannot support statistics applications described in Section 2.2.3.

Notations: p parties, m dishonest parties, N domain size, output in $\mathbb{F}_q$.

Scheme	Hard-ness	Compress- ibility	FPO	Key size
Dishonest-majority schemes				
Trivial	None	✓	✓	$N \cdot \log q$
[BGI15]	PRG	✓	✓	$O(\sqrt{N} \cdot q^{\frac{p-1}{2}} \cdot (\log q + \lambda))$
[CBM15]	DDH	✗	✓	$O(\sqrt{N} \cdot (\lambda + \log q))$
[KPM24]	DDH	✗	✗	$O(\sqrt{N} \cdot (\lambda + \log q))$
[GWW25]	PRG	✓	✓	$O(\sqrt{N} \cdot p^3 \lambda^3 \cdot (\log q + \lambda))$
Ours	LWE	✓(noisy)	✓	$O(\sqrt{N} \cdot \sqrt{(1 + \lambda)} \cdot \log q)$
Honest-majority schemes				
[BK022]	PRG	✓	✓	$O(\sqrt[4]{N} \cdot \sqrt{q^{p^m}} \cdot \binom{p-1}{m} \cdot (\lambda + \log q))$
[BK022]	None	✓	✓	$O(\sqrt{N} \cdot \binom{p-1}{m} \cdot \log q)$

Table 11.26 / Comparison of the multi-party DPF schemes

Finally, our adaptation of [CBM15] provides the best of both worlds: the **best asymptotic key size** with no dependence on p (i.e., $O(\sqrt{N} \cdot \sqrt{(1 + \lambda)} \cdot \log q)$) and **all desirable share properties**. In other words, we improved the initial scheme by providing an essential share property absent from [CBM15].

11.4.6 Extension to comparison functions

The full version of this chapter [DMH⁺25] also extends our DPF scheme to comparison function.

11.5 Leveraging FSS noise in DP computations

Several works [Ogi24, SZS⁺23, SSV⁺22] have studied the interactions between “cryptographic noise” and DP. On the one hand, [Ogi24, SZS⁺23] have shown how to leverage the noise introduced by homomorphic encryption schemes. On the other hand, [SSV⁺22] designed an LWE-based protocol to securely aggregate dense vectors; using the LWE noise to provide DP guarantees.

Similarly, we want to show how to leverage the noise induced by our noisy schemes

for DP guarantees in statistics applications. We demonstrate this approach for private histograms and highlight promising future directions to design FSS-based differentially private computations.

Thus, noisy FSS provides *by design* stronger privacy guarantees than classic FSS. Indeed, the classic FSS privacy property guarantees that the function remain private during the MPC protocol, but the MPC output (e.g., a histogram) can still leak information about the input data. As we can leverage the FSS noise to build DP guarantees, the leakage is mitigated in statistics applications based on noisy schemes.

Note that classic FSS protocols (e.g., [GWW25]) can be updated to add a DP noise on their shares at the end of their Eval algorithm. Such a modification would transform them directly into a noisy scheme as they would no longer satisfy the classic FSS correctness. Hence, adapting classic FSS to provide DP guarantees naturally leads to (trivial) noisy schemes.

11.5.1 Differentially private histogram with FSS noise

As detailed in Algorithm 1, to compute the histogram of n data inputs $x_1, \dots, x_n \in \{1 \dots N\}$, we need to aggregate n secret point functions $f_1, \dots, f_n$ such that $f_i(x_i) = 1$, 0 otherwise. In Algorithm 1, if the DPF scheme $\mathcal{E}$ is a noisy scheme, the protocol would output a histogram noised with a sum of discrete Gaussians. This subsection analyses this noise and the DP guarantees that it provides.

Theorem 11.22 provides a stochastic dominance on our DPF noise distribution. However, DP guarantees require knowing what is the *minimum* noise added to a single point. The Eval_χ of Algorithm 14 systematically adds a noise drawn from a distribution χ to the evaluation results.

As explained in Theorem 11.24, we use $\chi \sim \mathcal{N}_{\mathbb{Z}}(0, \sigma_{\text{flood}}^2)$ to guarantee FPO (see the corresponding proof for a lower bound on σ_{flood}^2). Thus, each noisy function output (i.e., $\sum_{i=1}^p \text{Eval}_\chi(k_i, x) \approx f_i(x)$) integrates a noise drawn from the sum of p discrete Gaussians with variance σ_{flood}^2 .

A private histogram requires aggregating n (noisy) point functions, so the total noise distribution added to the histogram is the sum of $n \times p$ discrete Gaussians with variance σ_{flood}^2 . We can trivially apply the results from [KLS21] (Proposition 1.10) for $\sigma = \sigma_{\text{flood}}$.

For basic histograms in which each data owner contributes to a single bin with a value 0 or 1, the sensitivity is $\Delta_1 = \Delta_2 = 1$. Based on this sensitivity and our discrete Gaussian variance σ_{flood}^2 , we can deduce the privacy level using Proposition 1.10.

We could very slightly lower the ε by also taking into account the LWE noise. However, the LWE noise and the flooding noise have different variances (σ_{LWE}^2 and σ_{flood}^2). Thus, we cannot apply naively Proposition 1.10 because it requires discrete

Gaussians with identical variance; we would need to redevelop this complex bound for a sum of discrete Gaussians with different variances.

Integrating the LWE noise would not significantly improve the ε . As explained in Theorem 11.24, the flooding noise is set to hide the influence of the LWE noise (i.e., to guarantee function-private output). The amplitude of the LWE noise is multiple orders of magnitude smaller than the flooding noise. Thus, the LWE noise would have a negligible impact on ε .

Note that Theorem 11.24 provides a lower bound on the noise variance to guarantee function-private output. We can naturally increase the variance σ_{flood}^2 (above the lower bound defined in Theorem 11.24) to lower ε ; enhancing then the privacy guarantees.

11.5.2 Potential extensions

Our goal in this section was to highlight the essential DP results necessary to incorporate FSS noise in differentially private computations. We demonstrated these results on a simple example, but several extensions are possible.

First, we could combine the FSS noise with other DP mechanisms. Noisy FSS provides “DP for free” as the noise is inherently present in the scheme. However, this noise could be combined with other noises in the DP computation (e.g., sharing secret functions f_i such that $f_i(x_i + e) = 1$ instead of $f_i(x_i) = 1$) to have the best accuracy-privacy trade-off.

Second, we have only presented noisy schemes based on Gaussian noises, but LWE variants (e.g., Learning with Rounding [BLMR13]) can output other noise distributions. Thus, future research could also study noisy FSS schemes with non-Gaussian noise. Such research could require new DP results compatible with these noise distributions.

Conclusion

First, this chapter formalized FSS properties that had been introduced informally in existing works. Then, we proposed a relaxed form of FSS allowing for some noise in the function output: noisy FSS. We adapted an existing scheme of [CBM15] in order to build a noisy DPF providing both the best key size, and all desirable share properties. Finally, we showed how to leverage this noise to provide differential privacy guarantees in FSS applications to statistics.

Part V

Towards Real-World Deployments

This final part explores two auxiliary contributions that address practical aspects of PET real-world deployment. On the one hand, we curate a new graph dataset called *Fedivertex*, which models interactions between servers in decentralized social media platforms. Because decentralized social media is a naturally well-suited for decentralized PPML deployments, our dataset challenges existing assumptions and enables new applications in this domain.

On the other hand, we analyze the energy consumption of three cryptographic PETs. While privacy is often seen as a value to protect at all costs, cryptographic techniques can incur significant performance overheads conflicting with carbon footprint reduction goals. We measure the energy consumption overhead induced by TLS, SSE, and FHE, and discuss orthogonal questions to outline promising directions toward sustainable PETs.

We conclude this part and the entire thesis with a final chapter summarizing the takeaways of this manuscript and presenting several directions for future work.

12 Fedivertex: a graph dataset based on decentralized social networks for trustworthy Machine Learning

Decentralized machine learning is increasingly popular, as it enables better scalability and control over the data. A major challenge in this setting is that learning dynamics depend on the topology of the communication graph, which motivates the use of real graph datasets for benchmarking decentralized algorithms. Unfortunately, existing graph datasets usually focus on user interactions, which provides unrealistic communication graphs to benchmark decentralized learning algorithms that commonly involve interactions between servers.

This chapter introduces *Fedivertex*, a new dataset of more than 350 graphs, covering seven decentralized social networks, crawled weekly over more than 6 months. Our graphs model interactions between servers on these decentralized social networks. These decentralized platforms being a great application area for PPML algorithms (notably in moderation or recommendation), our dataset is particularly relevant for benchmarking decentralized PPML algorithms. Fedivertex challenges existing works as its properties significantly differs from existing graph datasets used in decentralized ML literature.

This chapter is drawn from [DC25b].

12.1 Introduction

Decentralized machine learning [HRS⁺24] has gained significant popularity in the past years. In this paradigm, each node possesses a local dataset and some computational resources, and collaborates with other participants by exchanging messages through peer-to-peer communication during the training of a global, potentially personalized, machine learning model. In comparison to federated learning [KMA⁺21], which keeps data local but orchestrates training through a central server, decentralized learning offers additional flexibility as it avoids the bottlenecks and single points of failure that arise from centralized supervision. The shift towards decentralized learning can also be motivated by trust, as the communication graph can reflect users' chosen collaborations; often referred to as nodes, instances, clients, or participants in this context. The network topology has an impact on the learning dynamics [KCD⁺21], particularly in the presence of data heterogeneity [HPMG20, VHK⁺21] and in terms of privacy guarantees [MCB24].

Decentralized learning has numerous real-world use cases [HRS⁺24], as nodes can represent healthcare institutions or sensors distributed across installations. One of the most compelling applications is for decentralized social networks [BRC⁺22].

In such cases, the graph often captures complex and diverse relationships, which explains its popularity in machine learning. In particular, this motivates various graph learning tasks [TL10], including community detection, node classification, and edge prediction. Social network dynamics also raise interesting questions related to polarization and time-evolving properties of the graph [TKK⁺09]. Addressing these questions requires access to relevant social network datasets that allow studying these properties.

The Fediverse (a contraction of “federation” and “universe”) provides decentralized and interoperable online social services. It is often seen as an alternative [AM22] to major social networks operated by for-profit companies, and it promotes a very different culture. The Fediverse is decentralized across many servers, called *instances*. Anyone can run an instance, which operates independently under the moderation of its owner, and instances collaborate with each other: a user of a given instance can interact with and follow users from other instances. Since 2018, the Fediverse has adopted ActivityPub [LTS⁺18], a protocol and open standard that provides a client-to-server API for creating and modifying content, as well as a federated server-to-server protocol for delivering notifications and content across servers. This enables interoperability between different instances and software. The diversity of platforms, the growing number of users, and the international impact of the Fediverse make it an interesting object of study for the machine learning community. In particular, agents in the Fediverse tend to be more aware of the potential ethical issues of machine learning than traditional users of social networks [SSAG23], and more interested in new features and improvements. This aligns closely with the goals of Trustworthy Machine Learning and the paradigm of collaborative learning, where agents are expected to monitor their participation based on expected benefits.

Importance of real-world graph datasets Decentralized learning is increasingly studied [KMA⁺21, LSTS20, MMR⁺17], with various algorithms based on gossip [BGPS06, Hen23, HBM20, KLB⁺20, LZZ⁺17, MR16, NXT20] or random walks [Eve23] [JRJ10, Hen23, MYH⁺20]. These results highlight the importance of communication graphs for the quality of the final model, the speed of convergence in the presence of heterogeneous data [LBT⁺23], personalization and privacy guarantees [CBU24, CEBM22, DEC25, MCB24], which motivates the use of recent real-world social networks.

Machine learning frequently relies on small social networks, such as the Karate Club [Zac76] or citation networks [GBL98, MNRS00]. Several larger digital social networks are also available via platforms like SNAP [LK14], in particular Facebook and Twitter graphs. It has been shown that for-profit platforms influence user graphs, as their recommendations about whom to follow accelerate the triadic closure process and exacerbate inequality in popularity [SSG16, ZGR18]. This motivates the study of social networks that do not follow this trend. In particular, the Fediverse enables analysis at the level of servers rather than at the level of

individual users, an approach that captures entities more likely to develop consistent collaboration policies. Existing work on the Fediverse remains limited, often focusing on a single network or on interactions between a fixed pair of networks, and typically does not provide reusable datasets [ARS⁺24,HYS24,ZGR18].

Our contributions This chapter presents the first dataset, called *Fedivertex*, covering multiple software platforms in the Fediverse to enable researchers to easily run experiments on decentralized machine learning tasks and to benchmark several graph learning tasks. By surveying seven different platforms and constructing different types of graphs, we are able to capture the diverse dynamics at play in the Fediverse. Our contributions are as follows:

- (1) *Fedivertex*, a large and diverse graph dataset based on the Fediverse. More precisely, our dataset encompasses seven Fediverse platforms, resulting in more than 350 graphs¹: 13 different graphs each with, at least, 24 weekly snapshots obtained through web crawls.
- (2) A Python package, `fedivertex`, available through PyPI, to easily access and use our dataset. The package includes built-in preprocessing tools to download and prepare the graphs for machine learning tasks.
- (3) A novel graph analysis task: *defederation prediction*, which aims to predict which edges or nodes will be removed from the graph at the next iteration, and we propose baselines for this task.

12.2 Dataset presentation

12.2.1 Fediverse software and graphs

In the Fediverse, software is run by servers referred to as *instances*, without any centralized control or coordination. Each instance hosts a subset of *users* and has its own internal rules and moderation. Despite maintaining sovereignty over their rules and storing data locally, instances are not isolated from each other, as they all use the same protocol and standard: ActivityPub [LTS⁺18]. This protocol enables communication between instances and even across services. For instance, a video from PeerTube can be shared on Mastodon; unlike traditional social network silos, where a Facebook user cannot use their account to read tweets or watch YouTube videos. One can think of this interoperability similarly to email, where a user from one provider (e.g., Gmail) can send a message to another (e.g., Outlook). When two users communicate, only their respective instances (and possibly a third instance hosting the interaction) are aware of the message. As a result, data permanence, confidentiality, and moderation depend on the instance.

¹The current snapshots are *only* a starting point as we continue to update the dataset to cover the temporal evolution for the longest period possible.

	Peertube	Mastodon	Pleroma	Misskey	Friendica	Bookwyrm	Lemmy
Type	Video streaming	Micro blogging	Micro blogging	Micro blogging	Micro blogging	Book cataloging	Social news
1st release	2018	2016	2017	2014	2010	2022	2019
# instances	1333	9652	616	1206	101	95	564
# users	583k	8 102k	76k	1 071k	12k	51k	520k
Graphs	follow	federation active user	federation active user	federation active user	federation	federation	federation fed. w/ blocks intra-inst. cross-inst.

Table 12.1 / Overview of the social networks included in Fedivertex. The number of instances and users has been extracted on May 13, 2025 from FediDB, a reference database for the Fediverse communities. These numbers are indicative as the networks evolve over time

Fedivertex focuses on interactions between *instances* within a given software platform. This focus on instance-level graphs is unique in the literature as typical social network graphs focus on user-level graphs (see Fig. 12.2). The instance-level granularity is much more realistic in the context of decentralized ML. For instance, it would be the right granularity for federated spam detection or content recommendation.

In particular, for each social network (except Peertube), the **federation graph** models the undirected communication graph between instances within that network. Federation graphs are naturally dense, because two instances are connected with an edge if they have interacted at least once.

We selected seven of the most popular software platforms in the Fediverse to ensure sufficient activity for graph-based analysis. Our selection covers diverse types of social network to reflect a range of communication dynamics. We provide statistics about each social media in Table 12.1.

Note that this leads to smaller graphs than user-level graphs, because each instance (i.e., graph in the node) hosts hundreds or thousands of users. For example, the Mastodon graphs contain more than 8K instances that host a total of 8M users. Thus, instance-level graphs are smaller, but they can still model interactions from a large user base.

12.2.2 Fediverse social networks

Peertube provides an alternative to YouTube. Users can watch, bookmark, and comment on videos, subscribe to channels, and create private and public playlists. Video search was added in 2020 with SepiaSearch but remains limited; recommendation features are also a limitation compared to Youtube. An instance u can follow an instance v to let its users see all the videos posted on v . We report this **follow graph** as a directed graph with edges of weight 1 for following.

Mastodon was created as an alternative to Twitter in 2016 and is supported by

the German non-profit organization Mastodon gGmbH. Users post short-form status messages of up to 500 characters, known as “toots.” It has experienced several surges in popularity, often in reaction to changes on Twitter, and is sometimes adopted in parallel with it [JST⁺24]. In addition to the federation graphs, introduced above, we also build a weighted, directed **active user graph**, with one node per instance. For each instance u , we take its 10k most recently active users. Whenever one of those users follows someone on instance v , we increase the edge weight by 1. Thus weight of the edge from u to v measures how much content seen on u originates from v . The graph thus contains self-loop as users follow others on the same instance.

Pleroma is a microblogging software similar to Mastodon, and we thus also report active user graph. Principal difference is allowing longer posts by default, up to 5000 characters, and offering a lightweight implementation that can potentially run on a Raspberry Pi.

Misskey is a microblogging platform as well, on which we report the active user graph. It was created in 2014 by Japanese software engineer Eiji “syuilo” Shinoda and allows posts of up to 3000 characters.

Friendica emerged in 2010 as an alternative to Google+ and Facebook, making it the oldest social network in our study, and does not support metadata for active users graph with our crawler.

Bookworm allows users to track their reading activity, write book reviews, and follow friends. Launched in 2022 by Mouse Reeve, it can be seen as an alternative to Goodreads.

Lemmy is organized into communities dedicated to specific topics, where users share links and discuss their content. Although communities are local to an instance, users can subscribe to those hosted by other instances and participate in discussions across instances. In addition to the federation graph, we report two other graphs. Firstly, the **intra-instance graph** where the instance u is linked to v if an user of u has published a message on instance v . This graph is directed and very sparse. Then, in **cross-instance graph**, two instances are connected as soon as there exists a pair of users who published a message in the same thread, but possibly on a third instance. This is an undirected graph, denser than the previous one.

12.2.3 Fedivertex package and availability

Our dataset can be directly downloaded from Kaggle [DC25a]. To facilitate its use, we also provide a Python package, **Fedivertex**, which allows users to directly load the graphs in NetworkX format through an easy-to-use interface. We facilitate interaction with Fedivertex package by releasing several notebooks to analyze the graphs. Finally, we follow the Gephi convention for graph encoding, allowing the graph CSV files to be opened directly in this software [BHJ09].

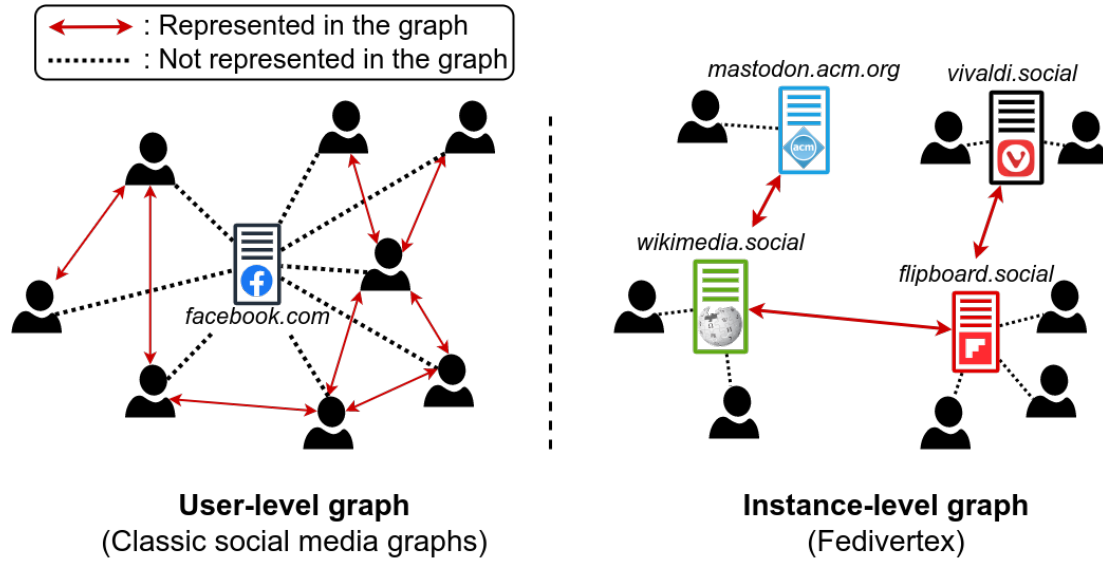


Figure 12.2 / Comparison between user-level and instance-level graphs

12.2.4 Construction via Web crawling

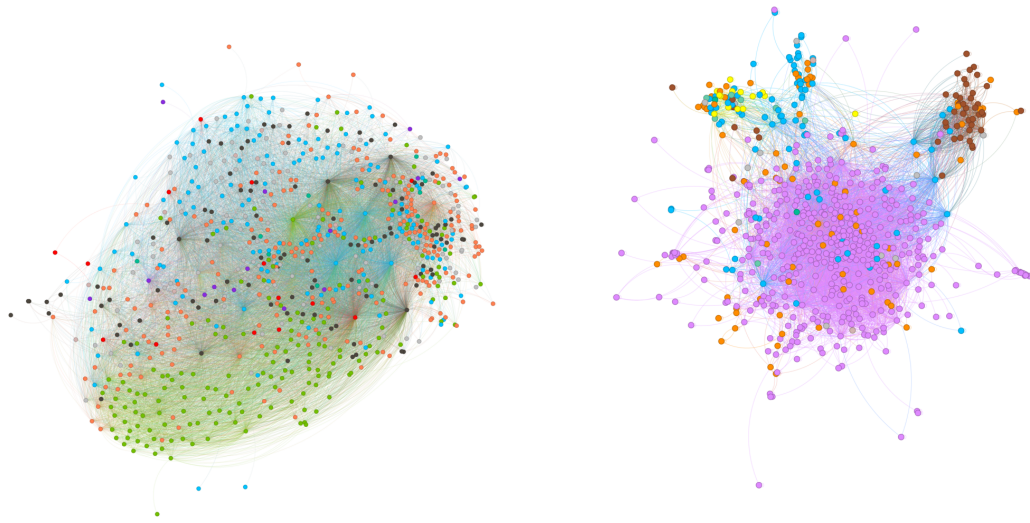
For each of the 13 graphs introduced above, we produce a new version every week. To identify all available servers for a given software, we query the Fediverse Observer², which provides a curated list of Fediverse instances commonly accepted by the community. We then query each of these instances to compute the edges of the graph. Relying on the Fediverse Observer list helps minimize server load and allows us to benefit from existing curation. Notably, the Fediverse Observer’s crawler runs daily and is also open-source. We release the code of our crawler for reproducibility and to allow extensions to other social media or other scraping parameters.

12.2.5 Ethical concerns

This chapter aims to bring more attention to the Fediverse social networks, who could benefit from Trustworthy Machine Learning applications, for instance to assist in moderation task [BRC⁺22] or with user experience and recommendation systems. However, Fediverse software has often been developed to avoid various downsides of hastily deployed machine learning models, from toxicity to invisible filter bubbles, and dark patterns to poor accuracy [CDW⁺23,MTT17]. It is thus part of the challenge and the motivation to focus on tasks where the improvement for the user overcomes the possible drawbacks. Hence, we decide to illustrate our datasets only on tasks that could respect the Fediverse mindset.

The collection of the dataset raises two major problems, the privacy and the possible disturbance for the service. Concerning the impact of the crawling, we designed our crawler to minimize the impact on the servers, by limiting the size

²<https://fediverse.observer>



(a) Peertube Follow graph on April 28th 2025

(b) Misskey Active User graph after removal of Misskey.io on May 7th 2025

Figure 12.3 / Examples of graphs communities based on the official languages in Fedivertex dataset. Colors encode the official language of each instance, with green for French, blue for English, black for German, red for Italian, Chinese in yellow, Japanese in purple, Korean in brown, and orange when there is no official language

of the queries, using a delay of 0.4 second between requests on a given instance to avoid disturbing their operations. We do not disguise our requests into real users' ones and we use a clearly identifiable user agent providing a direct contact to us. We respect the policy of the instances by following the instructions given by `robots.txt` files.

In order to respect the privacy of the users, we did several design choices. First, our dataset represents instance-level interactions and not user-level (see differences in Fig. 12.2), which is a better granularity for privacy. Second, we only report general metadata but never store actual messages or content from the social networks. Third, we only use public API endpoints, which do not require accounts on this platform: we do not try to circumvent these privacy practices by creating accounts to access more information. Fourth, we also respect informal privacy practices. For instance, we ignore all users using the hashtag `#NOBOT` in their profile as it is an informal anti-bot policy on Mastodon. Finally, we limit the access by post-processing instances names to avoid direct clicking links. The whole scraping process was supervised by the legal department of our institutes to ensure compliance with GDPR.

Graph	Directed	Avg. Deg.	Avg. Path Length	Cluster. Coef.	Small- world σ
Bookworm Federation	No	55	1.23	0.89	1.14
Friendica Federation	No	156	1.41	0.85	1.41
Lemmy Federation	No	355	1.18	0.94	1.15
Lemmy Intra-instance	Yes	13	2.03	0.69	3.80
Lemmy Cross-instance	Yes	42	1.60	0.82	1.89
Mastodon Active user	Yes	125	2.09	0.73	21.95
Misskey Federation	No	317	1.66	0.76	2.21
Misskey Active user	Yes	19	2.36	0.62	20.78
Peertube Follow	Yes	23	2.82	0.53	4.45
Pleroma Federation	No	269	1.64	0.82	2.26
Pleroma Active user	Yes	7	3.95	0.30	2.53
Comparison with SNAP Social graphs					
Facebook Ego	No	47	3.7	0.61	39.44
Github	No	29	3.25	0.14	31.49
Wikipedia Vote	No	15	3.25	0.17	519.64

Table 12.4 / Small world properties of the Fedivertex graphs and SNAP Social graphs

12.3 Dataset analysis

12.3.1 Dataset properties

The Fedivertex dataset presents different characteristics depending on both the graph and the software considered. We share in this section a few observations. First, all graphs are provided with their temporal evolution, with new nodes and edges appearing or disappearing each week. For readability, we present only a subset of the graphs and refer the reader to our Kaggle notebooks for a more systematic review.

Communities. Fedivertex contains language labels for Peertube, Lemmy, Bookworm, and Misskey. For Peertube, the labels are directly extracted from the instance descriptions. For the others, we infer the label from the language of the instance description. We report on Fig. 12.3a the labels for the Peertube graph, which exhibits a clear French-speaking community and Misskey, which is dominated by the Japanese community but also exhibits smaller Korean and Chinese communities, and we refer to Section 12.4.3 for the associated prediction task.

Graph statistics. We report few metrics in Table 12.4 that are usually applied

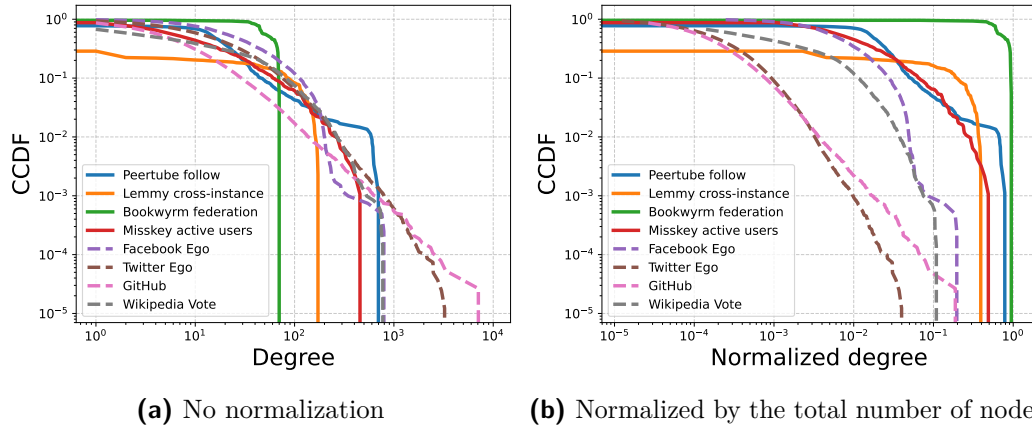


Figure 12.5 / Complementary Cumulative Distribution Function (CCDF) of the degree for several Fedivertex graphs and other widely-used social network graphs

for social networks. All the reported graphs exhibit small-world properties to a certain extent, as they satisfy $\sigma > 1$, which means that a node is more likely to connect to the neighbors of its neighbors and the average path length is small. However, the strength of the small world properties depends on the software.

12.3.2 Comparison with existing graph datasets

We compare our graph with the most popular social network graphs from SNAP [LK14]. We include the Wikipedia Vote graph [LHK10] (which encodes all Wikipedia voting data up to January 2008) as well as the Twitter and Facebook Ego graphs [LM12], corresponding respectively to the Follow and Friend relationships. We also include the GitHub graph [RAS21], where nodes are developers who have starred at least 10 repositories and edges represent mutual follower relationships. On Fig. 12.5, while GitHub and Twitter exhibit the classical power-law decay over much of the support of the distribution, consistently with preferential attachment networks [BP16], the degree distribution of Fedivertex is more diverse. We note some similarities between the Facebook Ego graph and the Peertube follow graph, with a smooth concave decay followed by a short power-law tail for the most popular users. Most of the graphs, however, exhibit only a concave curve, which suggests that the attachment dynamics in Fedivertex differ from those in traditional social networks, allowing for a smoother distribution of node importance. This difference with Fedivertex is confirmed by the statistics of Table 12.4: Fedivertex has a wider range of average path length (from 1.18 to 3.95 versus from 3.25 to 3.7) and of degree (from 7 to 355 versus 15 to 47) and have smaller small-world σ .

12.4 Applications

12.4.1 Decentralized machine learning and statistics

Fedivertex graphs are particularly well adapted to experiments testing fully decentralized machine learning, as they provide a credible use-case scenario. We illustrate this by reproducing the main figures of [CBU24]. They propose training a global model with differential privacy by performing a random walk on the communication graph: at each step, the stochastic gradient is computed on the local dataset of the current node and sanitized by adding Gaussian noise. They derive privacy guarantees in the Pairwise Network Differential Privacy setting, where each pair of nodes has a specific privacy budget depending on their relative position, a high budget corresponds to a greater risk of leaking information. In particular, [CBU24] establishes a connection between the structure of these privacy budgets and the communicability of the graph, showing that nodes close to each other have higher privacy budget than far apart ones. Using graphs with different topologies is interesting to verify that similar patterns appear for privacy losses and known graph quantities such as centrality or communicability.

On Fig. 12.6, we see that the link between communicability and privacy budgets is clear on Fedivertex graphs, with the same patterns visible in Fig. 12.6a and Fig. 12.6b. However, it also shows that real-world graphs can be more challenging in terms of convergence, as Fig. 12.6c exhibits slower convergence on the Mastodon active-user graph than on a synthetic graph with the same number of nodes. This could be explained by the presence of nodes with low centrality, typically connected by only a single edge, which makes it harder for the random walk to visit them frequently enough within the chosen number of steps compared with the more regular graph tested in the original paper.

12.4.2 New links and defederation prediction

A key feature of Fedivertex is its support for temporal graph analysis, as we release weekly snapshots for each graph. Understanding temporal changes in graphs is seen as a major challenge in graph learning [MPG22, RCF⁺20]. This is particularly relevant in social networks, where the creation of links often speeds up the triadic closure of the graph, as friends of friends tend to become friends over time [DKA11, ZGR18], especially when platforms actively recommend new connections [SSG16]. We refer to Table 12.4 for Fedivertex graphs' clustering coefficients.

An interesting behavior observed in the Fediverse is that edges between instances are sometimes deleted, a phenomenon that has received little attention so far, likely because edge deletions are uncommon in other social networks. However, in the context of Fedivertex, predicting deletions is interesting for several reasons. First, in some platforms, deletions are as important and can dominate the change in the number of edges and nodes, as seen in Fig. 12.8. Secondly, avoiding centralization

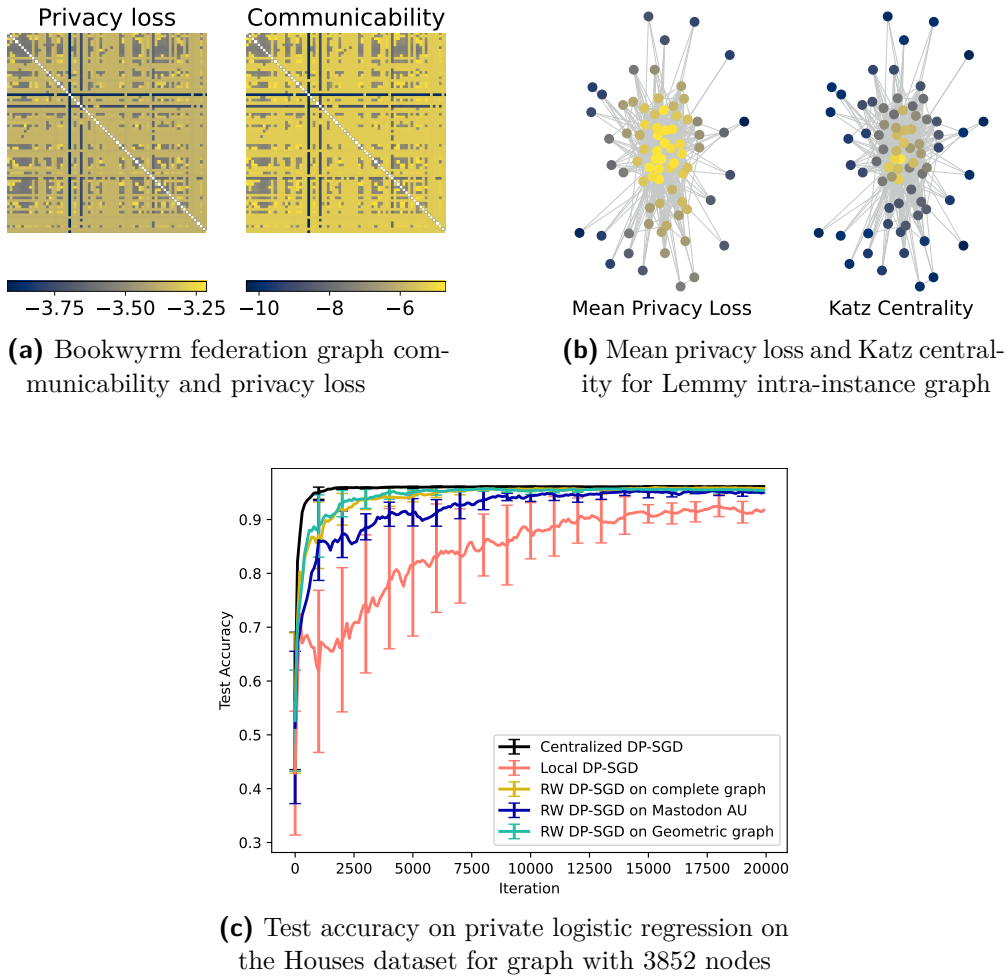


Figure 12.6 / Experiments reproducing the results of [CBU24] with Fedivertex graphs

around a single central server is a key challenge in the Fediverse, and understanding defederation [LRK⁺25] could help maintain a sufficiently decentralized structure. Finally, new link prediction and edge deletion are complementary tasks that may benefit from being studied jointly.

From Fig. 12.8, we observe that federation graphs are the most stable over time, as one could expect from their construction in comparison to active users or cross-instance graphs, where activity can fluctuate. However, while some networks grow during the studied period (Friendica federation and Lemmy cross-instance), others show variations dominated by the loss of edges (such as the Pleroma federation or Misskey active users). More precisely, by restricting our analysis to the subgraph of the nodes present at all 14 dates, most federation graphs have an increasing number of edges, with sometimes sharp drops as we can see for Misskey and Lemmy in Fig. 12.8a. Overall, the network is thus growing, but also shows defederation peaks that are quicker than our weekly crawling. In Fig. 12.8b, we can see that the other graphs do not share this clear increasing trend, but tend to alternate between more edge creation and more edge deletion. Finally, the number of nodes itself varies over time, with new servers appearing and others

Graph		AA		CN		Jaccard		Random	
		Add	Del	Add	Del	Add	Del	Add	Del
Mastodon	AU	38	10	40	9	14	10	0.7 ± 0.5	6 ± 1.4
(Top 1000)									
Misskey	AU	4.3	1.3	4.3	1.7	0.7	2	0.2 ± 0.2	2 ± 0.8
(Top 200)									
Misskey	Fed.	494	62	491	63	396	60	42 ± 6	51 ± 3.7
(Top 1000)									

Table 12.7 / Comparison of Adamic-Adar (AA), Common Neighbors (CN), Jaccard and Random method to predict new edge (Add) and edge deletion (Del) on different graphs by reporting the number of correct predictions in Top- K scores (the higher the better). We report the average of three runs over disjoint periods of time

being deleted, leading to the complex evolution reported in Fig. 12.8c.

We formally introduce the defederation prediction task. Let $\mathcal{G}_t = (\mathcal{V}_t, \mathcal{E}_t)$ be the graph collected at time t and compare it to the graph $\mathcal{G}_{t'} = (\mathcal{V}_{t'}, \mathcal{E}_{t'})$ collected at $t' > t$. Let $\mathcal{G}_c$ be the subgraph induced by $\mathcal{V}_c = \mathcal{V}_t \cap \mathcal{V}_{t'}$; the goal is to predict all edges either created ($e \in (\mathcal{V}_c \times \mathcal{V}_c) \cap (\mathcal{E}_{t'} \setminus \mathcal{E}_t)$) or deleted ($e \in (\mathcal{V}_c \times \mathcal{V}_c) \cap (\mathcal{E}_t \setminus \mathcal{E}_{t'})$) between t and t' . The possible new edges lie in the set $(\mathcal{V}_c \times \mathcal{V}_c) \setminus \mathcal{E}_t$, whereas the deleted ones are in $\mathcal{E}_t$, a set significantly smaller if the graph is sparse. Similarly, one could predict nodes that drop from the graph. In particular, reliable prediction could help detect instances which stopped running because of technical problems despite being active in the graph, and possibly provide technical help to such instances. More formally, the goal would be to predict $\mathcal{V}_t \setminus \mathcal{V}_{t+1}$ given $\mathcal{G}_t$.

New link prediction can be done based on the topology [LK07], by using the fact that similar nodes are more likely to connect. Methods are thus often based on computing scores for each possible pair of nodes, and then return as prediction the edges with the highest scores. Common scores include the number of common neighbors, the Jaccard score, and the Adamic-Adar score. These scores are then evaluated by looking among the top- K predictions how many are indeed new edges, as we report in Table 12.7. Intuitively, deletion could be seen as the opposite of edge creation, so we propose as a baseline, to return the edges with the lowest scores. However, this approach has limited success for federation graphs. We believe this might be due to defederation being extremely quick, and thus the granularity of our current dataset does not seem sufficient to achieve better than random. An interesting future work could be to use our crawler with higher frequency during defederation periods. It might also indicate that other methods should be developed for this task, opening interesting questions for future work. We refer the reader to the appendix of the complete paper [DC25b] for more analysis.

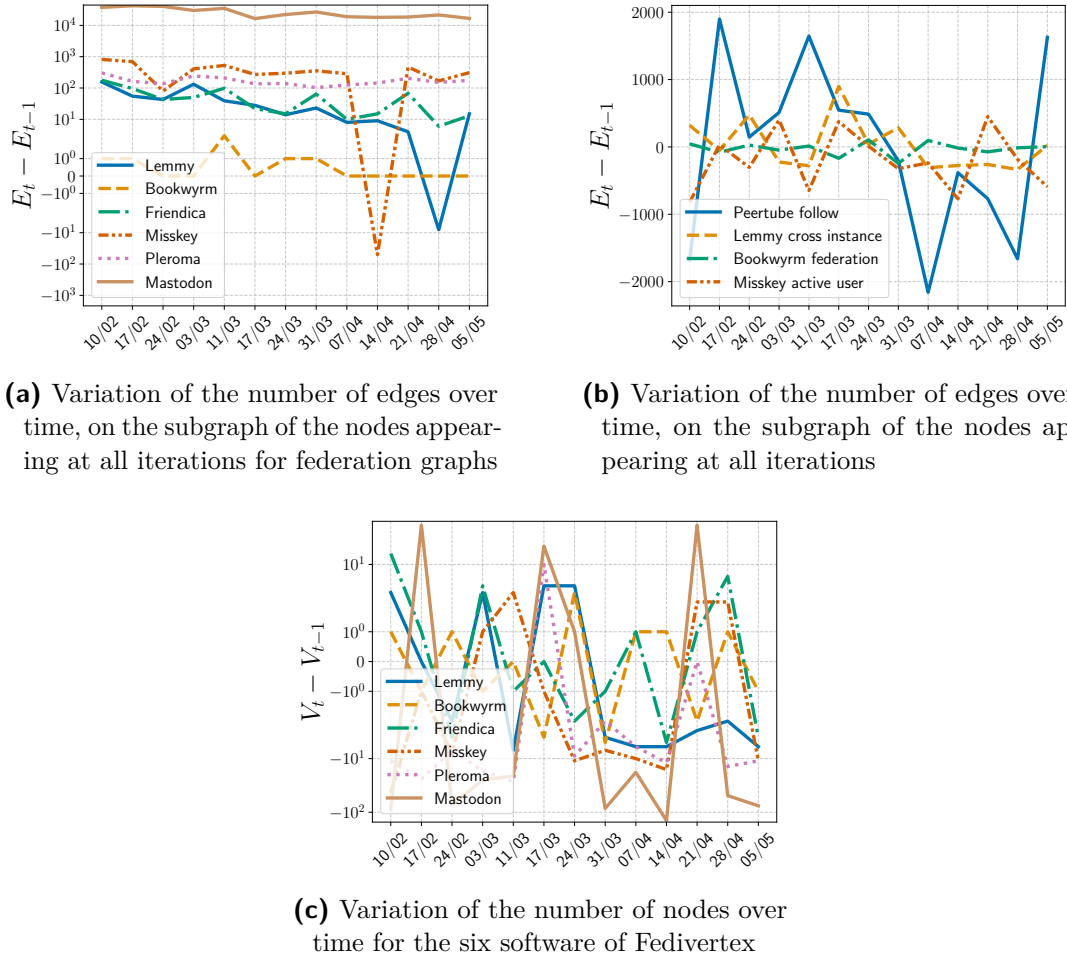


Figure 12.8 / Temporal evolution of Fedivertex graphs

12.4.3 Community detection

The Fediverse social networks are used by many communities that might overlap. The same communities might also use several of the Fediverse software platforms. To illustrate the feasibility of community detection with our dataset, we use the official languages of each instance as ground truth labels, as shown in Fig. 12.3, and test three algorithms: Louvain [BGLL08], Greedy Modularity [CNM04], and Label Propagation [CG10] on the Peertube follow graph and the Misskey active user graph. To avoid many unique labels, we keep only the 5 most represented languages in each graph and ignore the other nodes using other languages. Results are in Table 12.9. We assess the quality of this detection using the Adjusted Rand Index (ARI) and the Adjusted Mutual Information (AMI). The Rand index corresponds to the proportion of node pairs belonging to the same cluster that are classified as such (i.e., the sum of true positives for all classes) over all node pairs, and the adjusted version corresponds to normalization with respect to a random clustering. Similarly, the AMI score corresponds to the mutual information between the ground truth and predicted labels, adjusted for chance. Finally, for each clustering, we report modularity: a metric for unsupervised clustering that

assesses the inherent quality of the partitioning. This suggests that other labels might be suitable as well for clustering the graphs. No method dominates in this benchmark, highlighting that our graphs exhibit diverse structures which may challenge algorithms in different ways.

Other Fedivertex graphs can be used for community detection, and additional labels could be derived from the data, for example, based on the names of the instances or their official descriptions. It is also possible to track the evolution of communities over time.

Graph	Algorithm	ARI	AMI	Modularity
Peertube follow	Louvain	0.055	0.123	0.2168
	Greedy Modularity	0.061	0.110	0.209
	Label Propagation	0.008	0.029	0.003
Misskey active user	Louvain	0.097	0.250	0.611
	Greedy Modularity	0.014	0.165	0.513
	Label Propagation	0.229	0.255	0.027

Table 12.9 / Performance of several community detection algorithms (average of 100 runs for Louvain)

Conclusion

This chapter introduced **Fedivertex**, a dataset modeling interactions between instances across several software platforms of the Fediverse. This is the first dataset publicly released to enable reproducible experiments on graphs from the Fediverse, and it allows the study of more diverse graph dynamics than existing social network datasets. We hope that these graphs can foster machine learning research on this topic and contribute to the development of trustworthy decentralized machine learning, notably on the Fediverse. Among possible applications, this dataset could support the development of decentralized spam detection, the prediction of new or deleted links, the prevention of instance shutdowns through early prediction, and many other tasks related to time-evolving graphs.

Energy Consumption of TLS, Searchable Encryption and Fully Homomorphic Encryption

Privacy-enhancing technologies (PETs) have attracted significant attention in response to privacy regulations, driving the development of applications that prioritize user data protection. At the same time, the information and communication technology (ICT) sector faces growing pressure to reduce its environmental footprint, particularly its energy consumption. While numerous studies have assessed the energy consumption of ICT applications, the environmental impact of cryptographic PETs remains largely unexplored.

This chapter investigates this question by measuring the energy consumption increase induced by three PETs compared to their non-private equivalents: TLS, Searchable Encryption, and Fully Homomorphic Encryption (FHE). Our results reveal highly variable energy consumption increases, ranging from $2\times$ for TLS to $10\times$ for Searchable Encryption and $100,000\times$ for FHE. Our experiments demonstrate a simple and reproducible methodology, based on existing open-source software, to quantify the energy consumption of PETs. Finally, we discuss orthogonal research directions, such as hardware acceleration, to outline promising directions toward sustainable PETs.

This chapter is drawn from [DPP25]

13.1 Introduction

Over the last ten years, awareness about privacy issues has significantly increased. Legislations such as the European GDPR (General Data Protection Regulation) marked a turning point by incentivizing practitioners to develop applications considered as “private by design.” Pushed by these legal incentives, the research community has proposed many novel privacy-enhancing technologies (PETs); enabling to implement existing applications without requiring the user to reveal their private data.

However, information and communication technologies (ICTs) are also coping with another major challenge: reducing their environmental footprint; in particular their carbon emissions. In their fight against climate change, many governments are willing to reduce their carbon footprint, and all industries (including ICTs) attempt to contribute to the carbon emission reductions.

Several existing works started exploring this problem either by estimating the carbon footprint of the ICT industry [GKL⁺22] or by measuring the footprint

of specific applications. Measurements are essential to identify the most energy-efficient technologies in order to fulfil carbon emission reduction goals.

Related works Existing works have measured the environmental footprint of various ICTs: online advertising [ANPG14], network communications [FBL⁺21], video streaming [AMO⁺24], Machine Learning (ML) [GKS⁺22, HHR⁺20], video calls [BF22], and blockchain [BDAK21, SBFK20]. However, there has been limited research on the impact of PETs.

On the one hand, several works [MSW11, NFL⁺14] have studied the energy consumption of the TLS protocol used to secure various web protocols (including HTTP). This represents important related works as TLS is a form of “soft” PET (i.e., a PET providing data security and processing data with consent [DWS⁺11]).

On the other hand, various ML paradigms have been studied under the lens of energy consumption, including differentially private ML [NDM⁺21, dR0vE23] and Federated ML [GWMD23, SRKB22]; two paradigms related to privacy-preserving ML. Our work is complementary to these research works because we notably measure the consumption of cryptographic techniques applied to privacy-preserving ML.

Gap in the literature Despite these preliminary works, the debate about environmental sustainability is absent from the cryptography community. This absence is particularly problematic because cryptographic PETs induce computation and communication overheads compared to their “non-private” equivalents. While privacy could be seen as a value to protect at all costs, the emission reduction goals may require to design PETs offering the best trade-off between privacy and carbon emissions.

Unfortunately, the literature provides no evaluation of the environmental footprint of cryptographic PETs. Such measurements have become essential to find the best trade-off between privacy and carbon emissions.

Our contributions We present a simple and reproducible methodology (used notably in ML) to measure the energy consumption of PETs. Using this methodology, we measure the energy consumption of three PETs: TLS, Searchable Encryption, and Fully Homomorphic Encryption. These three PETs were selected because they combine practical relevance with different maturity levels. On the one hand, they have well-identified industrial applications: web browsing, encrypted databases, and privacy-preserving ML. On the other hand, they have different maturity levels from the widely deployed TLS protocol to the emerging FHE schemes. Our experiments highlight the relative increase between the PET and its non-private equivalent. We observe increases ranging from a twofold overhead with TLS to a 100,000-fold overhead with FHE. Finally, we discuss several orthogonal questions (including hardware acceleration) to identify promising directions for sustainable PET design.

Our goal is not to categorize PETs between environmentally acceptable and unacceptable. To fulfil carbon emission goals, it is not mandatory to reduce the carbon emissions of all technologies: some highly energy-consuming technologies can be considered “worth the emissions” because their service is essential. Our work simply provides orders of magnitude enabling decision makers to assess the privacy-efficiency trade-offs of specific services. Such figures have become essential in the public debate as press articles rely on estimations and measurements from the scientific literature to provide high-level perspectives on sustainability issues [Gri20].

13.2 Methodology

Our work adopts a methodology commonly used in scientific studies (notably in ML research [dR0vE23,SGJ⁺24]) to measure the energy consumption of various algorithms. This section presents the methodology, its underlying assumptions, and the general philosophy guiding its use in our study.

13.2.1 Life-Cycle Assessment

Life-Cycle Assessment (LCA) is a standard methodology [HR017] to measure the environmental footprint associated with all stages of a product’s life. Such stages can include anything from the manufacturing to the recycling. This approach enables to estimate the energy consumption, carbon emissions, and resource consumption in a comprehensive manner.

An LCA requires **defining a scope** based on an analysis goal; this will specify what is included in and excluded from the analysis. Let us assume that we want to compare the footprint of two objects. If the two objects use the same materials and end-of-life processes, the material extraction and the end of life are not necessary in the analysis. Indeed, these stages affect equally the two objects, so they will not change the comparison (i.e., our objective).

The analysis goal also defines an analysis metric which quantifies the impact of a product. The two most common metrics are the energy consumption (in kWh) and the carbon emissions (in kg eq. CO₂) [SGJ⁺24]. However, some analyses also quantify other variables such as water consumption [BF10].

LCA and digital services While LCA initially focused on physical objects, it is possible to extend this approach to digital services. However, LCA on such services focuses mostly on the service usage. For example, for an ML service, an LCA studies the impacts of the model training [HHR⁺20,LVL23,SGJ⁺24] and inference [FKW⁺24].

One may wonder why the hardware is not always included in the setup. Even though hardware is a significant part of the ICT footprint [GKL⁺22], the hardware-related costs are irrelevant in the scope of these works [HHR⁺20,SGJ⁺24]. These

works (like ours) want to identify the most energy-efficient services and execute them on a standard hardware. In other words, the hardware-related footprint is excluded because they want to minimize the footprint of the service usage.

Like most related works on the environmental footprint of ICTs [GKS⁺22, HHR⁺20] [MSW11, NDM⁺21, NFL⁺14, dROvE23, SRKB22, SGJ⁺24], our LCA focuses on the service usage (especially the energy consumption). Therefore, our experiments quantify the environmental impact in terms of **energy consumption**.

13.2.2 Software-based measurement

Our scope requires measuring the energy consumption induced by a software. A natural approach is to use a power meter. While reliable, such hardware-based approaches are inconvenient as they limit the reproduction.

On the contrary, recent works [JOL⁺23, KHN⁺18, SGJ⁺24] have promoted software-based approaches to measure the energy consumption. Software-based measurements are valuable as they simplify the reproduction of an experiment: they simply require a package installation. In particular, [KHN⁺18] showed that Intel’s Running Average Power Limit (RAPL) was a powerful tool to measure the power consumption of a machine. RAPL-based measurement consists in polling the RAPL interface every x seconds and extrapolate the consumption based on these discrete measurements. Such measurements were notably used in ML research [SGJ⁺24]. Recently, [JOL⁺23] highlighted that RAPL induces a low overhead.

Considering the advantages of RAPL, like related ML works [SGJ⁺24], we rely on this software-based approach to measure energy consumption. [SGJ⁺24] extended this approach to include GPU and RAM consumption.

From the energy consumption, note that it is possible to estimate the carbon emission. To compute the carbon emissions, one needs to multiply the energy consumption to the “carbon intensity” of the country in which the server/client is located. The carbon intensity corresponds to the amount of greenhouse gas emitted per kWh of electricity produced. This information is directly available in some public databases (such as those of the Ember and Energy Institute). In other words, the energy consumption and the energy emissions are perfectly correlated, so we do not provide it.

Indirect communication-related energy consumption Our experiments exclude the energy consumption indirectly induced by network communications. While it is straightforward to measure the energy consumption of a server or client, estimating the total energy required to transmit information between two devices is more challenging. To the best of our knowledge, no study has yet estimated the complete footprint of client-server communications, as this would require computing the amortized energy consumption of all network devices involved (e.g., routers, switches, antennas, wires) for transmitting a bit of information. [FBL⁺21] made an initial contribution by studying communication between two

servers within the same network backbone, but their work does not provide a generalizable result.

Anyway, the technologies studied in our work induce only minor communication overheads. For example, in HTTPS web browsing, the size of the ciphertext sent to the server is comparable to the plaintext, apart from a one-time secret-key exchange. Thus, their communication overhead is negligible compared to that of “communication-intensive PETs,” such as multiparty computations.

13.2.3 Is the runtime a good proxy for the energy consumption?

As research papers typically benchmark the runtime of their algorithms, one may want to extrapolate the energy consumption based on such measurements. While runtime is correlated to energy consumption, other parameters influence it. Thus, we cannot trivially deduce the energy consumption based on the runtime.

To reduce their runtime, research works (including in PETs [KÜ23,MEHP25]) parallelize their computation across multiple cores. When parallelized, the energy consumption of an algorithm is proportional to the number of cores in use. Thus, a runtime can be divided by 4, 16, or even 32 via parallelization, but the energy consumption would remain similar because spread over several cores. Similarly, powerful CPUs and GPUs can also reduce the runtime, but induce a higher energy consumption than smaller chips.

This observation shows that specialized energy measurement techniques (such as RAPL [KHN⁺18]) are essential to quantify the consumption and cannot be obtained using simpler methods such as runtime measurement. For example, RAPL-based measurements account for the energy consumption of all cores.

In conclusion, we provide a complementary perspective to existing PET papers because we report energy measurements that cannot be *precisely* extrapolated from existing experimental results. Our work echoes with recent ML work [HHR⁺20] that promoted a systematic reporting of carbon footprint and energy consumption in ML papers (in addition to existing runtime measurements).

13.3 Results

This section measures the **relative difference** between the energy consumption of three PETs and their non-private equivalents. Since it is not feasible to benchmark all PETs in a single study, we focus on three representative technologies selected for their maturity and industrial relevance: FHE, Searchable Encryption, and TLS. On the one hand, these technologies cover different maturity levels: TLS is widely deployed, Searchable Encryption has been actively studied for two decades, and FHE has grown rapidly since the introduction of more efficient schemes [CKKS17,CGGI20] less than ten years ago. On the other hand, they each have well-identified applications: encrypted machine learning, encrypted databases,

and secure web browsing. With this scope, our goal is both to initiate research on the environmental footprint of PETs and to demonstrate an accessible and reproducible methodology.

This scope naturally leaves room for future studies on other PETs, such as multiparty computation (MPC), functional encryption, or trusted execution environments (TEE). These technologies raise distinct challenges –ranging from massive communication overheads in MPC to indirect hardware-related costs in TEE (see Section 13.4.1)– that call for dedicated investigation.

13.3.1 Experimental setup

We execute our experiments on a dedicated server with 32 GB, an Intel Xeon-E3 1245 v5, and no GPU. We run the client and the server on the same machine.

As the hardware can slightly change the energy consumption of an algorithm, one may wonder whether we should execute our experiment on different devices. However, remember that we mainly want to measure the relative footprint difference between private and non-private deployments. A standard hardware change would influence the absolute values, but the relative difference should not be significantly changed. This measurement on a single hardware platform aligns with common practices in similar studies on ML algorithms [dR0vE23, GWMD23].

The only hardware that could significantly influence a PET benchmark is cryptographic accelerators. We leave them out-of-scope because a dedicated analysis is necessary to estimate the production cost of this hardware. Section 13.4.1 further discusses the use of such hardware.

We use the software CodeCarbon [SGJ⁺24] to measure the energy consumption. This software uses a RAPL-based measurement, and measures RAM, CPU, and GPU consumption. We configured CodeCarbon to poll the energy consumption every 1 ms (like in [JOL⁺23]). Our measurement aggregates the computational costs of the client and the server.

Our experiments systematically use the default configuration proposed in the documentation of the tested software. Thus, the measurements reported in our paper correspond to the expected default behavior. For implementation details, we refer to our **publicly available codebase**:

<https://github.com/MarcTOK/privacy-carbon-experiments>

13.3.2 Fully Homomorphic Encryption (FHE)

FHE is a type of cryptographic scheme [CKKS17, CGGI20] allowing to perform arithmetic operations on encrypted data. With the increasing use of ML, there is a growing concern regarding the use of personal data in these systems. To address this problem, FHE has appeared as a nice solution to execute traditional ML pipelines under encryption.

Software libraries Our goal is to measure the energy consumption of an encrypted ML operations compared to the equivalent plaintext operations. We use the software library Concrete ML developed by the company Zama. This library is stable and promoted by the company for real-world use cases. We compare the encrypted operations provided by Concrete ML to the plaintext ML performed by Scikit Learn (a popular ML framework). Note that Scikit Learn is also used as baseline by Zama in their accuracy benchmarks.

Note that Concrete ML only supports TFHE [CGGI20], but there exists another popular FHE scheme in the literature, namely CKKS [CKKS17]. Unfortunately, existing CKKS implementations are much more basic than Concrete ML, and do not support high-level operations like ML. While Concrete ML is designed for industrial applications and maintained by a company, other FHE implementations are not as mature. Thus, we only experiment on TFHE, because other implementations either provide insufficient functionalities, or are too immature (i.e., providing an unfair comparison to an industrial software like Concrete ML).

Anyway, our work does not aim to identify the best FHE implementation, or even the best privacy-preserving ML framework in general. We want to estimate an order of magnitude of the energy consumption increase induced by an FHE-based application compared to its plaintext equivalent. Moreover, the FHE literature is particularly active, so reproducibility studies would be needed in the upcoming years to measure the evolution.

Data We use Scikit Learn to generate synthetic datasets with varying number of features. In ML, a sample is a data point included in a dataset, and the features are the dimensions/characteristics of a data point.

Encrypted Machine Learning inference

Among all ML operations, ML inference (i.e., evaluating data on a trained ML model) is a particularly important operation for practitioners. Indeed, many companies have trained a powerful model (e.g., a Large Language Model) and provide “Inference as a Service”: customers send input data and the company returns the model output. To enhance privacy, researchers have designed encrypted ML frameworks [CGGI20].

Results Fig. 13.1 presents measurements for various classification and regression models. These results are average measurements over 100 samples; each sample having 30 features. On linear models (e.g., Logistic regression), the encrypted inference is $100\times$ more expensive than the plaintext inference. On tree models (Decision Tree, Random Forest, and XGB), the encrypted inference is at least $100,000\times$ more expensive than the plaintext inference. Encrypted inference on neural networks is also $100,000\times$ more expensive than plaintext.

To better understand this overhead, Fig. 13.2a studies the influence of the number

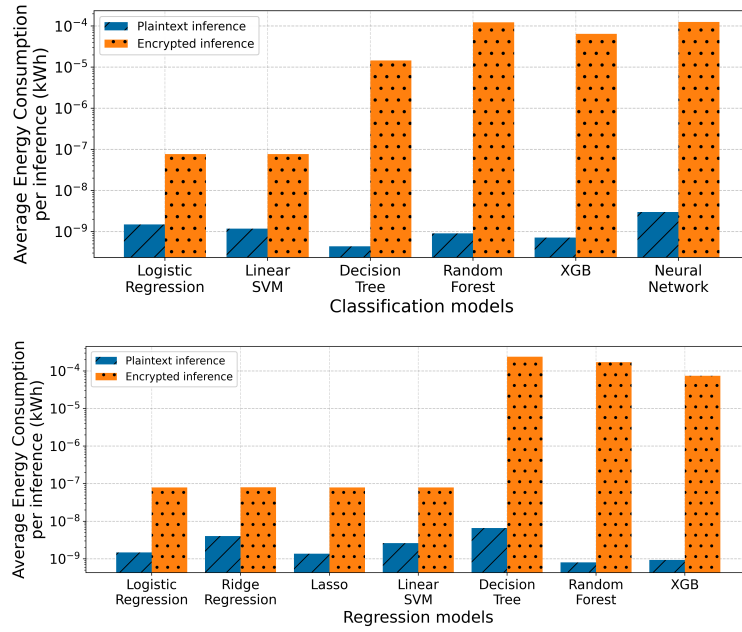


Figure 13.1 / Average energy consumption of encrypted and plaintext ML inference using various classification models (100 samples, 30 features)

of features on the consumption of three popular classification models: logistic regression, random forest, and neural networks. We observe that the scaling of the encrypted inferences does not match perfectly the scaling of the plaintext inferences. First, the encrypted logistic regression consumption increases relatively faster than the cost of the plaintext inference. Second, the encrypted neural network seems to have the same scaling pattern as the encrypted logistic regression. The encrypted neural network does not have results for more than 200 features because we stopped inferences taking more than several hours (to limit the energy consumption of our experiments). Third, the encrypted random forest consumption is approximately constant. This behavior seems to be an artifact coming from the fact that Concrete ML transforms decision trees into matrix multiplications (inducing significant constant costs).

Fig. 13.2b illustrates the consumption in function of the number of samples. On the one hand, we see that plaintext algorithms are optimized to process batches of data points: the average cost is amortized when the number of samples grows. On the other hand, we observe on the logistic regression that the encrypted inference does not provide a similar amortization. Like on Fig. 13.2a, we stopped experiments requiring several hours (which explains the smaller number of results for the random forest and neural network).

Discussions Our experiments show that encrypted ML inference induces significant overheads. However, this PET is still relatively recent, so there is room for optimization. For example, [KMY⁺24] recently presented an encrypted XGB 100× more efficient than Concrete ML’s XGB. This improvement resonates with

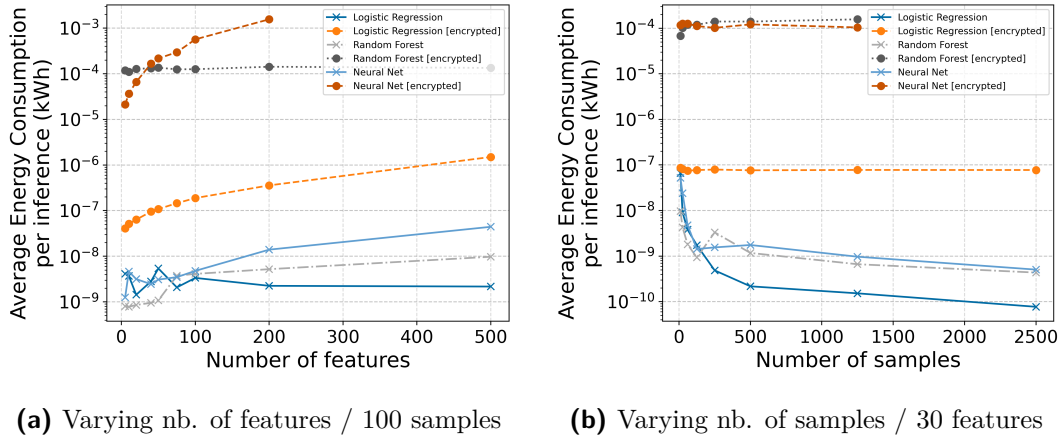


Figure 13.2 / Average energy consumption of encrypted and plaintext ML inference for varying number of features and samples

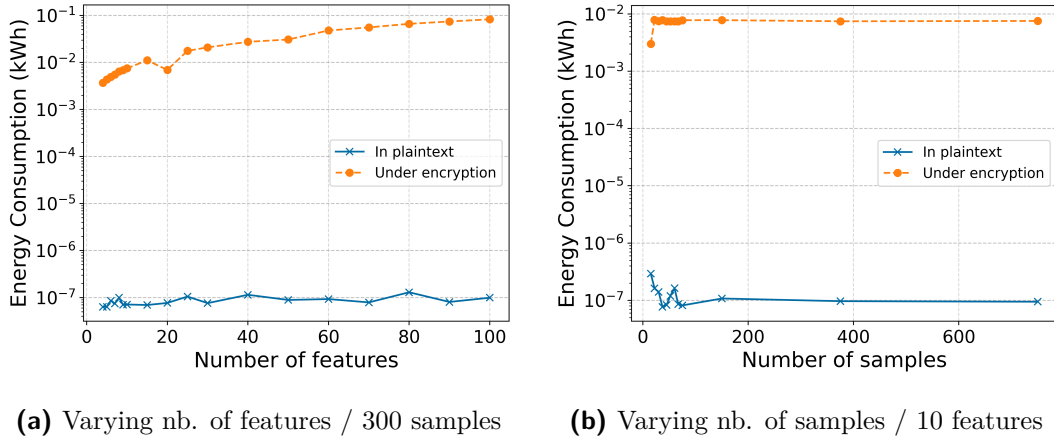


Figure 13.3 / Energy consumption of encrypted and plaintext ML training of a logistic regression for varying numbers of features and samples

Fig. 13.2a that identified a major constant factor for encrypted Random Forest (i.e., a tree-based model related to XGB). Our benchmark could be re-executed once Concrete ML integrates improved algorithms.

Encrypted Machine Learning training

Next to inference, ML training is the other operation attracting a lot of privacy concerns.

Results Fig. 13.3a compares the encrypted and plaintext training (of a logistic regression) for a varying number of features. The encrypted training is at least $100,000\times$ more expensive and its energy consumption increases faster with the number of features.

Contrary to inference, the logistic regression training is not influenced by the number of samples. Its training algorithm is iterative and process a fixed-sized

batch during each iteration. Thus, its cost only depends on the batch size and number of iterations. Fig. 13.3b confirms this behavior experimentally.

Discussions Like encrypted ML inference, secure training induces a massive consumption increase. Similarly, there is room for optimizations.

However, other (non-cryptographic) approaches to privacy-preserving ML training exist. For example, [dR0vE23] studied the energy consumption of synthetic data generation in ML context. Synthetic data generation is a pre-processing step used in privacy-preserving statistics and ML: instead of encrypting its private data, the data owner generates synthetic data based on its data and can transmit the synthetic data in plaintext. In this paradigm, the server can then process plaintext data without privacy issues as it has only access to synthetic data (and not the initial private data). Some techniques such as [JYS18] ensure that the synthetic data does not leak private information.

[dR0vE23] showed that their studied synthetic data generation requires 1 Wh to generate a synthetic dataset based on the adult dataset (14 features), and 0.01 Wh to train the logistic regression. They also used a RAPL-based energy measurement, so our results are comparable. In comparison, Fig. 13.3a shows that the encrypted ML training requires 10× more energy for a training on a similar dataset.

This comparison emphasizes that the PET choice has a major impact on the energy consumption. It also highlights an interesting future work: comparing all privacy-preserving ML paradigms to identify the most energy-efficient one.

13.3.3 Searchable Encryption

With the rise of cloud services, vast amounts of personal data are stored on outsourced databases, raising privacy concerns since providers may not be fully trusted. To solve this issue, researchers introduced searchable encryption [CGK006], a database model that enables encrypted data storage and querying while preventing the provider from accessing data or query content.

Software libraries This experiment¹ uses SWiSSSE [GPPW24] as baseline for encrypted database. SWiSSSE is an encrypted database system comparable to Redis (a traditional database specialized in key-value storage).

[GPPW24] already compared the runtime of SWiSSSE to Redis. We reproduce their benchmark using new metrics: energy.

Data Like in [GPPW24], we populate the database using the Enron email dataset.

¹Due to hardware issues (unrelated to the experiment), we executed this experiment on a MacOS machine while the other experiments are executed on a Debian server.

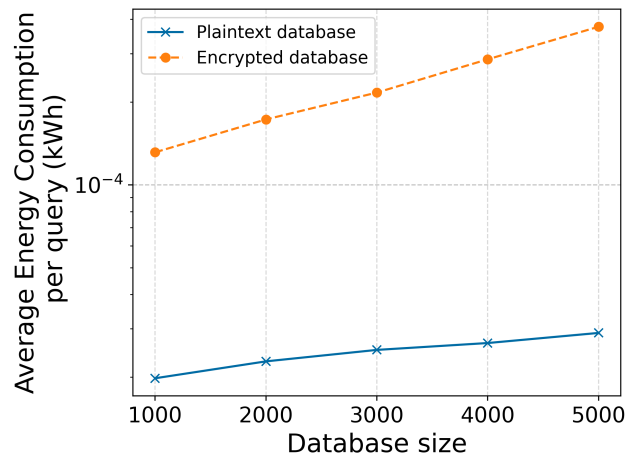


Figure 13.4 / Average energy consumption of encrypted and plaintext queries (SWiSSSE vs. Redis) for varying database sizes (1000 queries)

Results Fig. 13.4 shows the results of this benchmark for varying database sizes. We observe that the consumption of the encrypted database is nearly $10\times$ higher than the consumption of the plaintext database. Moreover, the consumption of the encrypted database increases slightly faster in function of the database size.

Discussions Our benchmark relies on SWiSSSE, which is a research prototype. Ideally, we would like to reproduce the results on professional products such as the “Queryable Encryption” plugin available in MongoDB. Unfortunately, part of this plugin requires a premium license; limiting its access and hindering reproducibility. We demonstrated our methodology on SWiSSSE, and leave for future work the evaluation of MongoDB Queryable Encryption.

13.3.4 TLS

HTTP (HyperText Transfer Protocol) is the foundational protocol used for data exchanges on the web. It operates as a plaintext protocol; making the transmitted data susceptible to eavesdropping and tampering. In contrast, HTTPS integrates HTTP with the Transport Layer Security (TLS), encrypting data to ensure confidentiality and integrity. HTTPS protects sensitive data, such as login credentials and payment information, from being intercepted or altered.

Software libraries We use NGinx as web server with TLS 1.3 and implemented a basic Python web client using the `requests` library. We compute the average energy consumption and carbon emissions over 1000 Web requests.

Data We run this experiment on five different websites: Wikipedia (Simplified English), New York Times, MDN, Mastodon Technical Blog, and xkcd. These websites are quite diverse: Wikipedia is an encyclopedia, New York Times a

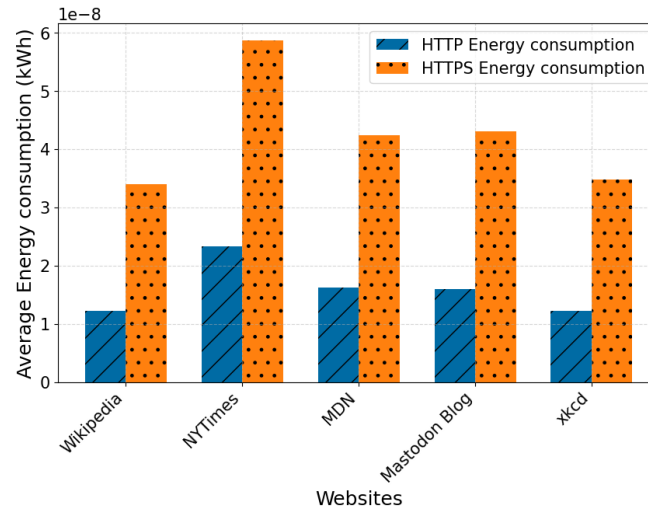


Figure 13.5 / Average energy consumption of HTTP and HTTPS requests on five different websites (1000 requests)

media, MDN a web developer documentation, Mastodon Blog is a blog, and xkcd is a minimal website publishing amusing comic strips. Wikipedia source files are publicly available, so we downloaded the “simplified English” archive from April 2007. We use the tool HTTrack to automatically download the static files (i.e., HTML, CSS, JS, and images) of the other websites.

Results Fig. 13.5 represents the average energy consumption over 1000 requests. The relative increase ranges from 152% for NYTimes to 182% for xkcd.

Unfortunately, we cannot compare our results to existing works on the energy consumption of TLS [MSW11, NFL⁺14]. While we analyze the energy consumption to estimate the energy consumption of HTTPS, these works had a completely different goal: estimating the impact of HTTPS on a smartphone battery life. Thus, these works have only measured client-side energy consumption.

Moreover, [MSW11] focused on the energy consumption of TLS only; they do not provide a relative difference between HTTP and HTTPS. Finally, [NFL⁺14] focused on specific Web applications (e.g., YouTube); contrary to our experiments that considered an easily reproducible setup: Python web client and static HTML files. [NFL⁺14] also integrated some network-related costs (e.g., Wi-Fi communications). These differences in the experimental setups prevent any naive comparison.

Discussions Our experiments only simulate static websites, and exclude any possible back-end operations (e.g., database interactions), even though such operations are common in Web applications. To assess the consumption of a Web application, an experiment would include both the cost related to HTTP(S) and the back-end costs. However, this is not our goal: we only want to compare HTTP to HTTPS. Hence, we exclude any backend work to focus solely on the footprint of HTTP data transfer.

13.4 Orthogonal discussions

13.4.1 The hidden footprint of cryptographic hardware

Energy consumption can be significantly reduced by using cryptographic accelerators [BAM21]. These chips can perform cryptographic operations much more efficiently than standard CPUs and GPUs.

However, the LCA must include the manufacturing impact, as this specialized hardware requires the production of additional hardware (in addition to the standard CPU). Since manufacturing accounts for most of the ICT sector's carbon footprint [GKL⁺22], using cryptographic accelerators does not necessarily reduce the overall carbon footprint.

Trusted Execution Environments (TEEs) [SAB15] are another example of cryptographic hardware with an indirect environmental footprint that should not be ignored. TEEs are now widely used in efficient secure protocols because they allow operations on confidential data without relying on expensive primitives like homomorphic encryption. For example, Signal uses TEEs in its Contact Discovery protocol.

As their use grows, more manufacturers now include TEEs by default in their products. However, the added manufacturing cost still represents an overhead that needs to be estimated. Like cryptographic accelerators, estimating this cost is difficult, as the necessary data is rarely made public by production facilities. This gap then represents a promising and interesting future work.

13.4.2 Privacy-Energy-Functionality trilemma

Our work initially examined the relationship between privacy and energy emissions in an isolated manner. However, our experiments indirectly revealed a broader trilemma involving privacy, energy emissions, and functionality.

The energy consumption of cryptographic PETs is tied to their functionality; more complex functionalities generally require more computationally expensive cryptographic primitives, leading to higher emissions. This phenomenon is evident in Fig. 13.1. While encryption always increases the consumption, the extent of this increase varies significantly across ML models. For instance, logistic regression exhibits a 100-fold increase, whereas encrypted neural networks result in a 100,000-fold increase. Consequently, simpler models like logistic regression lead to a smaller consumption.

Reducing functionality can therefore serve as an effective strategy to mitigate the overhead induced by the privacy enhancement.

13.4.3 Mitigating overhead via decentralization

Besides functionality, trust is another powerful leverage to optimize the privacy-energy trade-off. The PET literature usually considers two edge cases: (1) a party trusted by all users requiring no PET and (2) a zero-trust world requiring (potentially) expensive PETs.

However, decentralized social media such as Mastodon introduced a new kind of trust assumption: decentralized and personalized trust. In these social media, each user can pick a specific server (that they trust to process their personal data), and then they can interact with any user (even from other servers) thanks to the protocol ActivityPub. Such decentralization is comparable to the decentralization of the email protocol.

We can formulate the Mastodon threat model as follows: each user trusts one specific server (among all possible servers), and they do not trust anyone else. Such decentralized trust avoids expensive cryptographic operations because each server can perform plaintext operations on the personal data of its users. For instance, on Peertube (a “decentralized Youtube”), each server can offer ML-based video recommendations to its users without relying on encrypted ML.

Decentralization “à la Mastodon” is then a promising direction to enhance privacy while avoiding significant energy consumption overheads.

[MR20] presented this decentralized trust as “social approach of privacy”, opposed to the technical approach adopted by cryptographic PETs. While this “social” approach provides weaker privacy guarantees, [LW23] showed that privacy was a key factor driving the adoption of Mastodon. Thus, the weaker guarantees are not necessarily an issue for user acceptability (even among privacy-aware users).

However, one may argue that decentralized protocols require infrastructure redundancy as it requires many servers. For example, Mastodon is a decentralized social media composed of 8K servers (in September 2025). But, it does not imply that Mastodon requires 10K times more servers than a centralized social media. Indeed, centralized social media commonly rely on a global network of servers. Thus, both decentralized and centralized social media rely on a network of servers; the only difference is in the server ownership. Thus, the subtle question of infrastructure and communication overheads in decentralized protocols must be carefully studied because even large-scale centralized services rely on distributed systems creating redundancies and overheads. This calls for dedicated future work (e.g., comparing Matrix and Signal protocols in secure messaging).

13.5 Conclusion

Our work studied the energy consumption of three cryptographic PETs with different levels of maturity. Our results highlight PET-induced energy consumption increases ranging from $2\times$ (for TLS) to $100,000\times$ (for FHE). This wide range

also suggests a relationship between maturity and efficiency: the most mature technology is the most energy-efficient.

However, this correlation should not hide the fact that FHE remains several orders of magnitude more expensive than its plaintext equivalent. While performance optimization is a central goal in cryptography research, this raises a key question: what is the optimal performance? Modern FHE schemes rely on inherently expensive primitives, unlike TLS, which makes reaching comparable efficiency difficult, if not impossible. This observation should not discourage research on FHE, but it illustrates a potential broader debate for the cryptography community about the trade-off between privacy and environmental footprint.

14 Conclusion

14.1 Summary of the Contributions

This thesis has explored the challenges of processing sparse data in privacy-preserving computations, investigating the question from multiple complementary angles. Our contributions advanced both the theoretical foundations and the practical applicability of privacy-enhancing technologies. We have shown that several cryptographic primitives, including oblivious sorting, secure shuffling, and Function Secret Sharing, can be leveraged to build efficient secure operations on sparse data. Importantly, there is no single universally optimal primitive: each offers distinct advantages depending on the use case, data size, sparsity, and threat model.

First, we revisited a common assumption underlying most protocols optimized for sparse data: that access-pattern leakage must always be avoided. While leakage is necessary to sparsity-optimized protocols, the real risks associated with it had not been clearly assessed. To shed light on this issue, we turned to searchable encryption, a cryptographic scheme for which access-pattern leakage is well studied. We contributed both a new attack exploiting this leakage and statistical methods to better represent the real-world risk induced by such attacks.

Second, we proposed sorting-based protocols for secure sparse matrix multiplication. These algorithms, which naturally support an outsourced MPC setting, outperform existing solutions by several orders of magnitude. In addition, we introduced smart padding techniques to minimize volume-pattern leakage necessary to (efficient) secure sparse multiplications.

Third, we advanced the state of multi-party Function Secret Sharing, a primitive that has emerged as highly promising for private sparse data aggregation. By relying on different cryptographic assumptions (honest-majority, DDH, and LWE), we constructed schemes with more compact key sizes, making them better suited for applications in statistics and machine learning.

Finally, we broadened the scope of the thesis with two contributions addressing practical aspects of deploying privacy-enhancing technologies (beyond data sparsity). On the one hand, we introduced *Fedivertex*, a new graph dataset supporting better benchmarks of decentralized protocols for privacy-preserving machine learning. On the other hand, we measured the energy consumption increase by three cryptographic privacy-enhancing technologies.

14.2 Promising Future Works

Throughout this thesis, we have identified several promising directions for extending our results. This subsection highlights four of the most notable ones.

Sparse-dense multiplications Our sorting-based protocols are highly efficient for multiplying two sparse matrices or vectors. However, many ML workloads rely instead on sparse-dense multiplications, for instance when training a neural network on sparse input data. While our current protocols can be applied in this setting, their performance gains vanish, since efficiency is inverse correlated to the number of zeros; dense matrices containing none. As in plaintext computation, separating algorithms for sparse-sparse and sparse-dense operations becomes a natural necessity in the secure setting.

Designing optimized protocols for sparse-dense multiplication thus represents an important next step. This challenge goes beyond a straightforward adaptation of sparse-sparse methods. New protocols (potentially not based on oblivious sorting) are required to perform these secure operations efficiently. This would align with plaintext algorithms that rely on fairly different approaches for sparse-sparse and sparse-dense operations [BGS11, Duf77].

Systematic comparison of FSS schemes Over the past decade, FSS has emerged as a powerful primitive with many applications. Since [BGI15], extending the efficiency of the two-party DPF to the multi-party setting has been an evident goal. The past two years have seen active research in this direction, with several concurrent works, including ours and [CKY25, KPM24, GWW25, KP25]. Our benchmarks already revealed that some recent schemes, such as [GWW25], perform poorly in practice, while others, like the information-theoretic scheme of [BK022], are surprisingly competitive. Yet systematic comparisons are lacking.

The field now needs a comprehensive, standardized evaluation that considers not only key sizes but also generation and evaluation costs. Such a study should include the latest two- and three-party schemes, in order to quantify the remaining performance gap between two-party and multi-party schemes. Although [CKY25, KP25] have matched the asymptotic key size of two-party schemes, practical overheads may still hinder efficiency.

Decentralized ML for decentralized social media With *Fedivertex*, we highlighted decentralized social media as an almost unexplored but highly promising application area for decentralized ML. While existing papers have often been motivated with hypothetical applications involving smartphones, decentralized social media like Mastodon is naturally adequate for decentralized ML because the servers are already connected via the ActivityPub protocol. Unlike smartphone-based scenarios, such use cases do not require setting up complicated peer-to-peer

communications. Moreover, decentralized social media may need ML to automatize and optimize operations like moderation or recommendation.

Fedivertex offers a realistic graph dataset for benchmarking decentralized ML tasks such as spam detection or hate-speech moderation in these networks. We hope this contribution will stimulate further research related to decentralized social media within the ML and PETs communities.

14.3 Broader Perspectives

While the previous section presented direct follow-up works of our contributions, we conclude with broader open problems related to our work.

Supporting real-world SSE deployments With the release of enterprise-grade implementations such as MongoDB Queryable Encryption, the conditions are now favorable for real-world SSE deployment. As with any secure protocol, the existence of potential attacks naturally raises concerns among practitioners. However, recent works (including ours and [BGW24, KM23, WXC⁺25]) have introduced statistical tools that enable a more rigorous assessment and control of these risks in practice. It is worth emphasizing that, even in the presence of possible leakage-abuse attacks, SSE still provides substantially stronger privacy guarantees than plaintext database systems.

While many remaining challenges are engineering-related and fall to industry, the scientific community still plays a crucial role in transferring cryptographic research into practice. One important direction is supporting more expressive queries. Historically, most research focused on simplified database models such as single-keyword search or one-dimensional range queries. These abstractions have been valuable to explore efficiency-privacy trade-offs, but practitioners typically need far richer queries. Recent works have started addressing this gap, for instance with conjunctive queries [WSW⁺24] and graph queries [FGPT24], but further progress is needed to support the variety of queries required in practice.

Another key task for researchers is auditing real-world implementations. As research prototypes transition into production systems, new attack surfaces can emerge. Works such as the security analysis of MongoDB Queryable Encryption by [GPT23] illustrate the importance of such efforts. Ensuring that encrypted databases are secure both in theory *and* in practice will remain a critical responsibility for the community.

Practical Oblivious RAM ORAM has been studied by cryptographers for already three decades [G096]. Its ability to emulate arbitrary data-dependent memory access make it *theoretically* very appealing as it offers the possibility to build MPC protocols for *any* plaintext algorithm. As discussed in Chapter 4, ORAM can in particular be used to implement sparse operations.

Despite recent advances [DS17,F0SZ23], existing schemes remain expensive. While modern ORAM protocols are far more efficient than early proposals, they are not yet practical for most real-world use cases. The cryptographic community continues relying on other primitives (such as oblivious sorting in Chapter 8) to build secure MPC protocols for plaintext algorithms that require data-dependent memory accesses.

For ORAM to become practical and widely used in MPC, especially on sparse data, its efficiency must be improved significantly. This is a challenging research direction, as ORAM has already been extensively studied. Nonetheless, progress here would have a massive impact (far beyond sparse computations), enabling secure and general-purpose computations.

Privacy-Carbon-Footprint trade-off Finally, Chapter 13 opened a long-avoided discussion in the PETs and cryptography communities: is stronger privacy always worth the environmental cost? This question goes beyond the focus of the thesis on cryptographic protocols for sparse data. Indeed, it concerns all PETs; protocols on sparse data being one of the many PETs described in the scientific literature.

Traditionally, cryptographers have designed increasingly sophisticated schemes to hide ever more information. Yet, with growing pressure to reduce carbon emissions, the overheads of such schemes cannot be ignored. For example, our comparison with [dR0vE23] showed that training a logistic regression with state-of-the-art FHE was 10× more energy-intensive than with synthetic data generation. Although these techniques offer different privacy guarantees, such disparities demand a broader reflection: how should we balance privacy guarantees against sustainability concerns?

Our results provide only a preliminary step, but they call for an interdisciplinary dialogue involving not only cryptographers, but also social scientists, policymakers, and civil society. Similar discussions have already emerged in other scientific domains (including in computer science), and it is both timely and necessary that they reach the cryptography community.



Acronyms

DC: Dining Cryptographers [problem]
DDH: Decisional Diffie-Hellman [assumption]
DP: Differential Privacy
ECC: Elliptic Curve Cryptography
FHE: Fully Homomorphic Encryption
FL: Federated Learning
FSS: Function Secret sharing
ICT: Information and Communication Technology
LCA: Life-Cycle Assessment
LWE: Learning With Error [assumption]
LWR: Learning With Rounding [assumption]
ML: Machine Learning
MPC: Multi-Party Computation
ORAM: Oblivious RAM
PET: Privacy-Enhancing Technology
PIR: Private Information Retrieval
PPC: Privacy-Preserving Computations
PR(N)G: PseudoRandom (Number) Generator
PPML: Privacy-Preserving Machine Learning
S(S)E: Searchable (Symmetric) Encryption
TEE: Trusted Execution Environment
TLS: Transport Layer Security
ZKP: Zero-Knowledge Proof

LLM Usage Disclosure

As in many other fields of work, my thesis has also been influenced by the advent of Large Language Models (LLMs). Following practices that have emerged in the ML community, I consider it important to disclose this usage and to explicitly state the use cases.

I have only used LLMs to polish my writing. In other words, when needed, I provided an LLM with a draft paragraph in order to check grammar and possibly improve the flow. I want to emphasize that I have *never* asked an LLM to generate a paragraph from scratch for my thesis; I have always provided draft text which the LLM was meant to polish.

Thus, the arguments and ideas presented in my chapters are entirely my own. I have often felt the need to use LLMs because, as a non-native English speaker, my style can sometimes be “clumsy”. I mainly used these tools to refine introductions, as they are essential to capture the reader’s attention and spark interest in the chapter or paper. Other sections, such as experimental results or algorithm descriptions, were rarely polished with LLMs, since these paragraphs follow a more fixed and straightforward structure.

As the polished paragraphs remain, at most, slight reformulations of my original drafts, I take full responsibility for all the content of this thesis. Even when using LLMs, I have systematically reviewed and reworked the polished texts; LLMs often adopt a non-academic tone that requires simplification or adjustment.

In addition to text polishing, I also used LLMs to generate basic scripts, for example for figure generation, since software libraries such as Matplotlib can sometimes be cryptic.

Naturally, I have *not* used LLMs to design new algorithms or provide proofs on my behalf. All chapters are based on my own ideas, developed and improved with the help of my collaborators (the contributions are specified in Introduction).

Over the course of my Ph.D., I experimented with different models, in particular those released by OpenAI, DeepSeek, and Mistral.

References

- [AAL18] Dan Alistarh, Zeyuan Allen-Zhu, and Jerry Li. Byzantine Stochastic Gradient Descent. IN: *Advances in Neural Information Processing Systems*, 2018. 48
- [ABG⁺24] Amit Agarwal, Elette Boyle, Niv Gilboa, Yuval Ishai, Mahimna Kelkar, and Yiping Ma. Compressing Unit-Vector Correlations via Sparse Pseudorandom Generators. IN: *Advances in Cryptology – CRYPTO 2024*, 2024. ISBN: 978-3-031-68397-8. DOI: [10.1007/978-3-031-68397-8_11](https://doi.org/10.1007/978-3-031-68397-8_11) 18
- [ABG⁺25] Anders Aamand, Fabrizio Boninsegna, Abigail Gentle, Jacob Imola, and Rasmus Pagh. Lightweight protocols for distributed private quantile estimation. IN: *Forty-second International Conference on Machine Learning*, 2025. 64
- [ABGS23] Diego F. Aranha, Carsten Baum, Kristian Gjøsteen, and Tjerand Silde. Verifiable Mix-Nets and Distributed Decryption for Voting from Lattice-Based Assumptions. IN: *Proceedings of the 2023 ACM SIGSAC Conference on Computer and Communications Security*, November 2023. ISBN: 979-8-4007-0050-7. DOI: [10.1145/3576915.3616683](https://doi.org/10.1145/3576915.3616683) 52, 53, 58
- [ACG⁺16] Martín Abadi, Andy Chu, Ian Goodfellow, H. Brendan McMahan, Ilya Mironov, Kunal Talwar, and Li Zhang. Deep Learning with Differential Privacy. IN: *Proceedings of the 2016 ACM SIGSAC Conference on Computer and Communications Security*, 2016. DOI: [10.1145/2976749.2978318](https://doi.org/10.1145/2976749.2978318) 9
- [ACL01] William Aiello, Fan Chung, and Linyuan Lu. A Random Graph Model for Power Law Graphs. *Experimental Mathematics*, 2001. ISSN: 1058-6458. DOI: [10.1080/10586458.2001.10504428](https://doi.org/10.1080/10586458.2001.10504428) 23
- [Adi08] Ben Adida. Helios: Web-based Open-Audit Voting. IN: *17th USENIX Security Symposium (USENIX Security 08)*, 2008. 47
- [AFK⁺25] Hilal Asi, Vitaly Feldman, Hannah Keller, Guy N. Rothblum, and Kunal Talwar. PREAMBLE: Private and Efficient Aggregation of Block Sparse Vectors and Applications, 2025. xv, 15, 20, 25, 158

- [AFO⁺21] Toshinori Araki, Jun Furukawa, Kazuma Ohara, Benny Pinkas, Hanan Rosemarin, and Hikaru Tsuchida. *Secure Graph Analysis at Scale*. IN: *Proceedings of the 2021 ACM SIGSAC Conference on Computer and Communications Security*, 2021. ISBN: 978-1-4503-8454-4.
DOI: [10.1145/3460120.3484560](https://doi.org/10.1145/3460120.3484560) 14
- [Agg16] Charu C. Aggarwal. *Recommender Systems*. 2016. ISBN: 978-3-319-29657-9.
DOI: [10.1007/978-3-319-29659-3](https://doi.org/10.1007/978-3-319-29659-3) 126
- [AHI⁺22] Gilad Asharov, Koki Hamada, Dai Ikarashi, Ryo Kikuchi, Ariel Nof, Benny Pinkas, Katsumi Takahashi, and Junichi Tomida. *Efficient Secure Three-Party Sorting with Applications to Data Analysis and Heavy Hitters*. IN: *Proceedings of the 2022 ACM SIGSAC Conference on Computer and Communications Security*, 2022. ISBN: 978-1-4503-9450-5.
DOI: [10.1145/3548606.3560691](https://doi.org/10.1145/3548606.3560691) xv, 14, 24, 26, 27, 115, 117, 118, 123
- [AKK⁺23] Pranav Shriram A, Nishat Koti, Varsha Bhat Kukkala, Arpita Patra, Bhavish Raj Gopal, and Somya Sangal. *Ruffle: Rapid 3-Party Shuffle Protocols*. *Proceedings on Privacy Enhancing Technologies*, 2023. ISSN: 2299-0984.
52, 53, 59
- [AKL⁺20] Gilad Asharov, Ilan Komargodski, Wei-Kai Lin, Kartik Nayak, Enoch Peserico, and Elaine Shi. *OptORAMA: Optimal Oblivious RAM*. IN: *Advances in Cryptology – EUROCRYPT 2020*, 2020. ISBN: 978-3-030-45724-2.
DOI: [10.1007/978-3-030-45724-2_14](https://doi.org/10.1007/978-3-030-45724-2_14) 36
- [ALPH01] Lada A. Adamic, Rajan M. Lukose, Amit R. Puniyani, and Bernardo A. Huberman. *Search in power-law networks*. *Physical Review E*, 2001.
DOI: [10.1103/PhysRevE.64.046135](https://doi.org/10.1103/PhysRevE.64.046135) 23
- [AM22] Jacopo Anderlini and Carlo Milani. *Emerging Forms of Sociotechnical Organisation: The Case of the Fediverse*. IN: *Digital Platforms and Algorithmic Subjectivities*, 2022. ISBN: 978-1-914386-12-1.
186
- [AMO⁺24] Samira Afzal, Narges Mehran, Zoha Azimi Ourimi, Farzad Tashtarian, Hadi Amirpour, Radu Prodan, and Christian Timmerer. *A Survey on Energy Consumption and Environmental Impact of Video Streaming*, January 2024.
DOI: [10.48550/arXiv.2401.09854](https://doi.org/10.48550/arXiv.2401.09854) 200
- [ANPG14] Abdurhman Albasir, Kshirasagar Naik, Bernard Plourde, and

- Nishith Goel. Experimental study of energy and bandwidth costs of web advertisements on smartphones. IN: *6th International Conference on Mobile Computing, Applications and Services*, November 2014.
DOI: [10.4108/icst.mobicaase.2014.257770](https://doi.org/10.4108/icst.mobicaase.2014.257770) 200
- [APS15] Martin R. Albrecht, Rachel Player, and Sam Scott. On the concrete hardness of Learning with Errors. *Journal of Mathematical Cryptology*, October 2015. ISSN: 1862-2984.
DOI: [10.1515/jmc-2015-0016](https://doi.org/10.1515/jmc-2015-0016) 174
- [APY20] Ittai Abraham, Benny Pinkas, and Avishay Yanai. Blinder – Scalable, Robust Anonymous Committed Broadcast. IN: *Proceedings of the 2020 ACM SIGSAC Conference on Computer and Communications Security*, October 2020. ISBN: 978-1-4503-7089-9.
DOI: [10.1145/3372297.3417261](https://doi.org/10.1145/3372297.3417261) 48, 49, 52, 53, 59
- [ARS⁺24] Vibhor Agarwal, Aravindh Raman, Nishanth Sastry, Ahmed M Abdelmoniem, Gareth Tyson, and Ignacio Castro. Decentralised moderation for interoperable social networks: a conversation-based approach for Pleroma and the Fediverse. IN: *Conference on Artificial Intelligence (AAAI)*, 2024. 187
- [AYR15] Daniel Arp, Fabian Yamaguchi, and Konrad Rieck. Torben: A Practical Side-Channel Attack for Deanonymizing Tor Communication. IN: *Proceedings of the 10th ACM Symposium on Information, Computer and Communications Security*, 2015. ISBN: 978-1-4503-3245-3.
DOI: [10.1145/2714576.2714627](https://doi.org/10.1145/2714576.2714627) 50, 56
- [BAM21] Mojtaba Bisheh-Niasar, Reza Azarderakhsh, and Mehran Mozaffari-Kermani. Cryptographic Accelerators for Digital Signature Based on Ed25519. *IEEE Transactions on Very Large Scale Integration (VLSI) Systems*, July 2021. ISSN: 1557-9999.
DOI: [10.1109/TVLSI.2021.3077885](https://doi.org/10.1109/TVLSI.2021.3077885) 211
- [BAT21] Constance Beguier, Mathieu Andreux, and Eric W. Tramel. Efficient Sparse Secure Aggregation for Federated Learning, 2021. 28, 115
- [BAT24] Tariq Bontekoe, Hassan Jameel Asghar, and Fatih Turkmen. Efficient Verifiable Differential Privacy with Input Authenticity in the Local and Shuffle Model, 2024. 72
- [BBC⁺13] Naomi Benger, David Bernhard, Dario Catalano, Manuel Charlemagne, David Conti, Biljana Cubaleska, Hernando Fernando, Dario Fiore, Steven Galbraith, David Galindo, et al. Final Report on

Main Computational Assumptions in Cryptography II. technical report, ICT-2007-216676 D. MAYA. 6. European Network of Excellence in Cryptology, 2013.

3

- [BBC⁺21] Dan Boneh, Elette Boyle, Henry Corrigan-Gibbs, Niv Gilboa, and Yuval Ishai. **Lightweight Techniques for Private Heavy Hitters**. IN: *2021 IEEE Symposium on Security and Privacy (SP)*, 2021.
DOI: [10.1109/SP40001.2021.00048](https://doi.org/10.1109/SP40001.2021.00048) 15, 17, 19, 155, 157, 158
- [BBG⁺20] James Henry Bell, Kallista A. Bonawitz, Adrià Gascón, Tancrede Lepoint, and Mariana Raykova. **Secure Single-Server Aggregation with (Poly)Logarithmic Overhead**. IN: *Proceedings of the 2020 ACM SIGSAC Conference on Computer and Communications Security*, 2020. ISBN: 978-1-4503-7089-9.
DOI: [10.1145/3372297.3417885](https://doi.org/10.1145/3372297.3417885) 49, 51, 52, 53, 60
- [BBG23] Borja Balle, James Bell, and Adrià Gascón. **Amplification by Shuffling without Shuffling**. IN: *Proceedings of the 2023 ACM SIGSAC Conference on Computer and Communications Security*, November 2023. ISBN: 979-8-4007-0050-7.
DOI: [10.1145/3576915.3623215](https://doi.org/10.1145/3576915.3623215) 58, 71, 74
- [BBGN19a] Borja Balle, James Bell, Adria Gascon, and Kobbi Nissim. **Improved Summation from Shuffling**. *arXiv:1909.11225 [cs]*, 2019.
41, 62, 64
- [BBGN19b] Borja Balle, James Bell, Adrià Gascón, and Kobbi Nissim. **The privacy blanket of the shuffle model**. IN: *Advances in Cryptology – CRYPTO 2019*, 2019. ISBN: 978-3-030-26951-7.
DOI: [10.1007/978-3-030-26951-7_22](https://doi.org/10.1007/978-3-030-26951-7_22) 41, 63
- [BBGN20] Borja Balle, James Bell, Adrià Gascón, and Kobbi Nissim. **Private Summation in the Multi-Message Shuffle Model**. IN: *Proceedings of the 2020 ACM SIGSAC Conference on Computer and Communications Security*, 2020. ISBN: 978-1-4503-7089-9.
DOI: [10.1145/3372297.3417242](https://doi.org/10.1145/3372297.3417242) 64
- [BC20] Victor Balcer and Albert Cheu. **Separating Local & Shuffled Differential Privacy via Histograms**. IN: *1st Conference on Information-Theoretic Cryptography*, 2020.
64
- [BC22] Ari Biswas and Graham Cormode. **Verifiable Differential Privacy For When The Curious Become Dishonest**, 2022.
72
- [BCGI22] Elette Boyle, Geoffroy Couteau, Niv Gilboa, and Yuval Ishai. **Function Secret Sharing and Homomorphic Secret Sharing**, 2022.

- 15, 16, 17, 138, 147, 149
- [BCJM21] Victor Balcer, Albert Cheu, Matthew Joseph, and Jieming Mao. Connecting robust shuffle privacy and pan-privacy. IN: *Proceedings of the 2021 ACM-SIAM Symposium on Discrete Algorithms (SODA)*, 2021. 64
- [BCV16] Mark Bun, Yi-Hsiu Chen, and Salil Vadhan. Separating Computational and Statistical Differential Privacy in the Client-Server Model. IN: *Theory of Cryptography*, 2016. ISBN: 978-3-662-53641-4. DOI: [10.1007/978-3-662-53641-4_23](https://doi.org/10.1007/978-3-662-53641-4_23) 72
- [BDK21] Abigael Okikijesu Bada, Amalia Damianou, Constantinos Marios Angelopoulos, and Vasilios Katos. Towards a Green Blockchain: A Review of Consensus Mechanisms and their Energy Consumption. IN: *2021 17th International Conference on Distributed Computing in Sensor Systems (DCOSS)*, July 2021. DOI: [10.1109/DCOSS52077.2021.00083](https://doi.org/10.1109/DCOSS52077.2021.00083) 200
- [BEM⁺17] Andrea Bittau, Ulfar Erlingsson, Petros Maniatis, Ilya Mironov, Ananth Raghunathan, David Lie, Mitch Rudominer, Ushasree Kode, Julien Tinnes, and Bernhard Seefeld. Prochlo: Strong Privacy for Analytics in the Crowd. IN: *Proceedings of the 26th Symposium on Operating Systems Principles*, 2017. ISBN: 978-1-4503-5085-3. DOI: [10.1145/3132747.3132769](https://doi.org/10.1145/3132747.3132769) 41, 42, 44, 52, 53, 59, 60, 73, 74
- [BF10] Markus Berger and Matthias Finkbeiner. Water Footprinting: How to Address Water Use in Life Cycle Assessment? *Sustainability*, April 2010. ISSN: 2071-1050. DOI: [10.3390/su2040919](https://doi.org/10.3390/su2040919) 201
- [BF17] Raphael Bost and Pierre-Alain Fouque. Thwarting Leakage Abuse Attacks against Searchable Encryption – A Formal Approach and Applications to Database Padding, 2017. 105
- [BF22] Francoise Berthoud and Marion Ficher. Évaluation de l’empreinte carbone d’une visioconférence entre deux utilisateurs du service rendez-vous. report, CNRS - EcoInfo, March 2022. 200
- [BG12] Stephanie Bayer and Jens Groth. Efficient Zero-Knowledge Argument for Correctness of a Shuffle. IN: *Advances in Cryptology – EUROCRYPT 2012*, 2012. DOI: [10.1007/978-3-642-29011-4_17](https://doi.org/10.1007/978-3-642-29011-4_17) 52, 53, 58, 61
- [BGG⁺22] James Bell, Adrià Gascón, Badih Ghazi, Ravi Kumar, Pasin Manurangsi, Mariana Raykova, and Phillipp Schoppmann. Distributed,

- Private, Sparse Histograms in the Two-Server Model. IN: *Proceedings of the 2022 ACM SIGSAC Conference on Computer and Communications Security*, 2022. ISBN: 978-1-4503-9450-5.
DOI: [10.1145/3548606.3559383](https://doi.org/10.1145/3548606.3559383) [xv](#), [24](#), [64](#), [115](#), [118](#)
- [BGH⁺25] Elette Boyle, Niv Gilboa, Matan Hamilis, Yuval Ishai, and Yaxin Tu. Improved Constructions for Distributed Multi-Point Functions. IN: *2025 IEEE Symposium on Security and Privacy (SP)*, 2025. ISBN: 979-8-3315-2236-0.
DOI: [10.1109/SP61157.2025.00044](https://doi.org/10.1109/SP61157.2025.00044) ISSN: 2375-1207 [xv](#), [15](#), [24](#), [25](#)
- [BGI15] Elette Boyle, Niv Gilboa, and Yuval Ishai. Function Secret Sharing. IN: *Advances in Cryptology - EUROCRYPT 2015*, 2015. ISBN: 978-3-662-46803-6.
DOI: [10.1007/978-3-662-46803-6_12](https://doi.org/10.1007/978-3-662-46803-6_12) [xvii](#), [14](#), [15](#), [16](#), [17](#), [18](#), [20](#), [133](#), [134](#), [135](#), [136](#), [137](#), [138](#), [139](#), [145](#), [146](#), [147](#), [148](#), [149](#), [150](#), [152](#), [154](#), [155](#), [159](#), [160](#), [161](#), [162](#), [163](#), [164](#), [167](#), [174](#), [180](#), [181](#), [215](#)
- [BGI16] Elette Boyle, Niv Gilboa, and Yuval Ishai. Function Secret Sharing: Improvements and Extensions. IN: *Proceedings of the 2016 ACM SIGSAC Conference on Computer and Communications Security*, 2016. ISBN: 978-1-4503-4139-4.
DOI: [10.1145/2976749.2978429](https://doi.org/10.1145/2976749.2978429) [16](#), [17](#), [161](#)
- [BGI19] Elette Boyle, Niv Gilboa, and Yuval Ishai. Secure Computation with Preprocessing via Function Secret Sharing. IN: *Theory of Cryptography*, 2019. ISBN: 978-3-030-36030-6.
DOI: [10.1007/978-3-030-36030-6_14](https://doi.org/10.1007/978-3-030-36030-6_14) [14](#), [16](#), [138](#)
- [BGIK22] Elette Boyle, Niv Gilboa, Yuval Ishai, and Victor I. Kolobov. Programmable Distributed Point Functions. IN: *Advances in Cryptology - CRYPTO 2022*, 2022.
[17](#)
- [BGL⁺22] James Bell, Adrià Gascón, Tancrede Lepoint, Baiyu Li, Sarah Meiklejohn, Mariana Raykova, and Cathie Yun. ACORN: Input Validation for Secure Aggregation, 2022.
[49](#), [73](#)
- [BGLL08] Vincent D Blondel, Jean-Loup Guillaume, Renaud Lambiotte, and Etienne Lefebvre. Fast unfolding of communities in large networks. *Journal of Statistical Mechanics: Theory and Experiment*, October 2008. ISSN: 1742-5468.
[197](#)
- [BGPS06] Stephen Boyd, Arpita Ghosh, Balaji Prabhakar, and Devavrat Shah. Randomized gossip algorithms. *IEEE transactions on information theory*, 2006.

- 186
- [BGS11] Aydın Buluç, John Gilbert, and Viral B. Shah. Implementing Sparse Matrices for Graph Algorithms. IN: *Graph Algorithms in the Language of Linear Algebra*, 2011. ISBN: 978-0-89871-991-8.
DOI: [10.1137/1.9780898719918.ch13](https://doi.org/10.1137/1.9780898719918.ch13) xiv, 22, 116, 121, 215
- [BGW24] Alexandra Boldyreva, Zichen Gui, and Bogdan Warinschi. Understanding Leakage in Searchable Encryption: a Quantitative Approach. *Proceedings on Privacy Enhancing Technologies*, October 2024. ISSN: 2299-0984.
DOI: [10.56553/popets-2024-0127](https://doi.org/10.56553/popets-2024-0127) 30, 36, 38, 96, 216
- [BHJ09] Mathieu Bastian, Sebastien Heymann, and Mathieu Jacomy. Gephi: an open source software for exploring and manipulating networks. IN: *Proceedings of the international AAAI conference on web and social media*, 2009. 189
- [BHJP14] Christoph Bösch, Pieter Hartel, Willem Jonker, and Andreas Peter. A Survey of Provably Secure Searchable Encryption. *ACM Computing Surveys*, August 2014. ISSN: 0360-0300.
DOI: [10.1145/2636328](https://doi.org/10.1145/2636328) xv
- [BHMS21] Benedikt Bünz, Yuncong Hu, Shin'ichiro Matsuo, and Elaine Shi. Non-Interactive Differentially Anonymous Router, 2021. 52, 53, 60, 74
- [BKKO20] Paul Bunn, Jonathan Katz, Eyal Kushilevitz, and Rafail Ostrovsky. Efficient 3-Party Distributed ORAM. IN: *Security and Cryptography for Networks*, 2020. ISBN: 978-3-030-57990-6.
DOI: [10.1007/978-3-030-57990-6_11](https://doi.org/10.1007/978-3-030-57990-6_11) 17
- [BKM⁺13] Michael Backes, Aniket Kate, Praveen Manoharan, Sebastian Meiser, and Esfandiar Mohammadi. AnoA: A Framework for Analyzing Anonymous Communication Protocols. IN: *2013 IEEE 26th Computer Security Foundations Symposium*, 2013.
DOI: [10.1109/CSF.2013.18](https://doi.org/10.1109/CSF.2013.18) 45, 70
- [BKM⁺20] Borja Balle, Peter Kairouz, Brendan McMahan, Om Thakkar, and Abhradeep Guha Thakurta. Privacy amplification via random checks. *Advances in Neural Information Processing Systems*, 2020. 6
- [BKM⁺22] Eugene Bagdasaryan, Peter Kairouz, Stefan Mellem, Adrià Gascón, Kallista Bonawitz, Deborah Estrin, and Marco Gruteser. Towards Sparse Federated Analytics: Location Heatmaps under Distributed Differential Privacy with Secure Aggregation. *Proceedings on Privacy Enhancing Technologies*, 2022. ISSN: 2299-0984.

- 42
- [BKM20] Laura Blackstone, Seny Kamara, and Tarik Moataz. Revisiting Leakage Abuse Attacks. IN: *Network and Distributed System Security Symposium (NDSS)*, 2020.
- 38, 39, 77, 79, 91, 94, 111, 112
- [BKMR24] Beatrice Biasioli, Elena Kirshanova, Chiara Marcolla, and Sergi Rovira. A Tool for Fast and Secure LWE Parameter Selection: the FHE case, 2024.
- 174
- [BKO22] Paul Bunn, Eyal Kushilevitz, and Rafail Ostrovsky. CNF-FSS and Its Applications. IN: *Public-Key Cryptography – PKC 2022*, 2022.
DOI: 10.1007/978-3-030-97121-2_11
- 17, 18, 20, 133, 138, 139, 140, 141, 142, 143, 145, 146, 147, 149, 150, 151, 152, 153, 154, 155, 161, 180, 181, 215
- [BL⁺07] James Bennett, Stan Lanning, et al. The netflix prize. IN: *Proceedings of KDD cup and workshop*, 2007.
- 23
- [BLAB21] Antoine Boutet, Thomas Lebrun, Jan Aalmoes, and Adrien Baud. MixNN: Protection of Federated Learning Against Inference Attacks by Mixing Neural Network Layers, 2021.
- 73
- [BLMR13] Dan Boneh, Kevin Lewi, Hart Montgomery, and Ananth Raghunathan. Key Homomorphic PRFs and Their Applications. IN: *Advances in Cryptology – CRYPTO 2013*, 2013. ISBN: 978-3-642-40041-4.
DOI: 10.1007/978-3-642-40041-4_23
- 164, 166, 171, 172, 183
- [BMD⁺17] Ferdinand Brasser, Urs Müller, Alexandra Dmitrienko, Kari Kostainen, Srdjan Capkun, and Ahmad-Reza Sadeghi. Software grand exposure: SGX cache attacks are practical. IN: *11th USENIX Workshop on Offensive Technologies (WOOT 17)*, 2017.
- 73
- [BMR25] Antoine Barczewski, Amal Mawass, and Jan Ramon. Differentially Private Empirical Cumulative Distribution Functions, February 2025.
DOI: 10.48550/arXiv.2502.06651
- 16, 19, 157, 158
- [BMW24] Alexander Burton, Samir Jordan Menon, and David J. Wu. Respire: High-Rate PIR for Databases with Small Records. IN: *Proceedings of the 2024 on ACM SIGSAC Conference on Computer and Communications Security*, December 2024. ISBN: 979-8-4007-0636-3.
DOI: 10.1145/3658644.3690328
- 32, 33
- [BNO08] Amos Beimel, Kobbi Nissim, and Eran Omri. Distributed Private

- Data Analysis: Simultaneously Solving How and What. IN: *Advances in Cryptology – CRYPTO 2008*, 2008.
DOI: [10.1007/978-3-540-85174-5_25](https://doi.org/10.1007/978-3-540-85174-5_25)
ISSN: 0302-9743, 1611-3349 Series Title: Lecture Notes in Computer Science 71
- [Bos16] Raphael Bost. Sophos: Forward Secure Searchable Encryption. IN: *2016 ACM SIGSAC Conference on Computer and Communications Security*, October 2016. ISBN: 978-1-4503-4139-4.
DOI: [10.1145/2976749.2978303](https://doi.org/10.1145/2976749.2978303) 37
- [Box76] George E. P. Box. Science and Statistics. *Journal of the American Statistical Association*, December 1976. ISSN: 0162-1459.
DOI: [10.1080/01621459.1976.10480949](https://doi.org/10.1080/01621459.1976.10480949) 104
- [BP16] Albert-László Barabási and Márton Pósfai. *Network science*. 2016. ISBN: 9781107076266. 193
- [BRC⁺22] Haris Bin Zia, Aravindh Raman, Ignacio Castro, Ishaku Hassan Anaobi, Emiliano De Cristofaro, Nishanth Sastry, and Gareth Tyson. Toxicity in the Decentralized Web and the Potential for Model Sharing. *Proc. ACM Meas. Anal. Comput. Syst.*, June 2022.
DOI: [10.1145/3530901](https://doi.org/10.1145/3530901) 185, 190
- [BS06] Justin Brickell and Vitaly Shmatikov. Efficient anonymity-preserving data collection. IN: *Proceedings of the 12th ACM SIGKDD international conference on Knowledge discovery and data mining - KDD '06*, 2006. ISBN: 978-1-59593-339-3.
DOI: [10.1145/1150402.1150415](https://doi.org/10.1145/1150402.1150415) 62
- [BS16] Mark Bun and Thomas Steinke. Concentrated Differential Privacy: Simplifications, Extensions, and Lower Bounds. IN: *Theory of Cryptography*, 2016. ISBN: 978-3-662-53641-4.
DOI: [10.1007/978-3-662-53641-4_24](https://doi.org/10.1007/978-3-662-53641-4_24) 6
- [Buk19] Preston Bukaty. The California Consumer Privacy Act (CCPA): An implementation guide. 2019. ISBN: 978-1-78778-132-0.
DOI: [10.2307/j.ctvjghvnn](https://doi.org/10.2307/j.ctvjghvnn) xii, 2
- [CAB⁺15] Chen Chen, Daniele E. Asoni, David Barrera, George Danezis, and Adrain Perrig. HORNET: High-speed Onion Routing at the Network Layer. IN: *Proceedings of the 22nd ACM SIGSAC Conference on Computer and Communications Security*, 2015. ISBN: 978-1-4503-3832-5.
DOI: [10.1145/2810103.2813628](https://doi.org/10.1145/2810103.2813628) 52, 53
- [Car21] Nicholas Carlini. Poisoning the Unlabeled Dataset of Semi-Supervised

Learning. IN: *30th USENIX Security Symposium (USENIX Security 21)*, 2021. ISBN: 978-1-939133-24-3.

73

- [CB17] Henry Corrigan-Gibbs and Dan Boneh. Prio: Private, Robust, and Scalable Computation of Aggregate Statistics. IN: *14th USENIX Symposium on Networked Systems Design and Implementation (NSDI 17)*, March 2017. ISBN: 978-1-931971-37-9.

xiii

- [CB22] Edwige Cyffers and Aurélien Bellet. Privacy amplification by decentralization. IN: *Proceedings of The 25th International Conference on Artificial Intelligence and Statistics*. PMLR, 2022.

6, 29

- [CBM15] Henry Corrigan-Gibbs, Dan Boneh, and David Mazières. Riposte: An Anonymous Messaging System Handling Millions of Users. IN: *2015 IEEE Symposium on Security and Privacy*, 2015.

DOI: 10.1109/SP.2015.27

xvii, 14, 18, 52, 53, 56, 137, 140, 144, 145, 146, 147, 149, 155, 156, 158, 159, 160, 161, 167, 171, 172, 174, 175, 180, 181, 183

- [CBU24] Edwige Cyffers, Aurélien Bellet, and Jalaj Upadhyay. Differentially private decentralized learning with random walks. *International Conference on Machine Learning (ICML)*, 2024.

186, 194, 195

- [CCG24] E. Chen, Yang Cao, and Yifei Ge. A Generalized Shuffle Framework for Privacy Amplification: Strengthening Privacy Guarantees and Enhancing Utility. *Proceedings of the AAAI Conference on Artificial Intelligence*, March 2024. ISSN: 2374-3468.

DOI: 10.1609/aaai.v38i10.29005

63

- [CCL⁺21] Jinming Cui, Chaochao Chen, Lingjuan Lyu, Carl Yang, and Wang Li. Exploiting Data Sparsity in Secure Cross-Platform Social Recommendation. *Advances in Neural Information Processing Systems*, 2021.

26, 30, 31, 115, 116, 127

- [CCPT23] Konstantinos Chatzikokolakis, Giovanni Cherubin, Catuscia Palamidessi, and Carmela Troncoso. Bayes Security: A Not So Average Metric. IN: *2023 IEEE 36th Computer Security Foundations Symposium (CSF) (CSF)*, July 2023.

DOI: 10.1109/CSF57540.2023.00011

7

- [CDI05] Ronald Cramer, Ivan Damgård, and Yuval Ishai. Share conversion, pseudorandom secret-sharing and applications to secure distributed computing. IN: *Theory of Cryptography*, 2005.

- 11, 146
- [CDJ⁺17] David Chaum, Debajyoti Das, Farid Javani, Aniket Kate, Anna Krasnova, Joeri De Ruiter, and Alan T. Sherman. cMix: Mixing with Minimal Real-Time Asymmetric Cryptographic Operations. IN: *Applied Cryptography and Network Security*, 2017. ISBN: 978-3-319-61204-1. DOI: 10.1007/978-3-319-61204-1_28 52, 53
- [CDW⁺23] Jiawei Chen, Hande Dong, Xiang Wang, Fuli Feng, Meng Wang, and Xiangnan He. Bias and debias in recommender system: a survey and future directions. *ACM Trans. Inf. Syst.*, 2023. ISSN: 1046-8188. DOI: 10.1145/3564284 190
- [CEBM22] Edwige Cyffers, Mathieu Even, Aurélien Bellet, and Laurent Mas-soulié. Muffliato: peer-to-peer privacy amplification for decentralized optimization and averaging. *Advances in Neural Information Processing Systems (NeurIPS)*, 2022. 29, 186
- [CF10] Henry Corrigan-Gibbs and Bryan Ford. Dissent: accountable anonymous group messaging. IN: *Proceedings of the 17th ACM conference on Computer and communications security*, 2010. ISBN: 978-1-4503-0245-6. DOI: 10.1145/1866307.1866346 52, 53
- [CFJ⁺21] Kewei Cheng, Tao Fan, Yilun Jin, Yang Liu, Tianjian Chen, Dimitrios Papadopoulos, and Qiang Yang. SecureBoost: A Lossless Federated Learning Framework. *IEEE Intelligent Systems*, 2021. ISSN: 1941-1294. DOI: 10.1109/MIS.2021.3082561 28
- [CG10] Gennaro Cordasco and Luisa Gargano. Community detection via semi-synchronous label propagation algorithms. IN: *2010 IEEE International Workshop on: Business Applications of Social Network Analysis (BASNA)*, December 2010. DOI: 10.1109/BASNA.2010.5730298 197
- [CGGI20] Ilaria Chillotti, Nicolas Gama, Mariya Georgieva, and Malika Izabachène. Tfhe: fast fully homomorphic encryption over the torus. *Journal of Cryptology*, 2020. 203, 204, 205
- [CGKO06] Reza Curtmola, Juan Garay, Seny Kamara, and Rafail Ostrovsky. Searchable Symmetric Encryption: Improved Definitions and Efficient Constructions. IN: *Proceedings of the 13th ACM Conference on Computer and Communications Security*, 2006. ISBN: 1-59593-518-5. DOI: 10.1145/1180405.1180417 xv, 35, 105, 208
- [CGP20] Melissa Chase, Esha Ghosh, and Oxana Poburinnaya. Secret-Shared

- Shuffle. IN: *Advances in Cryptology – ASIACRYPT 2020*, 2020. ISBN: 978-3-030-64840-4.
DOI: [10.1007/978-3-030-64840-4_12](https://doi.org/10.1007/978-3-030-64840-4_12) 59
- [CGPR15] David Cash, Paul Grubbs, Jason Perry, and Thomas Ristenpart. Leakage-Abuse Attacks Against Searchable Encryption. IN: *Proceedings of the 22nd ACM SIGSAC Conference on Computer and Communications Security*, 2015. ISBN: 978-1-4503-3832-5.
DOI: [10.1145/2810103.2813700](https://doi.org/10.1145/2810103.2813700)
xv, 38, 77, 79, 80, 83, 84, 85, 86, 89, 91, 92, 94, 105, 109, 110, 111
- [CGS⁺21] Mayee Chen, Karan Goel, Nimit S. Sohoni, Fait Poms, Kayvon Fatahalian, and Christopher Re. Mandoline: Model Evaluation under Distribution Shift. IN: *Proceedings of the 38th International Conference on Machine Learning*, July 2021.
ISSN: 2640-3498 112
- [Cha81] David L Chaum. Untraceable electronic mail, return addresses, and digital pseudonyms. *Communications of the ACM*, 1981.
45, 52, 53, 54
- [Cha88] David Chaum. The dining cryptographers problem: Unconditional sender and recipient untraceability. *Journal of Cryptology*, 1988. ISSN: 1432-1378.
DOI: [10.1007/BF00206326](https://doi.org/10.1007/BF00206326) 45, 56
- [Che21] Albert Cheu. Differential Privacy in the Shuffle Model: A Survey of Separations. *arXiv:2107.11839 [cs]*, 2021.
41, 43, 63, 64, 71, 73
- [CJG21] Xiaoyu Cao, Jinyuan Jia, and Neil Zhenqiang Gong. Data Poisoning Attacks to Local Differential Privacy Protocols. IN: *30th USENIX Security Symposium (USENIX Security 21)*, 2021. ISBN: 978-1-939133-24-3.
73
- [CKKS17] Jung Hee Cheon, Andrey Kim, Miran Kim, and Yongsoo Song. Homomorphic Encryption for Arithmetic of Approximate Numbers. en. IN: *Advances in Cryptology – ASIACRYPT 2017*. Springer International Publishing, 2017. ISBN: 978-3-319-70694-8.
DOI: [10.1007/978-3-319-70694-8_15](https://doi.org/10.1007/978-3-319-70694-8_15) 203, 204, 205
- [CKR⁺20] Hao Chen, Miran Kim, Ilya Razenshteyn, Dragos Rotaru, Yongsoo Song, and Sameer Wagh. Maliciously Secure Matrix Multiplication with Applications to Private Deep Learning. IN: *Advances in Cryptology – ASIACRYPT 2020*, 2020. ISBN: 978-3-030-64840-4.
DOI: [10.1007/978-3-030-64840-4_2](https://doi.org/10.1007/978-3-030-64840-4_2) 13, 121
- [CKS22] Clement Canonne, Gautam Kamath, and Thomas Steinke. Discrete

- Gaussian for Differential Privacy. *Journal of Privacy and Confidentiality*, July 2022. ISSN: 2575-8527.
DOI: [10.29012/jpc.784](https://doi.org/10.29012/jpc.784) 8, 176
- [CKY25] Geoffroy Couteau, Naman Kumar, and Xiaxi Ye. Multiparty Homomorphic Secret Sharing and More from LPN and MQ, 2025.
URL: <https://eprint.iacr.org/2025/966> 18, 215
- [CLRZ18] G. Chen, T. Lai, M. K. Reiter, and Y. Zhang. Differentially Private Access Patterns for Searchable Symmetric Encryption. IN: *IEEE INFOCOM 2018 - IEEE Conference on Computer Communications*, 2018.
92, 93
- [CLT14] Jean-Sébastien Coron, Tancrede Lepoint, and Mehdi Tibouchi. Scale-Invariant Fully Homomorphic Encryption over the Integers. IN: *Public-Key Cryptography – PKC 2014*, 2014. ISBN: 978-3-642-54631-0.
DOI: [10.1007/978-3-642-54631-0_18](https://doi.org/10.1007/978-3-642-54631-0_18) 161
- [CMG09] Meeyoung Cha, Alan Mislove, and Krishna P. Gummadi. A measurement driven analysis of information propagation in the flickr social network. IN: *Proceedings of the 18th international conference on World wide web*, 2009. ISBN: 978-1-60558-487-4.
DOI: [10.1145/1526709.1526806](https://doi.org/10.1145/1526709.1526806) 23
- [CMZ14] Melissa Chase, Sarah Meiklejohn, and Greg Zaverucha. Algebraic MACs and keyed-verification anonymous credentials. IN: *Proceedings of the 2014 ACM SIGSAC Conference on Computer and Communications Security*, 2014.
50
- [CMZH24] Nan Cheng, Aikaterini Mitrokotsa, Feng Zhang, and Frank Hartmann. Efficient Two-Party Secure Aggregation via Incremental Distributed Point Function. IN: *2024 IEEE 9th European Symposium on Security and Privacy (EuroS&P)*, July 2024.
DOI: [10.1109/EuroSP60621.2024.00050](https://doi.org/10.1109/EuroSP60621.2024.00050) 19
- [CNM04] Aaron Clauset, M. E. J. Newman, and Cristopher Moore. Finding community structure in very large networks. *Physical Review E*, December 2004.
DOI: [10.1103/PhysRevE.70.066111](https://doi.org/10.1103/PhysRevE.70.066111) 197
- [CPH95] Clifford C. Clogg, Eva Petkova, and Adamantios Haritou. Statistical Methods for Comparing Regression Coefficients Between Models. *American Journal of Sociology*, March 1995. ISSN: 0002-9602.
DOI: [10.1086/230638](https://doi.org/10.1086/230638) 106
- [CSU⁺19] Albert Cheu, Adam Smith, Jonathan Ullman, David Zeber, and

- Maxim Zhilyaev. Distributed Differential Privacy via Shuffling. IN: *Advances in Cryptology – EUROCRYPT 2019*, 2019. ISBN: 978-3-030-17653-2.
DOI: [10.1007/978-3-030-17653-2_13](https://doi.org/10.1007/978-3-030-17653-2_13) 41, 42, 43, 45, 71
- [CSX17] Yudong Chen, Lili Su, and Jiaming Xu. Distributed Statistical Machine Learning in Adversarial Settings: Byzantine Gradient Descent. *Proceedings of the ACM on Measurement and Analysis of Computing Systems*, 2017.
DOI: [10.1145/3154503](https://doi.org/10.1145/3154503) 48
- [CW87] D. Coppersmith and S. Winograd. Matrix multiplication via arithmetic progressions. IN: *Proceedings of the nineteenth annual ACM symposium on Theory of computing*, 1987. ISBN: 978-0-89791-221-1.
DOI: [10.1145/28395.28396](https://doi.org/10.1145/28395.28396) 13
- [CWZ⁺22] Di Chai, Leye Wang, Junxue Zhang, Liu Yang, Shuwei Cai, Kai Chen, and Qiang Yang. Practical lossless federated singular vector decomposition over billion-scale data. IN: *Proceedings of the 28th ACM SIGKDD conference on knowledge discovery and data mining*, 2022.
28
- [CZW⁺21] Chaochao Chen, Jun Zhou, Li Wang, Xibin Wu, Wenjing Fang, Jin Tan, Lei Wang, Alex X. Liu, Hao Wang, and Cheng Hong. When Homomorphic Encryption Marries Secret Sharing: Secure Large-Scale Sparse Logistic Regression and Applications in Risk Control. IN: *Proceedings of the 27th ACM SIGKDD Conference on Knowledge Discovery & Data Mining*, 2021. ISBN: 978-1-4503-8332-5.
DOI: [10.1145/3447548.3467210](https://doi.org/10.1145/3447548.3467210) 26, 115, 116
- [DC25a] Marc Damie and Edwige Cyffers. Fedivertex, 2025.
DOI: [10.34740/KAGGLE/DS/6877842](https://doi.org/10.34740/KAGGLE/DS/6877842) 189
- [DC25b] Marc Damie and Edwige Cyffers. Fedivertex: a Graph Dataset based on Decentralized Social Networks for Trustworthy Machine Learning, 2025.
DOI: [10.48550/arXiv.2505.20882](https://doi.org/10.48550/arXiv.2505.20882) xvii, xix, 185, 196
- [dCP22] Leo de Castro and Anitgoni Polychroniadou. Lightweight, Maliciously Secure Verifiable Function Secret Sharing. IN: *Advances in Cryptology – EUROCRYPT 2022*, 2022. ISBN: 978-3-031-06944-4.
DOI: [10.1007/978-3-031-06944-4_6](https://doi.org/10.1007/978-3-031-06944-4_6) 17
- [DDE⁺23] Morten Dahl, Daniel Demmler, Sarah El Kazdadi, Arthur Meyre, Jean-Baptiste Orfila, Dragos Rotaru, Nigel P. Smart, Samuel Tap, and Michael Walter. Noah's Ark: Efficient Threshold-FHE Using Noise Flooding. IN: *Proceedings of the 11th Workshop on Encrypted*

- Computing & Applied Homomorphic Cryptography*, November 2023. ISBN: 979-8-4007-0255-6.
DOI: [10.1145/3605759.3625259](https://doi.org/10.1145/3605759.3625259) 167, 177
- [DDHP22] Marco Dijkslag, Marc Damie, Florian Hahn, and Andreas Peter. Passive Query-Recovery Attack Against Secure Conjunctive Keyword Search Schemes. IN: *Applied Cryptography and Network Security*, 2022. ISBN: 978-3-031-09234-3.
DOI: [10.1007/978-3-031-09234-3_7](https://doi.org/10.1007/978-3-031-09234-3_7) xix, 35
- [DDM03] G. Danezis, R. Dingledine, and N. Mathewson. Mixminion: design of a type III anonymous remailer protocol. IN: *2003 Symposium on Security and Privacy, 2003*. 2003.
DOI: [10.1109/SECPRI.2003.1199323](https://doi.org/10.1109/SECPRI.2003.1199323) 52, 53
- [DEC25] Florine W. Dekker, Zekeriya Erkin, and Mauro Conti. Topology-Based Reconstruction Prevention for Decentralised Learning. *Proceedings on Privacy Enhancing Technologies*, 2025. ISSN: 2299-0984.
DOI: [10.56553/popets-2025-0030](https://doi.org/10.56553/popets-2025-0030) 186
- [DHP02] Iain S. Duff, Michael A. Heroux, and Roldan Pozo. An overview of the sparse basic linear algebra subprograms: The new standard from the BLAS technical forum. *ACM Transactions on Mathematical Software*, 2002. ISSN: 0098-3500.
DOI: [10.1145/567806.567810](https://doi.org/10.1145/567806.567810) xiv
- [DHP21] Marc Damie, Florian Hahn, and Andreas Peter. A Highly Accurate Query-Recovery Attack against Searchable Encryption using Non-Indexed Documents. IN: *30th USENIX Security Symposium (USENIX Security 21)*, 2021. ISBN: 978-1-939133-24-3.
xvii, xviii, 77, 95, 97, 98, 102, 109, 110, 111, 112
- [DHPR25a] Marc Damie, Florian Hahn, Andreas Peter, and Jan Ramon. DDH-based schemes for multi-party Function Secret Sharing. IN: *Nordic Conference on Secure IT Systems (NordSec)*, 2025.
xvii, xviii, 140
- [DHPR25b] Marc Damie, Florian Hahn, Andreas Peter, and Jan Ramon. Eliminating exponential key growth in PRG-based Distributed Point Functions. IN: *European Symposium on Research in Computer Security Workshops*, 2025.
xvii, xviii, 133
- [DHPR25c] Marc Damie, Florian Hahn, Andreas Peter, and Jan Ramon. How to Securely Shuffle? A survey about Secure Shufflers for privacy-preserving computations, 2025.
DOI: [10.48550/arXiv.2507.01487](https://doi.org/10.48550/arXiv.2507.01487) xvii, xviii, 41, 54
- [DHPR25d] Marc Damie, Florian Hahn, Andreas Peter, and Jan Ramon. Se-

cure Sparse Matrix Multiplications and their Applications to Privacy-Preserving Machine Learning, 2025.

xvii, xviii, 115, 129

- [DHRW16] Yevgeniy Dodis, Shai Halevi, Ron D. Rothblum, and Daniel Wichs. Spooky Encryption and Its Applications. IN: *Advances in Cryptology – CRYPTO 2016*, 2016. ISBN: 978-3-662-53014-6.

DOI: 10.1007/978-3-662-53015-3_4

18

- [Dij21] Marco Dijkslag. Passive query-recovery attack against secure conjunctive keyword search schemes. M.Sc. thesis, University of Twente, 2021.

xix

- [DKA11] Daniel M. Dunlavy, Tamara G. Kolda, and Evrim Acar. Temporal link prediction using matrix and tensor factorizations. *ACM Trans. Knowl. Discov. Data*, 2011. ISSN: 1556-4681.

DOI: 10.1145/1921632.1921636

194

- [DLHP25] Marc Damie, Jean-Benoist Leger, Florian Hahn, and Andreas Peter. Revisiting the Attacker's Knowledge in Inference Attacks Against Searchable Symmetric Encryption. IN: *Applied Cryptography and Network Security*, 2025. ISBN: 978-3-031-95764-2.

DOI: 10.1007/978-3-031-95764-2_15

xvii, xviii, 95

- [DLL⁺19] Jean-Guillaume Dumas, Pascal Lafourcade, Julio Lopez Fenner, David Lucas, Jean-Baptiste Orfila, Clément Pernet, and Maxime Puys. Secure Multiparty Matrix Multiplication Based on Strassen-Winograd Algorithm. IN: *Advances in Information and Computer Security*, 2019. ISBN: 978-3-030-26834-3.

DOI: 10.1007/978-3-030-26834-3_5

13

- [DMH⁺25] Marc Damie, Federico Mazzzone, Florian Hahn, Andreas Peter, and Jan Ramon. Noisy Function Secret Sharing and its applications to Differential Privacy, 2025.

xvii, xviii, 158, 181

- [DMS04] Roger Dingledine, Nick Mathewson, and Paul Syverson. Tor: The second-generation onion router. technical report, Naval Research Lab Washington DC, 2004.

52, 53, 54, 64

- [DPP25] Marc Damie, Mihai Pop, and Merijn Posthuma. Energy Consumption of TLS, Searchable Encryption and Fully Homomorphic Encryption, 2025.

DOI: 10.48550/arXiv.2508.04583

xvii, xix, 199

- [DR14] Cynthia Dwork and Aaron Roth. The Algorithmic Foundations of Dif-

- ferential Privacy. *Foundations and Trends® in Theoretical Computer Science*, August 2014. ISSN: 1551-305X.
DOI: [10.1561/04000000042](https://doi.org/10.1561/04000000042) 5, 6, 8
- [dROvE23] Pepijn de Reus, Ana Oprescu, and Koen van Elsen. Energy cost and machine learning accuracy impact of k-anonymisation and synthetic data techniques. IN: *2023 International Conference on ICT for Sustainability (ICT4S)*, June 2023.
DOI: [10.1109/ICT4S58814.2023.00015](https://doi.org/10.1109/ICT4S58814.2023.00015) 200, 201, 202, 204, 208, 217
- [DS05] George Danezis and Andrei Serjantov. Statistical Disclosure or Intersection Attacks on Anonymity Systems. IN: *Information Hiding*, 2005. ISBN: 978-3-540-30114-1.
DOI: [10.1007/978-3-540-30114-1_21](https://doi.org/10.1007/978-3-540-30114-1_21) 50
- [DS17] Jack Doerner and Abhi Shelat. Scaling ORAM for Secure Computation. IN: *Proceedings of the 2017 ACM SIGSAC Conference on Computer and Communications Security*, 2017. ISBN: 978-1-4503-4946-8.
DOI: [10.1145/3133956.3133967](https://doi.org/10.1145/3133956.3133967) 14, 17, 37, 217
- [dSAC⁺20] Lucas Airam C. de Souza, Gabriel Antonio F. Rebello, Gustavo F. Camilo, Lucas C. B. Guimarães, and Otto Carlos M. B. Duarte. DFedForest: Decentralized Federated Forest. IN: *2020 IEEE International Conference on Blockchain (Blockchain)*, 2020.
DOI: [10.1109/Blockchain50366.2020.00019](https://doi.org/10.1109/Blockchain50366.2020.00019) 28
- [DST23] Dominique Dittert, Thomas Schneider, and Amos Treiber. Too Close for Comfort? Measuring Success of Sampled-Data Leakage Attacks Against Encrypted Search. IN: *Cloud Computing Security Workshop (CCSW '23)*, 2023.
95, 102
- [Duf77] I.S. Duff. A survey of sparse matrix research. *Proceedings of the IEEE*, 1977. ISSN: 1558-2256.
DOI: [10.1109/PROC.1977.10514](https://doi.org/10.1109/PROC.1977.10514) xiv, 215
- [DWA⁺23] F. Betül Durak, Chenkai Weng, Erik Anderson, Kim Laine, and Melissa Chase. Precio: Private Aggregate Measurement via Oblivious Shuffling, 2023.
62, 63
- [DWA⁺24] F. Betül Durak, Chenkai Weng, Erik Anderson, Kim Laine, and Melissa Chase. Precio: Private Aggregate Measurement via Oblivious Shuffling. IN: *Proceedings of the 2024 ACM SIGSAC Conference on Computer and Communications Security*, 2024.

- [Dwo06] Cynthia Dwork. Differential Privacy. IN: *Automata, Languages and Programming*, 2006. ISBN: 978-3-540-35908-1.
DOI: [10.1007/11787006_1](https://doi.org/10.1007/11787006_1) 5, 71
- [DWS⁺11] Mina Deng, Kim Wuyts, Riccardo Scandariato, Bart Preneel, and Wouter Joosen. A privacy threat analysis framework: supporting the elicitation and fulfillment of privacy requirements. *Requirements Engineering*, March 2011. ISSN: 1432-010X.
DOI: [10.1007/s00766-010-0115-7](https://doi.org/10.1007/s00766-010-0115-7) 200
- [EB22] Saba Eskandarian and Dan Boneh. Clarion: Anonymous Communication from Multiparty Shuffling Protocols. IN: *Proceedings 2022 Network and Distributed System Security Symposium*, 2022. ISBN: 978-1-891562-74-7.
DOI: [10.14722/ndss.2022.24141](https://doi.org/10.14722/ndss.2022.24141) 3, 45, 52, 53, 59
- [ECZB21] Saba Eskandarian, Henry Corrigan-Gibbs, Matei Zaharia, and Dan Boneh. Express: Lowering the cost of metadata-hiding communication with cryptographic privacy. IN: *30th USENIX Security Symposium (USENIX Security 21)*, 2021. 47
- [EFM⁺19] Ulfar Erlingsson, Vitaly Feldman, Ilya Mironov, Ananth Raghunathan, Kunal Talwar, and Abhradeep Thakurta. Amplification by Shuffling: From Local to Central Differential Privacy via Anonymity. IN: *Proceedings of the 2019 Annual ACM-SIAM Symposium on Discrete Algorithms (SODA)*, 2019.
DOI: [10.1137/1.9781611975482.151](https://doi.org/10.1137/1.9781611975482.151) 6, 41, 63
- [EFM⁺20] Ulfar Erlingsson, Vitaly Feldman, Ilya Mironov, Ananth Raghunathan, Shuang Song, Kunal Talwar, and Abhradeep Thakurta. Encode, Shuffle, Analyze Privacy Revisited: Formalizations and Empirical Evaluation. *arXiv:2001.03618 [cs]*, 2020. 62
- [EKR18] David Evans, Vladimir Kolesnikov, and Mike Rosulek. A Pragmatic Introduction to Secure Multi-Party Computation. *Foundations and Trends® in Privacy and Security*, 2018. ISSN: 2474-1558.
DOI: [10.1561/33000000019](https://doi.org/10.1561/33000000019) xiii, 2, 11, 13, 121
- [Eve23] Mathieu Even. Stochastic gradient descent under Markovian sampling schemes. IN: *Proceedings of the 40th International Conference on Machine Learning*. PMLR, 2023. 186
- [EXWB23] Maximilian Egger, Marvin Xhemrishi, Antonia Wachter-Zeh, and Rawad Bitar. Sparse and Private Distributed Matrix Multiplication with Straggler Tolerance, 2023.

- [EY09] Matthew Edman and Bülent Yener. On anonymity in an electronic society: a survey of anonymous communication systems. *ACM Computing Surveys (CSUR)*, 2009. 26
- [FBL⁺21] Marion Ficher, Françoise Berthoud, Anne-Laure Ligozat, Patrick Sigonneau, Maxime Wisslé, and Badis Tebbani. Assessing the carbon footprint of the data transmission on a backbone network. en. IN: *24th Conference on Innovation in Clouds, Internet and Networks*, March 2021. 47
- URL: <https://hal.archives-ouvertes.fr/hal-03196527> 200, 202
- [FGD⁺25] Aidan Ferguson, Perry Gibson, Lara D’Agata, Parker McLeod, Ferhat Yaman, Amitabh Das, Ian Colbert, and José Cano. Exploiting unstructured sparsity in fully homomorphic encrypted dnns. IN: *Proceedings of the 5th Workshop on Machine Learning and Systems*, 2025. ISBN: 9798400715389. 26, 30, 31
- DOI: [10.1145/3721146.3721948](https://doi.org/10.1145/3721146.3721948)
- [FGPT24] Francesca Falzon, Esha Ghosh, Kenneth G. Paterson, and Roberto Tamassia. PathGES: An Efficient and Secure Graph Encryption Scheme for Shortest Path Queries. IN: *Proceedings of the 2024 on ACM SIGSAC Conference on Computer and Communications Security*, December 2024. ISBN: 979-8-4007-0636-3. 35, 216
- DOI: [10.1145/3658644.3670305](https://doi.org/10.1145/3658644.3670305)
- [FJS07] Joan Feigenbaum, Aaron Johnson, and Paul Syverson. A Model of Onion Routing with Provable Anonymity. IN: *Financial Cryptography and Data Security*, 2007. ISBN: 978-3-540-77366-5. 45
- DOI: [10.1007/978-3-540-77366-5_9](https://doi.org/10.1007/978-3-540-77366-5_9)
- [FKO⁺17] Giulia Fanti, Peter Kairouz, Sewoong Oh, Kannan Ramchandran, and Pramod Viswanath. Hiding the Rumor Source. *IEEE Transactions on Information Theory*, 2017. ISSN: 1557-9654. 52, 53, 60
- DOI: [10.1109/TIT.2017.2696960](https://doi.org/10.1109/TIT.2017.2696960)
- [FKW⁺24] Ahmad Faiz, Sotaro Kaneda, Ruhan Wang, Rita Chukwunyere Osi, Prateek Sharma, Fan Chen, and Lei Jiang. LLMCarbon: modeling the end-to-end carbon footprint of large language models. IN: *The Twelfth International Conference on Learning Representations*, 2024. 201
- [FLSZ17] Prastudy Fauzi, Helger Lipmaa, Janno Siim, and Michał Zając. An Efficient Pairing-Based Shuffle Argument. IN: *Advances in Cryptology – ASIACRYPT 2017*, 2017. ISBN: 978-3-319-70697-9. 58
- DOI: [10.1007/978-3-319-70697-9_4](https://doi.org/10.1007/978-3-319-70697-9_4)

- [FM02] Michael J. Freedman and Robert Morris. **Tarzan: a peer-to-peer anonymizing network layer**. IN: *Proceedings of the 9th ACM conference on Computer and communications security*, 2002. ISBN: 978-1-58113-612-8.
DOI: [10.1145/586110.586137](https://doi.org/10.1145/586110.586137) [52], [53]
- [FMA⁺20] Francesca Falzon, Evangelia Anna Markatou, Akshima, David Cash, Adam Rivkin, Jesse Stern, and Roberto Tamassia. **Full Database Reconstruction in Two Dimensions**. IN: *Proceedings of the 2020 ACM SIGSAC Conference on Computer and Communications Security*, November 2020. ISBN: 978-1-4503-7089-9.
DOI: [10.1145/3372297.3417275](https://doi.org/10.1145/3372297.3417275) [77]
- [FMT23] Vitaly Feldman, Audra McMillan, and Kunal Talwar. **Stronger Privacy Amplification by Shuffling for Rényi and Approximate Differential Privacy**, October 2023. [63]
- [FMTT18] Vitaly Feldman, Ilya Mironov, Kunal Talwar, and Abhradeep Thakurta. **Privacy Amplification by Iteration**. IN: *2018 IEEE 59th Annual Symposium on Foundations of Computer Science (FOCS)*, 2018.
DOI: [10.1109/FOCS.2018.00056](https://doi.org/10.1109/FOCS.2018.00056) [6]
- [FOSZ23] Brett Falk, Rafail Ostrovsky, Matan Shtepel, and Jacob Zhang. **GigaDORAM: breaking the billion address barrier**. IN: *Proceedings of the 32nd USENIX Conference on Security Symposium*, August 2023. ISBN: 978-1-939133-37-3. [37], [217]
- [FS01] Jun Furukawa and Kazue Sako. **An Efficient Scheme for Proving a Shuffle**. IN: *Advances in Cryptology — CRYPTO 2001*, 2001.
DOI: [10.1007/3-540-44647-8_22](https://doi.org/10.1007/3-540-44647-8_22) [52], [53]
- [FSB⁺24] Jordan Frery, Andrei Stoian, Roman Bredehoft, Luis Montero, Celia Kherfallah, Benoit Chevallier-Mames, and Arthur Meyre. **Privacy-Preserving Tree-Based Inference with TFHE**. IN: *Mobile, Secure, and Programmable Networking*, 2024. ISBN: 978-3-031-52426-4.
DOI: [10.1007/978-3-031-52426-4_10](https://doi.org/10.1007/978-3-031-52426-4_10) [28]
- [GBL98] C. Lee Giles, Kurt D. Bollacker, and Steve Lawrence. **Citeseer: an automatic citation indexing system**. IN: *Proceedings of the Third ACM Conference on Digital Libraries*, 1998. ISBN: 0897919653.
DOI: [10.1145/276675.276685](https://doi.org/10.1145/276675.276685) [186]
- [GBSG06] José María Gómez Hidalgo, Guillermo Cajigas Bringas, Enrique Puertas Sáenz, and Francisco Carrero García. **Content based SMS**

- spam filtering. IN: *Proceedings of the 2006 ACM symposium on Document engineering*, 2006. ISBN: 978-1-59593-515-1.
DOI: [10.1145/1166160.1166191](https://doi.org/10.1145/1166160.1166191) 130
- [GDD⁺21] Antonious Girgis, Deepesh Data, Suhas Diggavi, Peter Kairouz, and Ananda Theertha Suresh. Shuffled Model of Differential Privacy in Federated Learning. IN: *Proceedings of The 24th International Conference on Artificial Intelligence and Statistics*, 2021. 64
- [GDPR16] Regulation (EU) 2016/679 of the European Parliament and of the Council of 27 April 2016 on the protection of natural persons with regard to the processing of personal data and on the free movement of such data, and repealing Directive 95/46/EC (General Data Protection Regulation), April 2016.
URL: <http://data.europa.eu/eli/reg/2016/679/oj/eng> xii, 2
- [Gen09] Craig Gentry. A Fully Homomorphic Encryption Scheme. PhD Thesis, Stanford University, 2009. 20
- [GGD25] Nilufer Guldali, Chakshu Gupta, and Marc Damie. IoTLS 2.0: How far has IoT industry come in securing communications with TLS?, 2025. xix
- [GGK⁺20] Badih Ghazi, Noah Golowich, Ravi Kumar, Pasin Manurangsi, Rasmus Pagh, and Ameya Velingker. Pure Differentially Private Summation from Anonymous Messages. IN: *1st Conference on Information-Theoretic Cryptography*, 2020. 45, 64
- [GGK⁺21] Badih Ghazi, Noah Golowich, Ravi Kumar, Rasmus Pagh, and Ameya Velingker. On the Power of Multiple Anonymous Messages: Frequency Estimation and Selection in the Shuffle Model of Differential Privacy. IN: *Advances in Cryptology – EUROCRYPT 2021*, 2021. ISBN: 978-3-030-77883-5.
DOI: [10.1007/978-3-030-77883-5_16](https://doi.org/10.1007/978-3-030-77883-5_16) 62, 64
- [GH13] Nethanel Gelernter and Amir Herzberg. On the limits of provable anonymity. IN: *Proceedings of the 12th ACM Workshop on privacy in the electronic society*, 2013. ISBN: 978-1-4503-2485-4.
DOI: [10.1145/2517840.2517850](https://doi.org/10.1145/2517840.2517850) 45, 70
- [GHI⁺83] Zvi Griliches, James J Heckman, Michael D Intriligator, Robert F Engle, Edward E Leamer, and Daniel McFadden. *Handbook of econometrics*. Elsevier, 1983. 100

- [GHJ⁺20] Daniel Günther, Marco Holz, Benjamin Judkewitz, Helen Möllering, Benny Pinkas, and Thomas Schneider. PEM: Privacy-preserving Epidemiological Modeling, 2020. 62, 63
- [GI14] Niv Gilboa and Yuval Ishai. Distributed Point Functions and Their Applications. IN: *Advances in Cryptology – EUROCRYPT 2014*, 2014. ISBN: 978-3-642-55220-5. DOI: [10.1007/978-3-642-55220-5_35](https://doi.org/10.1007/978-3-642-55220-5_35) 14, 16, 18, 33, 61, 155, 158, 164
- [GIK⁺24] Adrià Gascón, Yuval Ishai, Mahimna Kelkar, Baiyu Li, Yiping Ma, and Mariana Raykova. Computationally Secure Aggregation and Private Information Retrieval in the Shuffle Model. IN: *Proceedings of the 2024 on ACM SIGSAC Conference on Computer and Communications Security*, December 2024. ISBN: 979-8-4007-0636-3. DOI: [10.1145/3658644.3670391](https://doi.org/10.1145/3658644.3670391) 64, 75
- [GJ04] Philippe Golle and Ari Juels. Dining Cryptographers Revisited. IN: *Advances in Cryptology - EUROCRYPT 2004*, 2004. ISBN: 978-3-540-24676-3. DOI: [10.1007/978-3-540-24676-3_27](https://doi.org/10.1007/978-3-540-24676-3_27) 56
- [GKL⁺22] Udit Gupta, Young Geun Kim, Sylvia Lee, Jordan Tse, Hsien-Hsin S. Lee, Gu-Yeon Wei, David Brooks, and Carole-Jean Wu. Chasing Carbon: The Elusive Environmental Footprint of Computing. *IEEE Micro*, July 2022. ISSN: 1937-4143. DOI: [10.1109/MM.2022.3163226](https://doi.org/10.1109/MM.2022.3163226) 199, 201, 211
- [GKLX22] S. Dov Gordon, Jonathan Katz, Mingyu Liang, and Jiayu Xu. Spreading the Privacy Blanket: Differentially Oblivious Shuffling for Differential Privacy. IN: *International Conference on Applied Cryptography and Network Security (ACNS)*. Springer, 2022. 42, 44
- [GKMP20] Badih Ghazi, Ravi Kumar, Pasin Manurangsi, and Rasmus Pagh. Private Counting from Anonymous Messages: Near-Optimal Accuracy with Vanishing Communication Overhead. IN: *Proceedings of the 37th International Conference on Machine Learning*, 2020. 63
- [GKS⁺22] Stefanos Georgiou, Maria Kechagia, Tushar Sharma, Federica Sarro, and Ying Zou. Green AI: do deep learning frameworks have different costs? IN: *Proceedings of the 44th International Conference on Software Engineering*, July 2022. ISBN: 978-1-4503-9221-1. DOI: [10.1145/3510003.3510221](https://doi.org/10.1145/3510003.3510221) 200, 202
- [GLMP18] Paul Grubbs, Marie-Sarah Lacharité, Brice Minaud, and Kenneth G Paterson. Pump up the volume: Practical database reconstruction from

- volume leakage on range queries. IN: *Proceedings of the 2018 ACM SIGSAC Conference on Computer and Communications Security*, 2018. 77
- [GMP15] Sanjam Garg, Payman Mohassel, and Charalampos Papamanthou. TWORAM: Round-Optimal Oblivious RAM with Applications to Searchable Encryption, 2015.
URL: <https://eprint.iacr.org/2015/1010> 37
- [GMPV20] Badih Ghazi, Pasin Manurangsi, Rasmus Pagh, and Ameya Velingker. Private aggregation from fewer anonymous messages. IN: *Annual International Conference on the Theory and Applications of Cryptographic Techniques*, 2020. 62
- [GO94] Oded Goldreich and Yair Oren. Definitions and properties of zero-knowledge proof systems. *Journal of Cryptology*, 1994. ISSN: 1432-1378.
DOI: [10.1007/BF00195207](https://doi.org/10.1007/BF00195207) 21
- [GO96] Oded Goldreich and Rafail Ostrovsky. Software protection and simulation on oblivious RAMs. *Journal of the ACM*, 1996. ISSN: 0004-5411.
DOI: [10.1145/233551.233553](https://doi.org/10.1145/233551.233553) 36, 216
- [Gol09] Oded Goldreich. *Foundations of Cryptography: Volume 2, Basic Applications*. 2009. ISBN: 978-1-107-39397-4. 122
- [GPP23] Zichen Gui, Kenneth G Paterson, and Sikhar Patranabis. Rethinking Searchable Symmetric Encryption. IN: *2023 IEEE Symposium on Security and Privacy (SP)*, 2023. 98
- [GPPW24] Zichen Gui, Kenneth G. Paterson, Sikhar Patranabis, and Bogdan Warinschi. SWiSSSE: System-Wide Security for Searchable Symmetric Encryption. *Proceedings on Privacy Enhancing Technologies*, 2024. ISSN: 2299-0984. 208
- [GPT23] Zichen Gui, Kenneth G. Paterson, and Tianxin Tang. Security Analysis of MongoDB Queryable Encryption. IN: *32nd USENIX Security Symposium (USENIX Security 23)*, 2023. ISBN: 978-1-939133-37-3. 216
- [GPV19] Badih Ghazi, Rasmus Pagh, and Ameya Velingker. Scalable and Differentially Private Distributed Aggregation in the Shuffled Model, 2019. 42

- [Gri20] Sarah Griffiths. Why your internet habits are not as clean as you think. *BBC*, March 2020. 201
- [Gul25] Nilufer Guldali. Analysis of TLS Usage in IoT Devices. B.Sc. thesis, University of Twente, 2025. xix
- [GWC⁺23] Tianyao Gu, Yilei Wang, Bingnan Chen, Afonso Tinoco, Elaine Shi, and Ke Yi. Efficient Oblivious Sorting and Shuffling for Hardware Enclaves, 2023. 52, 53, 60, 74
- [GWMD23] Elia Guerra, Francesc Wilhelmi, Marco Miozzo, and Paolo Dini. The cost of training machine learning models over distributed data sources. *IEEE Open Journal of the Communications Society*, 2023. 200, 204
- [GWW25] Aarushi Goel, Mingyuan Wang, and Zhiheng Wang. Multiparty Distributed Point Functions. IN: *Advances in Cryptology – CRYPTO 2025*. Springer Nature Switzerland, 2025. ISBN: 978-3-032-01884-7. DOI: [10.1007/978-3-032-01884-7_5](https://doi.org/10.1007/978-3-032-01884-7_5) 18, 143, 145, 146, 148, 149, 158, 159, 160, 180, 181, 182, 215
- [GZZ24] Jiacheng Gao, Yuan Zhang, and Sheng Zhong. Multiparty Shuffle: Linear Online Phase is Almost for Free, 2024. 59
- [HAVK22] Thomas Humphries, Rasoul Akhavan Mahdavi, Shannon Veitch, and Florian Kerschbaum. Selective MPC: Distributed Computation of Differentially Private Key-Value Statistics. IN: *Proceedings of the 2022 ACM SIGSAC Conference on Computer and Communications Security*, 2022. ISBN: 978-1-4503-9450-5. DOI: [10.1145/3548606.3560559](https://doi.org/10.1145/3548606.3560559) 55, 64
- [HBM20] Hadrien Hendrikx, Francis Bach, and Laurent Massoulié. Dual-free stochastic decentralized optimization with variance reduction. IN: *Conference on Neural Information Processing Systems (NeurIPS)*, 2020. 186
- [HCSL24] Bjorn Ho, Huanhuan Chen, Zeshun Shi, and Kaitai Liang. Similar Data is Powerful: Enhancing Inference Attacks on SSE with Volume Leakages. IN: *Computer Security – ESORICS 2024*, 2024. ISBN: 978-3-031-70903-6. DOI: [10.1007/978-3-031-70903-6_6](https://doi.org/10.1007/978-3-031-70903-6_6) 96
- [Hen23] Hadrien Hendrikx. A principled framework for the design and anal-

- ysis of token algorithms. IN: *Proceedings of The 26th International Conference on Artificial Intelligence and Statistics*. PMLR, 2023. 186
- [HHC⁺23] Alexandra Henzinger, Matthew M. Hong, Henry Corrigan-Gibbs, Sarah Meiklejohn, and Vinod Vaikuntanathan. One Server for the Price of Two: Simple and Fast Single-Server Private Information Retrieval. IN: *32nd USENIX Security Symposium (USENIX Security 23)*, 2023. ISBN: 978-1-939133-37-3. 32
- [HHR⁺20] Peter Henderson, Jieru Hu, Joshua Romoff, Emma Brunskill, Dan Jurafsky, and Joelle Pineau. Towards the systematic reporting of the energy and carbon footprints of machine learning. *Journal of Machine Learning Research*, 2020. 200, 201, 202, 203
- [HICT14] Koki Hamada, Dai Ikarashi, Koji Chida, and Katsumi Takahashi. Oblivious Radix Sort: An Efficient Sorting Algorithm for Practical Secure Multi-party Computation, 2014. 14, 116, 122, 123, 124
- [HK16] F. Maxwell Harper and Joseph A. Konstan. The MovieLens Datasets: History and Context. *ACM Transactions on Interactive Intelligent Systems*, 2016. ISSN: 2160-6455. DOI: [10.1145/2827872](https://doi.org/10.1145/2827872) 23, 130
- [HM20] Thomas Haines and Johannes Müller. SoK: Techniques for Verifiable Mix Nets. IN: *2020 IEEE 33rd Computer Security Foundations Symposium (CSF)*, 2020. DOI: [10.1109/CSF49147.2020.00012](https://doi.org/10.1109/CSF49147.2020.00012) 57
- [HMPS14] Susan Hohenberger, Steven Myers, Rafael Pass, and Abhi Shelat. ANONIZE: A Large-Scale Anonymous Survey System. IN: *2014 IEEE Symposium on Security and Privacy*, 2014. DOI: [10.1109/SP.2014.31](https://doi.org/10.1109/SP.2014.31) 52, 53, 62
- [HN07] Lingxin Hao and Daniel Q Naiman. Quantile regression. 2007. 108
- [Hoo12] de Hoogh S.J.A. Design of large scale applications of secure multiparty computation : secure linear programming. PhD Thesis, Technische Universiteit Eindhoven, 2012. DOI: [10.6100/IR735328](https://doi.org/10.6100/IR735328) 13, 14, 121, 122, 124
- [HPMG20] Kevin Hsieh, Amar Phanishayee, Onur Mutlu, and Phillip Gibbons. The Non-IID Data Quagmire of Decentralized Machine Learning. IN: *International Conference on Machine Learning (ICML)*, November 2020.

- [HPP20] Chloé Héban, Duong Hieu Phan, and David Pointcheval. **Linearly-Homomorphic Signatures and Scalable Mix-Nets**. IN: *Public-Key Cryptography – PKC 2020*, 2020. ISBN: 978-3-030-45388-6.
DOI: [10.1007/978-3-030-45388-6_21](https://doi.org/10.1007/978-3-030-45388-6_21) 185
- [HR69] Josef Hadar and William R Russell. **Rules for ordering uncertain prospects**. *The American economic review*, 1969. 58
- [HRO17] Michael Z. Hauschild, Ralph K. Rosenbaum, and Stig Irving Olsen. **Life Cycle Assessment: Theory and Practice**. September 2017. ISBN: 978-3-319-56475-3. 164
- [HRS⁺24] Ehsan Hallaji, Roozbeh Razavi-Far, Mehrdad Saif, Boyu Wang, and Qiang Yang. **Decentralized Federated Learning: A Survey on Security and Privacy**. *IEEE Transactions on Big Data*, April 2024. ISSN: 2332-7790.
DOI: [10.1109/TBDATA.2024.3362191](https://doi.org/10.1109/TBDATA.2024.3362191) 201
- [HXN⁺16] Ruili Huang, Menghang Xia, Dac-Trung Nguyen, Tongan Zhao, Srilatha Sakamuru, Jinghua Zhao, Sampada A. Shahane, Anna Rossoshek, and Anton Simeonov. **Tox21Challenge to Build Predictive Models of Nuclear Receptor and Stress Response Pathways as Mediated by Exposure to Environmental Chemicals and Drugs**. *Frontiers in Environmental Science*, 2016. ISSN: 2296-665X. 185
- [HYS24] Shiori Hironaka, Mitsuo Yoshida, and Kazuyuki Shudo. **Comparing user activity on X and Mastodon**. IN: *2024 IEEE International Conference on Big Data (BigData)*, 2024.
DOI: [10.1109/bigdata62323.2024.10825212](https://doi.org/10.1109/bigdata62323.2024.10825212) 23
- [IKK12] Mohammad Saiful Islam, Mehmet Kuzu, and Murat Kantarcioglu. **Access pattern disclosure on searchable encryption: Ramification, attack and mitigation**. IN: *Network and Distributed System Security Symposium (NDSS)*, 2012. xv, 77, 79, 80, 83, 86, 89, 90, 91, 92, 112
- [IKOS06] Yuval Ishai, Eyal Kushilevitz, Rafail Ostrovsky, and Amit Sahai. **Cryptography from Anonymity**. IN: *2006 47th Annual IEEE Symposium on Foundations of Computer Science (FOCS'06)*, 2006.
DOI: [10.1109/FOCS.2006.25](https://doi.org/10.1109/FOCS.2006.25) 41, 45, 50, 62, 64, 68, 70, 73
- [Int15] Intel. **Software guard extensions programming reference**. *Intel Corporation*, 2015. 21, 59

- [JGB⁺24] Neha Jawalkar, Kanav Gupta, Arkaprava Basu, Nishanth Chandran, Divya Gupta, and Rahul Sharma. Orca: FSS-based Secure Training and Inference with GPUs. In: *2024 IEEE Symposium on Security and Privacy (SP)*, 2024. 28, 156
- [JOL⁺23] Mathilde Jay, Vladimir Ostapenco, Laurent Lefevre, Denis Trystram, Anne-Cécile Orgerie, and Benjamin Fichel. An experimental comparison of software-based power meters: focus on CPU and GPU. In: *2023 IEEE/ACM 23rd International Symposium on Cluster, Cloud and Internet Computing (CCGrid)*, May 2023. DOI: [10.1109/CCGrid57682.2023.00020](https://doi.org/10.1109/CCGrid57682.2023.00020) 202, 204
- [JRJ10] Björn Johansson, Mabien Rabi, and Mikael Johansson. A randomized incremental subgradient method for distributed optimization in networked systems. *SIAM Journal on Optimization*, 2010. DOI: [10.1137/08073038X](https://doi.org/10.1137/08073038X) 186
- [JST⁺24] Ujun Jeong, Paras Sheth, Anique Tahir, Faisal Alatawi, H Russell Bernard, and Huan Liu. Exploring platform migration patterns between twitter and mastodon: a user behavior study. In: *Proceedings of the International AAAI Conference on Web and Social Media*, 2024. 189
- [JYS18] James Jordon, Jinsung Yoon, and Mihaela van der Schaar. PATE-GAN: Generating Synthetic Data with Differential Privacy Guarantees. In: *International conference on learning representations*, December 2018. 208
- [KB78] Roger Koenker and Gilbert Bassett. Regression quantiles. *Econometrica*, 1978. 108
- [KBS⁺19] Christiane Kuhn, Martin Beck, Stefan Schiffner, Eduard Jorswieck, and Thorsten Strufe. On Privacy Notions in Anonymous Communication. *Proceedings on Privacy Enhancing Technologies*, 2019. ISSN: 2299-0984. DOI: [10.2478/popets-2019-0022](https://doi.org/10.2478/popets-2019-0022) 45
- [KCD⁺21] Hanna Kavalionak, Emanuele Carlini, Patrizio Dazzi, Luca Ferrucci, Matteo Mordacchini, and Massimo Coppola. Impact of Network Topology on the Convergence of Decentralized Federated Learning Systems. In: *2021 IEEE Symposium on Computers and Communications (ISCC)*, September 2021. DOI: [10.1109/ISCC53001.2021.9631460](https://doi.org/10.1109/ISCC53001.2021.9631460) 185

- [KCDF17] Albert Kwon, Henry Corrigan-Gibbs, Srinivas Devadas, and Bryan Ford. Atom: Horizontally Scaling Strong Anonymity. IN: *Proceedings of the 26th Symposium on Operating Systems Principles*, 2017. ISBN: 978-1-4503-5085-3.
DOI: [10.1145/3132747.3132755](https://doi.org/10.1145/3132747.3132755) 52, 53, 58, 61
- [KCY23] Fumiyuki Kato, Yang Cao, and Masatoshi Yoshikawa. Olive: oblivious federated learning on trusted execution environment against the risk of sparsification. *Proc. VLDB Endow.*, June 2023. ISSN: 2150-8097.
DOI: [10.14778/3603581.3603583](https://doi.org/10.14778/3603581.3603583) 24
- [Kel20] Marcel Keller. MP-SPDZ: A Versatile Framework for Multi-Party Computation. IN: *Proceedings of the 2020 ACM SIGSAC Conference on Computer and Communications Security*, 2020. ISBN: 978-1-4503-7089-9.
DOI: [10.1145/3372297.3417872](https://doi.org/10.1145/3372297.3417872) 115
- [KH01] Roger Koenker and Kevin F. Hallock. Quantile Regression. *Journal of Economic Perspectives*, 2001. ISSN: 0895-3309.
DOI: [10.1257/jep.15.4.143](https://doi.org/10.1257/jep.15.4.143) 108
- [KHN⁺18] Kashif Nizam Khan, Mikael Hirki, Tapio Niemi, Jukka K. Nurminen, and Zhonghong Ou. RAPL in Action: Experiences in Using RAPL for Power Measurements. *ACM Trans. Model. Perform. Eval. Comput. Syst.*, March 2018. ISSN: 2376-3639.
DOI: [10.1145/3177754](https://doi.org/10.1145/3177754) 202, 203
- [KKM⁺22] Seny Kamara, Abdelkarim Kati, Tarik Moataz, Thomas Schneider, Amos Treiber, and Michael Yonli. SoK: Cryptanalysis of Encrypted Search with LEAKER - A framework for LEakage AttacK Evaluation on Real-world data. IN: *IEEE European Symposium on Security and Privacy (EuroS&P'22)*, June 2022. 96
- [KKN20] Hemant H Kumar, VR Karthik, and Mydhili K Nair. Federated k-means clustering: a novel edge ai based approach for privacy preservation. IN: *2020 IEEE International Conference on Cloud Computing in Emerging Markets (CCEM)*. IEEE, 2020. 28
- [KKNO16] Georgios Kellaris, George Kollios, Kobbi Nissim, and Adam O'neill. Generic attacks on secure outsourced databases. IN: *Proceedings of the 2016 ACM SIGSAC Conference on Computer and Communications Security*, 2016. 77
- [KLB⁺20] Anastasia Koloskova, Nicolas Loizou, Sadra Boreiri, Martin Jaggi, and Sebastian Stich. A unified theory of decentralized SGD with

- changing topology and local updates. IN: *International Conference on Machine Learning (ICML)*, 2020. 186
- [KLDF16] Albert Kwon, David Lazar, Srinivas Devadas, and Bryan Ford. Riffle: An Efficient Communication System With Strong Anonymity. *Proceedings on Privacy Enhancing Technologies*, 2016. ISSN: 2299-0984.
DOI: [10.1515/popets-2016-0008](https://doi.org/10.1515/popets-2016-0008) 52, 53, 58
- [KLS21] Peter Kairouz, Ziyu Liu, and Thomas Steinke. The Distributed Discrete Gaussian Mechanism for Federated Learning with Secure Aggregation. IN: *Proceedings of the 38th International Conference on Machine Learning*, July 2021. 8, 9, 176, 182
- [KM14] Daniel Kifer and Ashwin Machanavajjhala. Pufferfish: a framework for mathematical privacy definitions. *ACM Transactions on Database Systems (TODS)*, 2014. 7
- [KM23] Seny Kamara and Tarik Moataz. Bayesian Leakage Analysis: A Framework for Analyzing Leakage in Encrypted Search, 2023. 96, 216
- [KMA⁺21] Peter Kairouz et al. Advances and Open Problems in Federated Learning. *Foundations and Trends® in Machine Learning*, 2021. ISSN: 1935-8237.
DOI: [10.1561/22000000083](https://doi.org/10.1561/22000000083) 28, 42, 44, 185, 186
- [KMPP22] Evgenios M. Kornaropoulos, Nathaniel Moyer, Charalampos Papamanthou, and Alexandros Psomas. Leakage Inversion: Towards Quantifying Privacy in Searchable Encryption. IN: *Proceedings of the 2022 ACM SIGSAC Conference on Computer and Communications Security*, November 2022. ISBN: 978-1-4503-9450-5.
DOI: [10.1145/3548606.3560593](https://doi.org/10.1145/3548606.3560593) 96
- [KMRR16] Jakub Konečný, H. Brendan McMahan, Daniel Ramage, and Peter Richtárik. Federated Optimization: Distributed Machine Learning for On-Device Intelligence, 2016. 28, 41, 69
- [KMY⁺17] Jakub Konečný, H. Brendan McMahan, Felix X. Yu, Peter Richtárik, Ananda Theertha Suresh, and Dave Bacon. Federated Learning: Strategies for Improving Communication Efficiency. *arXiv:1610.05492 [cs]*, 2017. 41
- [KMY⁺24] Ronny Ko, Rasoul Akhavan Mahdavi, Byoungwoo Yoon, Makoto

- Onizuka, and Florian Kerschbaum. *SilentWood: Private Inference Over Gradient-Boosting Decision Forests*, November 2024.
DOI: [10.48550/arXiv.2411.15494](https://doi.org/10.48550/arXiv.2411.15494) 28, 206
- [KÖ23] Jakub Klemsa and Melek Önen. *PARMESAN: Parallel ARithMEticS over ENcrypted data*, 2023. 203
- [KOR19] Daniel Kales, Olamide Omolola, and Sebastian Ramacher. *Revisiting User Privacy for Certificate Transparency*. IN: *2019 IEEE European Symposium on Security and Privacy (EuroS&P)*, June 2019.
DOI: [10.1109/EuroSP.2019.00039](https://doi.org/10.1109/EuroSP.2019.00039) 33
- [KP25] Toomas Krips and Pille Pullonen-Raudvere. *Multi-Party Distributed Point Functions with Polylogarithmic Key Size from Invariants of Matrices*, 2025. 18, 140, 146, 147, 215
- [KPM24] Chandan Kumar, Sikhar Patranabis, and Debdeep Mukhopadhyay. *Compact Key Function Secret Sharing with Non-linear Decoder*. *IACR Communications in Cryptology*, July 2024. ISSN: 3006-5496.
DOI: [10.62056/a3c3c3w9p](https://doi.org/10.62056/a3c3c3w9p) 14, 17, 18, 138, 146, 147, 150, 154, 155, 156, 159, 161, 163, 180, 181, 215
- [KPPS21] Nishat Koti, Mahak Pancholi, Arpita Patra, and Ajith Suresh. *SWIFT: super-fast and robust Privacy-Preserving machine learning*. IN: *30th USENIX Security Symposium (USENIX Security 21)*, 2021. 13, 14, 115, 121, 124
- [KPT19] Evgenios M. Kornaropoulos, Charalampos Papamanthou, and Roberto Tamassia. *Data Recovery on Encrypted Databases with k-Nearest Neighbor Query Leakage*. IN: *2019 IEEE Symposium on Security and Privacy (SP)*, May 2019.
DOI: [10.1109/SP.2019.00015](https://doi.org/10.1109/SP.2019.00015) 77
- [KPT20] Evgenios M. Kornaropoulos, Charalampos Papamanthou, and Roberto Tamassia. *The State of the Uniform: Attacks on Encrypted Databases Beyond the Uniform Query Distribution*. IN: *2020 IEEE Symposium on Security and Privacy (SP)*, May 2020. ISBN: 978-1-72813-497-0.
DOI: [10.1109/SP40000.2020.00029](https://doi.org/10.1109/SP40000.2020.00029) 77
- [KRS22] Mahimna Kelkar, Mariana Raykova, and Karn Seth. *Secure Poisson Regression*. IN: *31st USENIX Security Symposium (USENIX Security 22)*, 2022. 124
- [KY04] Bryan Klimt and Yiming Yang. *Introducing the enron corpus*. IN: *CEAS*, 2004.

- [LBT⁺23] Batiste Le Bars, Aurélien Bellet, Marc Tommasi, Erick Lavoie, and Anne-Marie Kermarrec. Refined convergence and topology learning for decentralized sgd with heterogeneous data. IN: *Conference on Uncertainty in Artificial Intelligence (AISTATS)*, 2023. 97
- [LCC⁺21] Ruixuan Liu, Yang Cao, Hong Chen, Ruoyang Guo, and Masatoshi Yoshikawa. Flame: Differentially private federated learning in the shuffle model. IN: *Proceedings of the AAAI Conference on Artificial Intelligence*, 2021. 186
- [LCL24] Tjard Jan Langhout, Huanhuan Chen, and Kaitai Liang. File-Injection Attacks on Searchable Encryption, Based on Binomial Structures. IN: *Computer Security – ESORICS 2024*, 2024. ISBN: 978-3-031-70896-1. DOI: [10.1007/978-3-031-70896-1_21](https://doi.org/10.1007/978-3-031-70896-1_21) 64
- [LDJ19] Tianbo Lu, Zeyu Du, and Z. Jane Wang. A Survey on Measuring Anonymity in Anonymous Communication Systems. *IEEE Access*, 2019. ISSN: 2169-3536. DOI: [10.1109/ACCESS.2019.2919322](https://doi.org/10.1109/ACCESS.2019.2919322) 77
- [Lei18] Charles E. Leiserson. Performance Engineering of Software Systems. Lecture 1: introduction and Matrix Multiplication Leiserson, 2018. 45
- [LHK10] Jure Leskovec, Daniel Huttenlocher, and Jon Kleinberg. Predicting positive and negative links in online social networks. IN: *Proceedings of the 19th international conference on World wide web*, 2010. ISBN: 978-1-60558-799-8. DOI: [10.1145/1772690.1772756](https://doi.org/10.1145/1772690.1772756) 120
- [Lin17] Yehuda Lindell. How to simulate it—a tutorial on the simulation proof technique. *Tutorials on the Foundations of Cryptography: Dedicated to Oded Goldreich*, 2017. 193
- [LK07] David Liben-Nowell and Jon Kleinberg. The link-prediction problem for social networks. *Journal of the American Society for Information Science and Technology*, 2007. ISSN: 1532-2890. DOI: [10.1002/asi.20591](https://doi.org/10.1002/asi.20591) 4
- [LK14] Jure Leskovec and Andrej Krevl. SNAP Datasets: Stanford large network dataset collection, 2014. 186, 193

- [LK23] Donghang Lu and Aniket Kate. RPM: Robust Anonymity at Scale. *Proceedings on Privacy Enhancing Technologies*, 2023. ISSN: 2299-0984.
DOI: [10.56553/popets-2023-0057](https://doi.org/10.56553/popets-2023-0057) 3, 14, 48, 52, 53, 59
- [LLL⁺20] Yang Liu, Yingting Liu, Zhijie Liu, Yuxuan Liang, Chuishi Meng, Junbo Zhang, and Yu Zheng. Federated Forest. *IEEE Transactions on Big Data*, 2020. ISSN: 2332-7790.
DOI: [10.1109/TBDATA.2020.2992755](https://doi.org/10.1109/TBDATA.2020.2992755) 28
- [LLV07] Ninghui Li, Tiancheng Li, and Suresh Venkatasubramanian. T-Closeness: Privacy Beyond k-Anonymity and l-Diversity. IN: *2007 IEEE 23rd International Conference on Data Engineering*, 2007.
DOI: [10.1109/ICDE.2007.367856](https://doi.org/10.1109/ICDE.2007.367856) 7
- [LLW17] Xiang Li, Charles X. Ling, and Huaimin Wang. The Convergence Behavior of Naive Bayes on Large Sparse Datasets. *ACM Transactions on Knowledge Discovery from Data*, 2017. ISSN: 1556-4681.
DOI: [10.1145/2948068](https://doi.org/10.1145/2948068) xiv, 23, 125
- [LM12] Jure Leskovec and Julian McAuley. Learning to Discover Social Circles in Ego Networks. IN: *Conference on Neural Information Processing Systems (NeurIPS)*, 2012.
193
- [LMP18] Marie-Sarah Lacharité, Brice Minaud, and Kenneth G Paterson. Improved reconstruction attacks on encrypted data using range query leakage. IN: *2018 IEEE Symposium on Security and Privacy (SP)*, 2018.
77
- [LO13] Steve Lu and Rafail Ostrovsky. Distributed Oblivious RAM for Secure Two-Party Computation. IN: *Theory of Cryptography*, 2013. ISBN: 978-3-642-36594-2.
DOI: [10.1007/978-3-642-36594-2_22](https://doi.org/10.1007/978-3-642-36594-2_22) 37
- [LRK⁺25] Samantha Lai, Yoel Roth, Kate Klonick, Mallory Knodel, Evan Prodromou, and Aaron Rodericks. New Paradigms in Trust and Safety: Navigating Defederation on Decentralized Social Media Platforms. technical report, Carnegie Endowment for International Peace, April 2025.
195
- [LSD23] Simon Langowski, Sacha Servan-Schreiber, and Srinivas Devadas. Trellis: Robust and Scalable Metadata-private Anonymous Broadcast. IN: *Network and Distributed System Security Symposium (NDSS)*, 2023.
52, 53, 72, 74

- [LSTS20] Tian Li, Anit Kumar Sahu, Ameet Talwalkar, and Virginia Smith. Federated learning: challenges, methods, and future directions. *IEEE Signal Processing Magazine*, 2020. ISSN: 1558-0792.
DOI: [10.1109/msp.2020.2975749](https://doi.org/10.1109/msp.2020.2975749) 186
- [LTS⁺18] Christine Lemmer-Webber, Jessica Tallon, Erin Shepherd, Amy Guy, and Evan Prodromou. Activitypub w3c recommendation, 2018. 186, 187
- [LVL23] Alexandra Sasha Luccioni, Sylvain Viguier, and Anne-Laure Ligozat. Estimating the Carbon Footprint of BLOOM, a 176B Parameter Language Model. *Journal of Machine Learning Research*, 2023. ISSN: 1533-7928. 201
- [LW23] Kijung Lee and Mian Wang. Uses and Gratifications of Alternative Social Media: Why do people use Mastodon?, March 2023.
DOI: [10.48550/arXiv.2303.01285](https://doi.org/10.48550/arXiv.2303.01285) 212
- [LWCQ23] Yiwei Li, Shuai Wang, Chong-Yung Chi, and Tony QS Quek. Differentially private federated clustering over non-iid data. *IEEE Internet of Things Journal*, 2023. 28
- [LWY22] Qiyao Luo, Yilei Wang, and Ke Yi. Frequency estimation in the shuffle model with almost a single message. IN: *Proceedings of the 2022 ACM SIGSAC Conference on Computer and Communications Security*, 2022. ISBN: 9781450394505.
DOI: [10.1145/3548606.3560608](https://doi.org/10.1145/3548606.3560608) 41, 64
- [LWZ11] Sven Laur, Jan Willemson, and Bingsheng Zhang. Round-Efficient Oblivious Database Manipulation. IN: *Information Security*, 2011. ISBN: 978-3-642-24861-0.
DOI: [10.1007/978-3-642-24861-0_18](https://doi.org/10.1007/978-3-642-24861-0_18) 14, 116, 123, 124
- [LYK⁺19] Donghang Lu, Thomas Yurek, Samarth Kulshreshtha, Rahul Govind, Aniket Kate, and Andrew Miller. HoneyBadgerMPC and AsynchroMix: Practical Asynchronous MPC and its Application to Anonymous Communication. IN: *Proceedings of the 2019 ACM SIGSAC Conference on Computer and Communications Security*, 2019. ISBN: 978-1-4503-6747-9.
DOI: [10.1145/3319535.3354238](https://doi.org/10.1145/3319535.3354238) 52, 53, 59
- [Lyo14] David Lyon. Surveillance, Snowden, and Big Data: Capacities, consequences, critique. *Big Data & Society*, July 2014. ISSN: 2053-9517.
DOI: [10.1177/2053951714541861](https://doi.org/10.1177/2053951714541861) xii
- [LZLR23] Tianpei Lu, Bingsheng Zhang, Lichun Li, and Kui Ren. Aegis: A

Lightning Fast Privacy-preserving Machine Learning Platform against Malicious Adversaries, 2023.

13, 14, 115, 121, 124

- [LZWT14] Chang Liu, Liehuang Zhu, Mingzhong Wang, and Yu-An Tan. Search pattern leakage in searchable encryption: Attacks and new construction. *Information Sciences*, 2014.

77

- [LZZ⁺17] Xiangru Lian, Ce Zhang, Huan Zhang, Cho-Jui Hsieh, Wei Zhang, and Ji Liu. Can decentralized algorithms outperform centralized algorithms? a case study for decentralized parallel stochastic gradient descent. IN: *Conference on Neural Information Processing Systems (NeurIPS)*, 2017. ISBN: 9781510860964.

186

- [Mau99] Ueli Maurer. Information-Theoretic Cryptography. IN: *Advances in Cryptology — CRYPTO’ 99*, 1999. ISBN: 978-3-540-48405-9.

DOI: 10.1007/3-540-48405-1_4

3, 72

- [MCB24] Abdellah El Mrini, Edwige Cyffers, and Aurélien Bellet. Privacy attacks in decentralized learning. *International Conference on Machine Learning (ICML)*, 2024.

185, 186

- [MD05] Steven J. Murdoch and George Danezis. Low-cost traffic analysis of Tor. IN: *2005 IEEE Symposium on Security and Privacy (S P’05)*, 2005.

DOI: 10.1109/SP.2005.12

4, 50

- [MEHP25] Federico Mazzone, Maarten Everts, Florian Hahn, and Andreas Peter. Efficient Ranking, Order Statistics, and Sorting under CKKS. IN: *34th USENIX Security Symposium (USENIX Security 25)*, 2025. ISBN: 978-1-939133-52-6.

20, 203

- [MFET23] Evangelia Anna Markatou, Francesca Falzon, Zachary Espiritu, and Roberto Tamassia. Attacks on Encrypted Response-Hiding Range Search Schemes in Multiple Dimensions. *Proceedings on Privacy Enhancing Technologies*, 2023. ISSN: 2299-0984.

35

- [MG23] Johannes Mono and Tim Güneysu. Implementing and Optimizing Matrix Triples with Homomorphic Encryption. IN: *Proceedings of the 2023 ACM Asia Conference on Computer and Communications Security*, 2023. ISBN: 9798400700989.

DOI: 10.1145/3579856.3590344

13, 121

- [Mil19] Tim Miller. Explanation in artificial intelligence: Insights from the social sciences. *Artificial Intelligence*, February 2019. ISSN: 0004-3702.
DOI: [10.1016/j.artint.2018.07.007](https://doi.org/10.1016/j.artint.2018.07.007) 104
- [MJÖC23] Mohamad Mansouri, Wafa Ben Jaballah, Melek Önen, and Mauro Conti. SoK: Secure Aggregation based on cryptographic schemes for Federated Learning. *Proceedings on Privacy Enhancing Technologies*, 2023.
24, 28, 42
- [MMR⁺17] Brendan McMahan, Eider Moore, Daniel Ramage, Seth Hampson, and Blaise Aguera y Arcas. Communication-Efficient Learning of Deep Networks from Decentralized Data. IN: *Conference on Uncertainty in Artificial Intelligence (AISTATS)*, 2017.
186
- [MNRS00] Andrew Kachites McCallum, Kamal Nigam, Jason Rennie, and Kristie Seymore. Automating the construction of internet portals with machine learning. *Information Retrieval*, 2000. ISSN: 1386-4564.
DOI: [10.1023/a:1009953814988](https://doi.org/10.1023/a:1009953814988) 186
- [Moo08] Adam D. Moore. *Defining Privacy*, 2008.
xii, 2
- [MPG22] Soumen Majhi, Matjaž Perc, and Dibakar Ghosh. Dynamics on higher-order networks: a review. *Journal of The Royal Society Interface*, 2022. ISSN: 1742-5662.
DOI: [10.1098/rsif.2022.0043](https://doi.org/10.1098/rsif.2022.0043) 194
- [MPRV09] Ilya Mironov, Omkant Pandey, Omer Reingold, and Salil Vadhan. Computational Differential Privacy. IN: *Advances in Cryptology - CRYPTO 2009*, 2009. ISBN: 978-3-642-03356-8.
DOI: [10.1007/978-3-642-03356-8_8](https://doi.org/10.1007/978-3-642-03356-8_8) 71, 72, 74
- [MR04] Jelena Mirkovic and Peter Reiher. A taxonomy of DDoS attack and DDoS defense mechanisms. *ACM SIGCOMM Computer Communication Review*, 2004. ISSN: 0146-4833.
DOI: [10.1145/997150.997156](https://doi.org/10.1145/997150.997156) 49
- [MR16] Aryan Mokhtari and Alejandro Ribeiro. Dsa: decentralized double stochastic averaging gradient algorithm. *Journal of Machine Learning Research*, 2016.
186
- [MR18] Payman Mohassel and Peter Rindal. ABY³: A Mixed Protocol Framework for Machine Learning. IN: *Proceedings of the 2018 ACM SIGSAC Conference on Computer and Communications Security*, 2018. ISBN: 978-1-4503-5693-0.
DOI: [10.1145/3243734.3243760](https://doi.org/10.1145/3243734.3243760) 11, 13, 14, 115, 121, 124

- [MR20] Aymeric Mansoux and Roel Roscam Abbing. Seven theses on the fediverse and the becoming of FLOSS. IN: Kristoffer Gansing and Inga Luchs, editors, *The Eternal Network*. Institute of Network Cultures, Amsterdam, and transmediale e.V., Berlin, 2020. ISBN: 978-94-92302-46-5. 212
- [MST24] Dimitris Mouris, Pratik Sarkar, and Nektarios Georgios Tsoutsos. PLASMA: Private, Lightweight Aggregated Statistics against Malicious Adversaries. *Proceedings on Privacy Enhancing Technologies*, 2024. ISSN: 2299-0984.
URL: <https://petsymposium.org/popets/2024/popets-2024-0064.php> 157
- [MSW11] Pedro Miranda, Matti Siekkinen, and Heikki Waris. TLS and energy consumption on a mobile device: A measurement study. IN: *2011 IEEE Symposium on Computers and Communications (ISCC)*, June 2011. DOI: [10.1109/ISCC.2011.5983970](https://doi.org/10.1109/ISCC.2011.5983970) 200, 202, 210
- [MTT17] Antonis Matakos, Evimaria Terzi, and Panayiotis Tsaparas. Measuring and moderating opinion polarization in social networks. *Data Mining and Knowledge Discovery*, 2017. ISSN: 1573-756X. DOI: [10.1007/s10618-017-0527-9](https://doi.org/10.1007/s10618-017-0527-9) 190
- [MW22] Samir Jordan Menon and David J. Wu. SPIRAL: Fast, High-Rate Single-Server PIR via FHE Composition. IN: *2022 IEEE Symposium on Security and Privacy (SP)*, 2022. DOI: [10.1109/SP46214.2022.9833700](https://doi.org/10.1109/SP46214.2022.9833700) 32, 33
- [MYH⁺20] Xianghui Mao, Kun Yuan, Yubin Hu, Yuantao Gu, Ali H. Sayed, and Wotao Yin. Walkman: A Communication-Efficient Random-Walk Algorithm for Decentralized Optimization. *IEEE Transactions on Signal Processing*, 2020. 186
- [MZ17] Payman Mohassel and Yupeng Zhang. SecureML: A System for Scalable Privacy-Preserving Machine Learning. IN: *2017 IEEE Symposium on Security and Privacy (SP)*, May 2017. DOI: [10.1109/SP.2017.12](https://doi.org/10.1109/SP.2017.12) xiii, 13, 27, 115, 121
- [NDCP24] Nicholas Ngai, Ioannis Demertzis, Javad Ghareh Chamani, and Dimitrios Papadopoulos. Distributed & scalable oblivious sorting and shuffling. IN: *2024 IEEE Symposium on Security and Privacy (SP)*. IEEE, 2024. 47, 52, 53, 60, 74
- [NDM⁺21] Rakshit Naidu, Harshita Diddee, Ajinkya Mulay, Aleti Vardhan,

- Krithika Ramesh, and Ahmed Zamzam. Towards Quantifying the Carbon Emissions of Differentially Private Machine Learning, 2021. 200, 202
- [Nef01] C. Andrew Neff. A verifiable secret shuffle and its application to e-voting. IN: *Proceedings of the 8th ACM conference on Computer and Communications Security*, 2001. 52, 53, 64
- [NFL⁺14] David Naylor, Alessandro Finamore, Ilias Leontiadis, Yan Grunenberg, Marco Mellia, Maurizio Munafò, Konstantina Papagiannaki, and Peter Steenkiste. The Cost of the "S" in HTTPS. IN: *Proceedings of the 10th ACM International on Conference on emerging Networking Experiments and Technologies*, December 2014. ISBN: 978-1-4503-3279-8. DOI: [10.1145/2674005.2674991](https://doi.org/10.1145/2674005.2674991) 200, 202, 210
- [NSD22] Zachary Newman, Sacha Servan-Schreiber, and Srinivas Devadas. Spectrum: High-bandwidth Anonymous Broadcast. IN: *19th USENIX Symposium on Networked Systems Design and Implementation (NSDI 22)*, April 2022. ISBN: 978-1-939133-27-4. 14, 17, 52, 53, 56, 57, 156
- [NSH19] Milad Nasr, Reza Shokri, and Amir Houmansadr. Comprehensive Privacy Analysis of Deep Learning: Passive and Active White-box Inference Attacks against Centralized and Federated Learning. IN: *2019 IEEE Symposium on Security and Privacy (SP)*, May 2019. 28
- [NW18] Kobbi Nissim and Alexandra Wood. Is privacy privacy? *Philosophical Transactions of the Royal Society A: Mathematical, Physical and Engineering Sciences*, 2018. DOI: [10.1098/rsta.2017.0358](https://doi.org/10.1098/rsta.2017.0358) xii, 2
- [NWI⁺15] Kartik Nayak, Xiao Shaun Wang, Stratis Ioannidis, Udi Weinsberg, Nina Taft, and Elaine Shi. GraphSC: Parallel Secure Computation Made Easy. IN: *2015 IEEE Symposium on Security and Privacy*, 2015. DOI: [10.1109/SP.2015.30](https://doi.org/10.1109/SP.2015.30) 26
- [NXTC20] Giovanni Neglia, Chuan Xu, Don Towsley, and Gianmarco Calbi. Decentralized gradient methods: does topology matter? IN: *Conference on Uncertainty in Artificial Intelligence (AISTATS)*, 2020. 186
- [Ogi24] Tabitha Ogilvie. Differential Privacy for Free? Harnessing the Noise in Approximate Homomorphic Encryption. IN: *Topics in Cryptology – CT-RSA 2024*, 2024. ISBN: 978-3-031-58868-6.

- DOI: [10.1007/978-3-031-58868-6_12](https://doi.org/10.1007/978-3-031-58868-6_12) 181
- [OK21] Simon Oya and Florian Kerschbaum. Hiding the Access Pattern is Not Enough: Exploiting Search Pattern Leakage in Searchable Encryption. IN: *30th USENIX Security Symposium (USENIX Security 21)*, 2021. xv, 77, 91, 95
- [OK22] Simon Oya and Florian Kerschbaum. IHOP: Improved Statistical Query Recovery against Searchable Symmetric Encryption through Quadratic Optimization. IN: *31st USENIX Security Symposium (USENIX Security 22)*, 2022. ISBN: 978-1-939133-31-1. 95, 96, 97, 109, 110, 111
- [Par73] Richard B. Parker. A Definition of Privacy. *Rutgers Law Review*, 1973. 2
- [PH10] Andreas Pfitzmann and Marit Hansen. A terminology for talking about privacy by data minimization: Anonymity, Unlinkability, Undetectability, Unobservability, Pseudonymity, and Identity Management, 2010. 45
- [PHE⁺17] Ania M Piotrowska, Jamie Hayes, Tariq Elahi, Sebastian Meiser, and George Danezis. The loopix anonymity system. IN: *26th USENIX Security Symposium (USENIX Security 17)*, 2017. 52, 53
- [Pop25] Mihai Pop. Measuring the energy consumption and carbon footprint of encrypted databases using CodeCarbon. B.Sc. thesis, University of Twente, 2025. xix
- [Pos25] Merijn Posthuma. The energy consumption and carbon footprint of HTTPS. B.Sc. thesis, University of Twente, 2025. xix
- [PS20] Arpita Patra and Ajith Suresh. BLAZE: Blazing Fast Privacy-Preserving Machine Learning. IN: *Proceedings 2020 Network and Distributed System Security Symposium*, 2020. ISBN: 978-1-891562-61-7. DOI: [10.14722/ndss.2020.24202](https://doi.org/10.14722/ndss.2020.24202) 13, 14, 115
- [PSSY21] Arpita Patra, Thomas Schneider, Ajith Suresh, and Hossein Yalame. ABY2. 0: Improved Mixed-Protocol Secure Two-Party Computation. IN: *30th USENIX Security Symposium (USENIX Security 21)*, 2021. 11
- [PSY23] Sarvar Patel, Joon Young Seo, and Kevin Yeo. Don't be Dense: Effi-

- cient Keyword PIR for Sparse Databases. IN: *32nd USENIX Security Symposium (USENIX Security 23)*, 2023. ISBN: 978-1-939133-37-3. 32, 34
- [PW16] David Pouliot and Charles V Wright. The shadow nemesis: Inference attacks on efficiently deployable, efficiently searchable encryption. IN: *Proceedings of the 2016 ACM SIGSAC conference on computer and communications security*, 2016. xv, 86, 87
- [RAS21] Benedek Rozemberczki, Carl Allen, and Rik Sarkar. Multi-scale attributed node embedding. *Journal of Complex Networks*, 2021. ISSN: 2051-1329. DOI: [10.1093/comnet/cnab014](https://doi.org/10.1093/comnet/cnab014) 193
- [RCF⁺20] Emanuele Rossi, Ben Chamberlain, Fabrizio Frasca, Davide Eynard, Federico Monti, and Michael Bronstein. Temporal Graph Networks for Deep Learning on Dynamic Graphs, October 2020. DOI: [10.48550/arXiv.2006.10637](https://doi.org/10.48550/arXiv.2006.10637) 194
- [Reg09] Oded Regev. On lattices, learning with errors, random linear codes, and cryptography. *Journal of the ACM*, 2009. ISSN: 0004-5411. DOI: [10.1145/1568318.1568324](https://doi.org/10.1145/1568318.1568324) 171
- [RGJvdM22] Amrita Roy Chowdhury, Chuan Guo, Somesh Jha, and Laurens van der Maaten. Eiffel: ensuring integrity for federated learning. IN: *Proceedings of the 2022 ACM SIGSAC Conference on Computer and Communications Security*, 2022. 73
- [RL08] Jason Rennie and Ken Lang. The 20 Newsgroups data set, 2008. 23
- [RMK14] Tim Ruffing, Pedro Moreno-Sanchez, and Aniket Kate. CoinShuffle: Practical Decentralized Coin Mixing for Bitcoin. IN: *Computer Security - ESORICS 2014*, 2014. DOI: [10.1007/978-3-319-11212-1_20](https://doi.org/10.1007/978-3-319-11212-1_20) 64
- [Rog15] Phillip Rogaway. The moral character of cryptographic work. *Cryptography ePrint Archive*, 2015. iv, xiii
- [RTPB22] Théo Ryffel, Pierre Tholoniati, David Pointcheval, and Francis Bach. AriaNN: Low-Interaction Privacy-Preserving Deep Learning via Function Secret Sharing. *Proceedings on Privacy Enhancing Technologies*, 2022. 14, 156, 161
- [SAB15] Mohamed Sabt, Mohammed Achemlal, and Abdelmadjid Bouab-

- dallah. Trusted Execution Environment: What It is, and What It is Not. IN: *2015 IEEE Trustcom/BigDataSE/ISPA*, August 2015.
DOI: [10.1109/Trustcom.2015.357](https://doi.org/10.1109/Trustcom.2015.357) 21, 211
- [SBFK20] Johannes Sedlmeir, Hans Ulrich Buhl, Gilbert Fridgen, and Robert Keller. The Energy Consumption of Blockchain Technology: Beyond Myth. *Business & Information Systems Engineering*, December 2020. ISSN: 1867-0202.
DOI: [10.1007/s12599-020-00656-x](https://doi.org/10.1007/s12599-020-00656-x) 200
- [SBR22] César Sabater, Aurélien Bellet, and Jan Ramon. An accurate, scalable and verifiable protocol for federated differentially private averaging. *Machine Learning*, November 2022. ISSN: 1573-0565.
DOI: [10.1007/s10994-022-06267-9](https://doi.org/10.1007/s10994-022-06267-9) (visited on 10/07/2025) 49
- [SBYP23] Sacha Servan-Schreiber, Simon Beyzerov, Eli Yablon, and Hyojae Park. Private Access Control for Function Secret Sharing. IN: *2023 IEEE Symposium on Security and Privacy (SP)*, 2023.
DOI: [10.1109/SP46215.2023.10179295](https://doi.org/10.1109/SP46215.2023.10179295) 17
- [Sch18] Berry Schoenmakers. MPyC—Python package for secure multiparty computation. IN: *Workshop on the Theory and Practice of MPC*, 2018.
118, 124
- [SCJ18] Sebastian U Stich, Jean-Baptiste Cordonnier, and Martin Jaggi. Sparsified SGD with Memory. IN: *Advances in Neural Information Processing Systems*, 2018.
28, 64
- [SDS02] Andrei Serjantov, Roger Dingledine, and Paul Syverson. From a trickle to a flood: active attacks on several mix types. IN: *International Workshop on Information Hiding*. Springer, 2002.
50
- [SEV⁺15] Yixin Sun, Anne Edmundson, Laurent Vanbever, Oscar Li, Jennifer Rexford, Mung Chiang, and Prateek Mittal. RAPTOR: Routing attacks on privacy in tor. IN: *24th USENIX Security Symposium (USENIX Security 15)*, 2015.
42, 50
- [SG23] Sajin Sasy and Ian Goldberg. SoK: Metadata-Protecting Communication Systems, 2023.
47, 64
- [SGJ⁺24] Victor Schmidt, Kamal Goyal, Aditya Joshi, Boris Feld, Liam Conell, Nikolas Laskaris, Doug Blank, Jonathan Wilson, Sorelle Friedler, and Sasha Luccioni. CodeCarbon: Estimate and Track Car-

- bon Emissions from Machine Learning Computing, version v2.4.1, May 2024.
DOI: [10.5281/zenodo.11171501](https://doi.org/10.5281/zenodo.11171501) [201, 202, 204]
- [SGRP19] Philipp Schoppmann, Adrià Gascón, Mariana Raykova, and Benny Pinkas. Make Some ROOM for the Zeros: Data Sparsity in Secure Distributed Machine Learning. IN: *Proceedings of the 2019 ACM SIGSAC Conference on Computer and Communications Security*, 2019. ISBN: 978-1-4503-6747-9.
DOI: [10.1145/3319535.3339816](https://doi.org/10.1145/3319535.3339816) [24, 26, 30, 31, 39, 115, 116, 127]
- [Sha79] Adi Shamir. How to share a secret. *Communications of the ACM*, 1979. ISSN: 0001-0782.
DOI: [10.1145/359168.359176](https://doi.org/10.1145/359168.359176) [10]
- [SHKR22] Virat Shejwalkar, Amir Houmansadr, Peter Kairouz, and Daniel Ramage. Back to the drawing board: a critical evaluation of poisoning attacks on production federated learning. IN: *2022 IEEE Symposium on Security and Privacy (SP)*, 2022.
DOI: [10.1109/SP46214.2022.9833647](https://doi.org/10.1109/SP46214.2022.9833647) [73]
- [SJG22] Sajin Sasy, Aaron Johnson, and Ian Goldberg. Fast Fully Oblivious Compaction and Shuffling. IN: *Proceedings of the 2022 ACM SIGSAC Conference on Computer and Communications Security*, 2022.
[51, 52, 53, 60, 73]
- [SKAP06] Jonathan Schler, Moshe Koppel, Shlomo Argamon, and James W Pennebaker. Effects of age and gender on blogging. IN: *AAAI spring symposium: Computational approaches to analyzing weblogs*, 2006.
[97]
- [SM25] Shuang Sun and Eleftheria Makri. SoK: Multiparty Computation in the Preprocessing Model, 2025.
[126]
- [SRKB22] Stefano Savazzi, Vittorio Rampa, Sanaz Kianoush, and Mehdi Bennis. An Energy and Carbon Footprint Analysis of Distributed and Federated Learning. *IEEE Transactions on Green Communications and Networking*, 2022. ISSN: 2473-2400.
DOI: [10.1109/TGCN.2022.3186439](https://doi.org/10.1109/TGCN.2022.3186439) [200, 202]
- [SS07] Moshe Shaked and J. George Shanthikumar. Univariate Stochastic Orders. IN: *Stochastic Orders*, 2007. ISBN: 978-0-387-34675-5.
DOI: [10.1007/978-0-387-34675-5_1](https://doi.org/10.1007/978-0-387-34675-5_1) [164]
- [SSA⁺18] Fatemeh Shirazi, Milivoj Simeonovski, Muhammad Rizwan Asghar, Michael Backes, and Claudia Diaz. A survey on routing in anonymous communication protocols. *ACM Computing Surveys (CSUR)*, 2018.
[47]

- [SSAG23] Thomas Struett, Aram Sinnreich, Patricia Aufderheide, and Robert Gehl. Can this platform survive? governance challenges for the Fediverse. *SSRN Electronic Journal*, 2023. ISSN: 1556-5068.
DOI: [10.2139/ssrn.4598303](https://doi.org/10.2139/ssrn.4598303) 186
- [SSG16] Jessica Su, Aneesh Sharma, and Sharad Goel. The effect of recommendations on network structure. IN: *Proceedings of the 25th International Conference on World Wide Web*, 2016. ISBN: 9781450341431.
DOI: [10.1145/2872427.2883040](https://doi.org/10.1145/2872427.2883040) 186, 194
- [SSTS21] Roei Schuster, Congzheng Song, Eran Tromer, and Vitaly Shmatikov. You Autocomplete Me: Poisoning Vulnerabilities in Neural Code Completion. IN: *30th USENIX Security Symposium (USENIX Security 21)*, 2021. ISBN: 978-1-939133-24-3.
73
- [SSV⁺22] Timothy Stevens, Christian Skalka, Christelle Vincent, John Ring, Samuel Clark, and Joseph Near. Efficient Differentially Private Secure Aggregation for Federated Learning via Hardness of Learning with Errors. IN: *31st USENIX Security Symposium (USENIX Security 22)*, 2022.
181
- [STSP23] Mohsen Shirali, Tobias Tefke, Ralf C. Staudemeyer, and Henrich C. Pöhls. A survey on anonymous communication systems with a focus on dining cryptographers networks. *IEEE Access*, 2023.
DOI: [10.1109/ACCESS.2023.3242870](https://doi.org/10.1109/ACCESS.2023.3242870) 47, 56
- [SU21] Adam Smith and Jonathan Ullman. Privacy in Statistics and Machine Learning: Approximate Differential Privacy. 2021.
5, 171
- [SvHA⁺19] Alex Sangers, Maran van Heesch, Thomas Attema, Thijs Veugen, Mark Wiggerman, Jan Veldsink, Oscar Bloemen, and Daniël Worm. Secure Multiparty PageRank Algorithm for Collaborative Fraud Detection. IN: *Financial Cryptography and Data Security*, 2019. ISBN: 978-3-030-32101-7.
DOI: [10.1007/978-3-030-32101-7_35](https://doi.org/10.1007/978-3-030-32101-7_35) xiii
- [SW06] Vitaly Shmatikov and Ming-Hsiu Wang. Timing Analysis in Low-Latency Mix Networks: Attacks and Defenses. IN: *Computer Security – ESORICS 2006*, 2006. ISBN: 978-3-540-44605-7.
DOI: [10.1007/11863908_2](https://doi.org/10.1007/11863908_2) 42
- [SW21] Elaine Shi and Ke Wu. Non-Interactive Anonymous Router. IN: *Advances in Cryptology – EUROCRYPT 2021*, 2021. ISBN: 978-3-030-77883-5.
DOI: [10.1007/978-3-030-77883-5_17](https://doi.org/10.1007/978-3-030-77883-5_17) 74

- [Swa94] Judith A. Swanson. The Public and the Private in Aristotle's Political Philosophy. 1994. ISBN: 978-1-5017-4083-1.
DOI: [10.7298/kehv-8s26](https://doi.org/10.7298/kehv-8s26) xii, 2
- [Swe02] Latanya Sweeney. K-anonymity: a model for protecting privacy. *International Journal of Uncertainty, Fuzziness and Knowledge-Based Systems*, 2002. ISSN: 0218-4885.
DOI: [10.1142/S0218488502001648](https://doi.org/10.1142/S0218488502001648) 7
- [SYB⁺24] Xiangfu Song, Dong Yin, Jianli Bai, Changyu Dong, and Ee-Chien Chang. Secret-Shared Shuffle with Malicious Security. IN: *Proceedings 2024 Network and Distributed System Security Symposium*, 2024. ISBN: 978-1-891562-93-8.
DOI: [10.14722/ndss.2024.24021](https://doi.org/10.14722/ndss.2024.24021) 59
- [SZS⁺23] Arnaud Grivet Sébert, Martin Zuber, Oana Stan, Renaud Sirdey, and Cédric Gouy-Pailler. When approximate design for fast homomorphic computation provides differential privacy guarantees, April 2023. 181
- [Tal22] Kunal Talwar. Differential Secrecy for Distributed Data and Applications to Robust Differentially Secure Vector Summation. IN: *3rd Symposium on Foundations of Responsible Computing (FORC 2022)*, 2022.
DOI: [10.4230/LIPIcs.FORC.2022.7](https://doi.org/10.4230/LIPIcs.FORC.2022.7) 166, 167
- [TKK⁺09] Riitta Toivonen, Lauri Kovanen, Mikko Kivelä, Jukka-Pekka Onnela, Jari Saramäki, and Kimmo Kaski. A comparative study of social network models: Network evolution models and nodal attribute models. *Social Networks*, October 2009. ISSN: 0378-8733.
DOI: [10.1016/j.socnet.2009.06.004](https://doi.org/10.1016/j.socnet.2009.06.004) 186
- [TL10] Lei Tang and Huan Liu. Graph Mining Applications to Social Network Analysis. IN: *Managing and Mining Graph Data*, 2010. ISBN: 978-1-4419-6045-0.
DOI: [10.1007/978-1-4419-6045-0_16](https://doi.org/10.1007/978-1-4419-6045-0_16) 186
- [Tof07] Tomas Toft. Primitives and applications for multi-party computation. PhD thesis, University of Aarhus, 2007. 123
- [Tsy04] Alexey Tsymbal. The problem of concept drift: definitions and related work. *Technical Report, Computer Science Department, Trinity College Dublin*, 2004. 112
- [vDar24] Bram van Dartel. The effect of data imbalances on membership

inference attacks in federated learning. M.Sc. thesis, University of Twente, 2024.

xix

- [vDDH25] Bram van Dartel, Marc Damie, and Florian Hahn. Evaluating Membership Inference Attacks in heterogeneous-data setups. IN: *Security in Machine Learning Applications (co-located with ACNS)*, 2025.

xix, 28

- [vdHLZZ15] Jelle van den Hooff, David Lazar, Matei Zaharia, and Nickolai Zeldovich. Vuvuzela: scalable private messaging resistant to traffic analysis. IN: *Proceedings of the 25th Symposium on Operating Systems Principles*, 2015. ISBN: 978-1-4503-3834-9.

DOI: 10.1145/2815400.2815417

47

- [vDTT⁺24] Sarah van Drumpt, Tjerk Timan, Soukaina Talie, Thijs Veugen, and Linda van de Burgwal. Digital transitions in healthcare: the need for transdisciplinary research to overcome barriers of privacy enhancing technologies uptake. *Health and Technology*, July 2024. ISSN: 2190-7196.

DOI: 10.1007/s12553-024-00850-x

xiii

- [Vél21] Carissa Véliz. Privacy is Power: Why and How You Should Take Back Control of Your Data. Transworld, 2021. ISBN: 978-0-552-17771-9.

xii

- [VHK⁺21] Thijs Vogels, Lie He, Anastasiia Koloskova, Sai Praneeth Karimireddy, Tao Lin, Sebastian U Stich, and Martin Jaggi. RelaySum for Decentralized Deep Learning on Heterogeneous Data. IN: *Advances in Neural Information Processing Systems*, 2021.

185

- [VSH22] Adithya Vadapalli, Kyle Storrier, and Ryan Henry. Sabre: Sender-Anonymous Messaging with Fast Audits. IN: *2022 IEEE Symposium on Security and Privacy (SP)*, 2022.

DOI: 10.1109/SP46214.2022.9833601

52, 56

- [VYS⁺23] Viet Vo, Xingliang Yuan, Shi-Feng Sun, Joseph K. Liu, Surya Nepal, and Cong Wang. ShieldDB: An Encrypted Document Database With Padding Countermeasures. *IEEE Transactions on Knowledge and Data Engineering*, April 2023. ISSN: 1558-2191.

DOI: 10.1109/TKDE.2021.3126607

105

- [WB90] Samuel Warren and Louis Brandeis. The right to privacy. *Harvard law review*, 1890.

xii, 2

- [WCFJ12] David Isaac Wolinsky, Henry Corrigan-Gibbs, Bryan Ford, and Aaron Johnson. Dissent in numbers: Making strong anonymity scale.

IN: *10th USENIX Symposium on Operating Systems Design and Implementation (OSDI 12)*, 2012.

56

- [WCP⁺17] Wenhao Wang, Guoxing Chen, Xiaorui Pan, Yinqian Zhang, Xiaofeng Wang, Vincent Bindschaedler, Haixu Tang, and Carl A. Gunter. Leaky Cauldron on the Dark Land: Understanding Memory Side-Channel Hazards in SGX. IN: *Proceedings of the 2017 ACM SIGSAC Conference on Computer and Communications Security*, 2017. ISBN: 978-1-4503-4946-8.

DOI: 10.1145/3133956.3134038

73

- [WDS⁺25] Shaowei Wang, Changyu Dong, Xiangfu Song, Jin Li, Zhili Zhou, Di Wang, and Han Wu. Beyond Statistical Estimation: Differentially Private Individual Computation via Shuffling. IN: *34th USENIX Security Symposium (USENIX Security 25)*, 2025.

42, 62, 64, 75

- [WGC19] Sameer Wagh, Divya Gupta, and Nishanth Chandran. SecureNN: 3-Party Secure Computation for Neural Network Training. *Proc. Priv. Enhancing Technol.*, 2019.

27, 115

- [WSF13] David Isaac Wolinsky, Ewa Syta, and Bryan Ford. Hang with your buddies to resist intersection attacks. IN: *Proceedings of the 2013 ACM SIGSAC conference on Computer & communications security - CCS '13*, 2013. ISBN: 978-1-4503-2477-9.

DOI: 10.1145/2508859.2516740

50

- [WSW⁺24] Yunling Wang, Shi-Feng Sun, Jianfeng Wang, Xiaofeng Chen, Joseph K. Liu, and Dawu Gu. Practical Non-interactive Encrypted Conjunctive Search with Leakage Suppression. IN: *Proceedings of the 2024 on ACM SIGSAC Conference on Computer and Communications Security*, December 2024. ISBN: 979-8-4007-0636-3.

DOI: 10.1145/3658644.3670355

216

- [WTB⁺21] Sameer Wagh, Shruti Tople, Fabrice Benhamouda, Eyal Kushilevitz, Prateek Mittal, and Tal Rabin. Falcon: Honest-Majority Maliciously Secure Framework for Private Deep Learning. *Proceedings on Privacy Enhancing Technologies*, 2021. ISSN: 2299-0984.

13, 14, 27, 115, 121, 124

- [WXC⁺25] Longxiang Wang, Lei Xu, Yufei Chen, Ying Zou, and Cong Wang. ALERT: Machine Learning-Enhanced Risk Estimation for Databases Supporting Encrypted Queries. IN: *34th USENIX Security Symposium (USENIX Security 25)*, 2025. ISBN: 978-1-939133-52-6.

216

- [Wyl19] Christopher Wylie. *Mindf*ck: Inside Cambridge Analytica's Plot to Break the World*. Profile Books, October 2019. ISBN: 9781782836773. xii
- [XBW22] Marvin Xhemrishi, Rawad Bitar, and Antonia Wachter-Zeh. Distributed Matrix-Vector Multiplication with Sparsity and Privacy Guarantees. IN: *2022 IEEE International Symposium on Information Theory (ISIT)*, 2022. DOI: [10.1109/ISIT50566.2022.9834805](https://doi.org/10.1109/ISIT50566.2022.9834805) 26
- [XDZ⁺21] Lei Xu, Huayi Duan, Anxin Zhou, Xingliang Yuan, and Cong Wang. Interpreting and mitigating leakage-abuse attacks in searchable symmetric encryption. *IEEE Transactions on Information Forensics and Security*, 2021. 105
- [XYW⁺19] Lei Xu, Xingliang Yuan, Cong Wang, Qian Wang, and Chungen Xu. Hardening database padding for searchable encryption. IN: *IEEE INFOCOM 2019-IEEE Conference on Computer Communications*, 2019. 38, 92, 93, 105, 109, 111
- [XZH24] Shuangqing Xu, Yifeng Zheng, and Zhongyun Hua. Camel : Communication-Efficient and Maliciously Secure Federated Learning in the Shuffle Model of Differential Privacy. IN: *Proceedings of the 2024 on ACM SIGSAC Conference on Computer and Communications Security*, December 2024. ISBN: 979-8-4007-0636-3. DOI: [10.1145/3658644.3690200](https://doi.org/10.1145/3658644.3690200) 64
- [YCF⁺25] Jiping Yu, Kun Chen, Xiaoyu Fan, Yunyi Chen, Xiaowei Zhu, and Wenguang Chen. Lodia: Towards Optimal Sparse Matrix-Vector Multiplication for Batched Fully Homomorphic Encryption. IN: *Proceedings of the 2025 ACM SIGSAC Conference on Computer and Communications Security*, 2025. xv, 26, 30, 31, 39
- [YDL⁺23] Qiantao Yang, Xuehui Du, Aodi Liu, Na Wang, Wenjuan Wang, and Xiangyu Wu. AdaSTopk: Adaptive federated shuffle model based on differential privacy. *Information Sciences*, September 2023. ISSN: 0020-0255. DOI: [10.1016/j.ins.2023.119186](https://doi.org/10.1016/j.ins.2023.119186) 71
- [Zac76] Wayne Zachary. An information flow model for conflict and fission in small groups. *Journal of anthropological research*, 1976. DOI: [10.1086/jar.33.4.3629752](https://doi.org/10.1086/jar.33.4.3629752) 186
- [ZAS24] Jasmin Zalonis, Frederik Armknecht, and Linda Scheu-Hachtel.

Differentially Private Functional Encryption. *Proceedings on Privacy Enhancing Technologies*, 2024. ISSN: 2299-0984.

164

- [ZGR18] Matteo Zignani, Sabrina Gaito, and Gian Paolo Rossi. Follow the “Mastodon”: Structure and Evolution of a Decentralized Online Social Network. *Proceedings of the International AAAI Conference on Web and Social Media*, 2018. ISSN: 2334-0770.

DOI: 10.1609/icwsm.v12i1.14988

186, 187, 194

- [ZJF⁺22] Lingchen Zhao, Jianlin Jiang, Bo Feng, Qian Wang, Chao Shen, and Qi Li. SEAR: Secure and Efficient Aggregation for Byzantine-Robust Federated Learning. *IEEE Transactions on Dependable and Secure Computing*, 2022. ISSN: 1941-0018.

DOI: 10.1109/TDSC.2021.3093711

73

- [ZKP16] Yupeng Zhang, Jonathan Katz, and Charalampos Papamanthou. All your queries are belong to us: The power of file-injection attacks on searchable encryption. IN: *25th USENIX Security Symposium (USENIX Security 16)*, 2016.

77

- [ZLH⁺22] En Zhang, Huimin Li, Yuchen Huang, Shuangxi Hong, Le Zhao, and Congmin Ji. Practical multi-party private collaborative k-means clustering. *Neurocomputing*, 2022. ISSN: 0925-2312.

DOI: 10.1016/j.neucom.2021.09.050

28

- [ZMKL05] Cai-Nicolas Ziegler, Sean M. McNee, Joseph A. Konstan, and Georg Lausen. Improving recommendation lists through topic diversification. IN: *Proceedings of the 14th international conference on World Wide Web*, 2005. ISBN: 978-1-59593-046-0.

DOI: 10.1145/1060745.1060754

23, 129, 130

- [ŽPG16] Indrė Žliobaitė, Mykola Pechenizkiy, and João Gama. An Overview of Concept Drift Applications. IN: *Big Data Analysis: New Algorithms for a New Society*, 2016. ISBN: 978-3-319-26989-4.

DOI: 10.1007/978-3-319-26989-4_4

112

- [ZS22] Mingxun Zhou and Elaine Shi. The Power of the Differentially Oblivious Shuffle in Distributed Privacy Mechanisms, 2022.

74

- [ZSCL24] Manning Zhang, Zeshun Shi, Huanhuan Chen, and Kaitai Liang. Inject Less, Recover More: Unlocking the Potential of Document Recovery in Injection Attacks Against SSE. IN: *2024 IEEE 37th Computer Security Foundations Symposium (CSF)*, July 2024. ISBN: 979-8-3503-6203-9.

DOI: 10.1109/CSF61375.2024.00029

77

- [Zub19] Shoshana Zuboff. Surveillance Capitalism and the Challenge of Collective Action. *New Labor Forum*, 2019. ISSN: 1095-7960.
URL: <https://www.jstor.org/stable/26675591> xii
- [ZWC⁺21] Yuhui Zhang, Zhiwei Wang, Jiangfeng Cao, Rui Hou, and Dan Meng. ShuffleFL: gradient-preserving federated learning using trusted execution environment. IN: *Proceedings of the 18th ACM International Conference on Computing Frontiers*, 2021. ISBN: 978-1-4503-8404-9.
DOI: [10.1145/3457388.3458665](https://doi.org/10.1145/3457388.3458665) 73
- [ZWX⁺23] Xianglong Zhang, Wei Wang, Peng Xu, Laurence T. Yang, and Kaitai Liang. High Recovery with Fewer Injections: Practical Binary Volumetric Injection Attacks against Dynamic Searchable Encryption. IN: *32nd USENIX Security Symposium (USENIX Security 23)*, February 2023. 77
- [ZYP24] Guy Zyskind, Avishay Yanai, and Alex "Sandy" Pentland. High-Throughput Three-Party DPFs with Applications to ORAM and Digital Currencies. IN: *Proceedings of the 2024 ACM SIGSAC Conference on Computer and Communications Security*, 2024. 14, 17, 149