N°d'ordre NNT : 2024ISAL0024

THESE de DOCTORAT DE L'INSA LYON, membre de l'UNIVERSITE DE LYON

délivré en partenariat international avec
Université de Passau

Ecole Doctorale N° 512
InfoMaths

Spécialité / discipline de doctorat :
Informatique

Soutenue publiquement le 12/04/2024, par :
Paul Lachat

Detecting Inference Attack Involving Sensor Data

Devant le jury composé de :

Cuppens, Frédéric, Professeur, Polytechnique Montreal	Rapporteur
Felfernig, Alexander, Professeur, Graz University of Technology	Rapporteur
Döller, Mario, Professeur, Kufstain University of Applied Science	Examinateur
Sassi, Salma, Maître de conférence, Université de Jendouba	Examinatrice
Granitzer, Michael, Professeur, Université de Passau	Examinateur

Brunie, Lionel, Professeur, INSA Lyon	Directeur de thèse
Kosch, Harald, Professeur, Université de Passau	Co-directeur de thèse
Bennani, Nadia, Maître de conférences	Co-superviseuse de thèse

Cette thèse est accessible à l'adresse : https://theses.insa-lyon.fr/publication/2024ISAL0024/these.pdf
© [P. Lachat], [2024], INSA Lyon, tous droits réservés

Département FEDORA – INSA Lyon - Ecoles Doctorales

SIGLE	ECOLE DOCTORALE	NOM ET COORDONNEES DU RESPONSABLE
ED 206 CHIMIE	**CHIMIE DE LYON** https://www.edchimie-lyon.fr Sec. : Renée EL MELHEM Bât. Blaise PASCAL, 3e étage secretariat@edchimie-lyon.fr	**M. Stéphane DANIELE** C2P2-CPE LYON-UMR 5265 Bâtiment F308, BP 2077 43 Boulevard du 11 novembre 1918 69616 Villeurbanne directeur@edchimie-lyon.fr
ED 341 E2M2	**ÉVOLUTION, ÉCOSYSTÈME, MICROBIOLOGIE, MODÉLISATION** http://e2m2.universite-lyon.fr Sec. : Bénédicte LANZA Bât. Atrium, UCB Lyon 1 Tél : 04.72.44.83.62 secretariat.e2m2@univ-lyon1.fr	**Mme Sandrine CHARLES** Université Claude Bernard Lyon 1 UFR Biosciences Bâtiment Mendel 43, boulevard du 11 Novembre 1918 69622 Villeurbanne CEDEX e2m2.codir@listes.univ-lyon1.fr
ED 205 EDISS	**INTERDISCIPLINAIRE SCIENCES-SANTÉ** http://ediss.universite-lyon.fr Sec. : Bénédicte LANZA Bât. Atrium, UCB Lyon 1 Tél : 04.72.44.83.62 secretariat.ediss@univ-lyon1.fr	**Mme Sylvie RICARD-BLUM** Laboratoire ICBMS - UMR 5246 CNRS - Université Lyon 1 Bâtiment Raulin - 2ème étage Nord 43 Boulevard du 11 novembre 1918 69622 Villeurbanne Cedex Tél : +33(0)4 72 44 82 32 sylvie.ricard-blum@univ-lyon1.fr
ED 34 EDML	**MATÉRIAUX DE LYON** http://ed34.universite-lyon.fr Sec. : Yann DE ORDENANA Tél : 04.72.18.62.44 yann.de-ordenana@ec-lyon.fr	**M. Stéphane BENAYOUN** Ecole Centrale de Lyon Laboratoire LTDS 36 avenue Guy de Collongue 69134 Ecully CEDEX Tél : 04.72.18.64.37 stephane.benayoun@ec-lyon.fr
ED 160 EEA	**ÉLECTRONIQUE, ÉLECTROTECHNIQUE, AUTOMATIQUE** https://edeea.universite-lyon.fr Sec. : Philomène TRECOURT Bâtiment Direction INSA Lyon Tél : 04.72.43.71.70 secretariat.edeea@insa-lyon.fr	**M. Philippe DELACHARTRE** INSA LYON Laboratoire CREATIS Bâtiment Blaise Pascal, 7 avenue Jean Capelle 69621 Villeurbanne CEDEX Tél : 04.72.43.88.63 philippe.delachartre@insa-lyon.fr
ED 512 INFOMATHS	**INFORMATIQUE ET MATHÉMATIQUES** http://edinfomaths.universite-lyon.fr Sec. : Renée EL MELHEM Bât. Blaise PASCAL, 3e étage Tél : 04.72.43.80.46 infomaths@univ-lyon1.fr	**M. Hamamache KHEDDOUCI** Université Claude Bernard Lyon 1 Bât. Nautibus 43, Boulevard du 11 novembre 1918 69 622 Villeurbanne Cedex France Tél : 04.72.44.83.69 direction.infomaths@listes.univ-lyon1.fr
ED 162 MEGA	**MÉCANIQUE, ÉNERGÉTIQUE, GÉNIE CIVIL, ACOUSTIQUE** http://edmega.universite-lyon.fr Sec. : Philomène TRECOURT Tél : 04.72.43.71.70 Bâtiment Direction INSA Lyon mega@insa-lyon.fr	**M. Etienne PARIZET** INSA Lyon Laboratoire LVA Bâtiment St. Exupéry 25 bis av. Jean Capelle 69621 Villeurbanne CEDEX etienne.parizet@insa-lyon.fr
ED 483 ScSo	**ScSo**[1] https://edsciencessociales.universite-lyon.fr Sec. : Mélina FAVETON Tél : 04.78.69.77.79 melina.faveton@univ-lyon2.fr	**M. Bruno MILLY** (INSA : J.Y. TOUSSAINT) Univ. Lyon 2 Campus Berges du Rhône 18, quai Claude Bernard 69365 LYON CEDEX 07 Bureau BEL 319 bruno.milly@univ-lyon2.fr

Cette thèse est accessible à l'adresse : https://theses.insa-lyon.fr/publication/2024ISAL0024/these.pdf
© [P. Lachat], [2024], INSA Lyon, tous droits réservés

[1] ScSo : Histoire, Géographie, Aménagement, Urbanisme, Archéologie, Science politique, Sociologie, Anthropologie

Co-tutelle thesis manuscript submitted at INSA Lyon & University of Passau

Detecting Inference Attacks Involving Sensor Data

Paul Lachat

Advisors

Dr. Nadia Bennani
Associate Professor at INSA Lyon

Dr. Veronika Rehn-Sonigo
Associate Professor at University of Franche-Comté

Supervisors

Prof. Dr. Lionel Brunie
Professor at INSA Lyon

Prof. Dr. Harald Kosch
Professor at University of Passau

A dissertation submitted to the faculty of computer science and mathematics in partial fulfillment of the requirements for the degree of doctor of natural sciences / engineering sciences / natural philosophy

Lyon, January 2024

Cette thèse est accessible à l'adresse : https://theses.insa-lyon.fr/publication/2024ISAL0024/these.pdf
© [P. Lachat], [2024], INSA Lyon, tous droits réservés

Cette thèse est accessible à l'adresse : https://theses.insa-lyon.fr/publication/2024ISAL0024/these.pdf
© [P. Lachat], [2024], INSA Lyon, tous droits réservés

Abstract

The collection of personal information by organizations has become increasingly essential for social interactions. Nevertheless, according to the GDPR (General Data Protection Regulation), the organizations have to protect collected data. Access Control (AC) mechanisms are traditionally used to secure information systems against unauthorized access to sensitive data. The increased availability of personal sensor data, thanks to IoT-oriented applications, motivates new services to offer insights about individuals. Consequently, data mining algorithms have been proposed to infer personal insights from collected sensor data. Although they can be used for genuine purposes, attackers can leverage those outcomes, combining them with other type of data, and further breaching individuals' privacy. Thus, bypassing AC mechanisms thanks to such insights is a concrete problem.

In this thesis, we address this problem by analyzing queries users issued to a sensor database, and by identifying when they obtain sufficient information to infer insights thanks to data mining algorithms. We refer to such a kind of inference as an *Inference Attack Involving Sensor Data (IAISD)*. Detecting them strengthens individuals' data protection. When attackers query the sensor database, the important information is not so much the exact value of the obtained data points, but rather if the relevant information (e.g., type of data) are obtained according to the conditions of disclosure of such algorithms. To fulfill this objective, this thesis consists in three contributions:

Raw sensor data based Inference ChannEl Model (RICE-M) models the query history of a querying user which contains information obtained from queries, as well as the conditions of disclosure associated to an insight. RICE-M enables first the modeling of queries issued to a sensor database as a set of metadata units. Those units are built from the query parameters (e.g., selected attributes), the query context (e.g., the identity of the querying user), and the query result metadata (e.g., the number of data points). This set constitutes the *query metadata*. Second, RICE-M models both the constraints that a user's knowledge must satisfy to apply data mining algorithm and the corresponding personal insight. Those descriptions correspond to the *inference channels* attackers leverage to perform IAISDs.

The second contribution of this thesis is *RICE-M based inference detection System (RICE-Sy)*. For each user, our system maintains a history log which keeps track of the queries metadata extracted from the queries they have issued to a sensor database. When a user issues a new query, the related query metadata is extracted and processed by the system. To correctly consider the current user's knowledge, RICE-Sy retrieves from the history log the metadata units that can be merged with the newly obtained units. It then determines if those units satisfy the constraints of a described inference channel, in which case an IAISD attempt is detected. Otherwise, the user's history log is updated with the new query metadata. To efficiently filter units from the history log, we endow RICE-Sy with two conceptual optimizations: the *Query Based Filtering (QBF)* and *Search Set Filtering (SSF)*.

The last contribution of the thesis is a query metadata sequence generator which objective is to evaluate the performance of RICE-Sy. To produce realistic sequences, we identify querying behaviors by analyzing inference attack strategies and the nature of sensor databases. Based on those behaviors, we define three archetypes: the *one-time attacker*, the *genuine employee*, and the *deceptive attacker*. We demonstrate the validity of the generated datasets by providing visualizations of sequences for each archetype. Thanks to the generator outcome, we evaluate RICE-Sy in terms of detection time per query and size of the history log. The results obtained validate the efficiency of QBF and SSF, and demonstrate the feasibility of detecting IAISDs at query-time using RICE-Sy.

Keywords: Data privacy, Sensor data, Inference detection system, Query modeling, Metadata

Cette thèse est accessible à l'adresse : https://theses.insa-lyon.fr/publication/2024ISAL0024/these.pdf
© [P. Lachat], [2024], INSA Lyon, tous droits réservés

Résumé

La collecte d'informations personnelles par des organisations est devenue de plus en plus important pour les interactions sociales. Néanmoins, conformément au règlement général sur la protection des données (GDPR), ces organisations doivent protéger les données collectées. Les mécanismes de contrôle d'accès (CA) sont traditionnellement utilisés pour sécuriser les systèmes d'information contre l'accès non autorisé aux données sensibles. La disponibilité accrue des données de capteurs personnels, grâce aux applications basées sur l'IoT (Internet of Things), motive de nouveaux services à offrir des connaissances sur les individus. Par conséquent, des algorithmes d'exploration de données ont été proposés pour déduire des informations personnelles à partir des données de capteurs collectées. Bien qu'ils puissent être utilisés à des fins légitimes, les attaquants peuvent exploiter ces résultats, en les combinant avec d'autres types de données et en portant atteinte à la vie privée des individus. Le contournement des mécanismes de CA grâce à ces informations constitue donc un problème concret.

Dans cette thèse, nous abordons ce problème en analysant les requêtes que les utilisateurs adressent à une base de données de capteurs, et en identifiant le moment où ils obtiennent suffisamment d'informations pour déduire des idées grâce à des algorithmes d'exploration de données. Nous appelons ce type d'inférence "attaque par inférence impliquant des données de capteurs" (*Inference Attack Involving Sensor Data (IAISD)*). La détection de ces attaques renforce la protection des données des individus. Lorsque les attaquants interrogent la base de données des capteurs, l'information importante n'est pas tant la valeur exacte des points de données obtenus, mais plutôt si les informations pertinentes (par exemple, le type de données) sont obtenues conformément aux conditions de divulgation de ces algorithmes. Pour atteindre cet objectif, cette thèse se compose de trois contributions :

"Le modèle de canal d'inférence basé sur les données brutes des capteurs" (*Raw sensor data based Inference ChannEl Model (RICE-M)*) modélise l'historique des requêtes d'un utilisateur, qui contient des informations obtenues à partir de requêtes, ainsi que les conditions de divulgation associées à une connaissance sur un individu. RICE-M permet tout d'abord de modéliser les requêtes adressées à une base de données de capteurs sous la forme d'un ensemble d'unités de métadonnées. Ces unités sont construites à partir des paramètres de la requête (par exemple, les attributs sélectionnés), du contexte de la requête (par exemple, l'identité de l'utilisateur effectuant la requête) et des métadonnées du résultat de la requête (par exemple, le nombre de points de données). Cet ensemble constitue les métadonnées de la requête. Deuxièmement, RICE-M modélise à la fois les contraintes que les connaissances d'un utilisateur doivent satisfaire pour appliquer l'algorithme d'exploration de données et la connaissance d'un individu obtenu dans ce cas. Ces descriptions correspondent aux canaux d'inférence sur lesquels les attaquants s'appuient pour effectuer des IAISD.

La deuxième contribution de cette thèse est le "système de détection d'inférence basé sur RICE-M" (*RICE-M based inference detection System (RICE-Sy)*). Pour chaque utilisateur, notre système maintient un historique qui garde la trace des métadonnées des requêtes extraites des requêtes qu'ils ont émises vers une base de données de capteurs. Lorsqu'un utilisateur émet une nouvelle requête, les métadonnées correspondantes sont extraites et traitées par le système. Pour correctement prendre en compte les connaissances de l'utilisateur actuel, RICE-Sy extrait de l'historique les unités de métadonnées qui peuvent être fusionnées avec les unités nouvellement obtenues. Il détermine ensuite si ces unités satisfont les contraintes d'un canal d'inférence décrit, auquel cas une tentative d'IAISD est détectée. Dans le cas contraire, l'historique de l'utilisateur est mis à jour avec les nouvelles métadonnées de la requête. Pour filtrer efficacement les unités de l'historique, nous dotons RICE-Sy de deux optimisations conceptuelles : le "filtrage basé sur les requêtes" (*Query Based Filtering (QBF)*) et le "filtrage des ensembles de recherche" (*Search Set*

Cette thèse est accessible à l'adresse : https://theses.insa-lyon.fr/publication/2024ISAL0024/these.pdf
© [P. Lachat], [2024], INSA Lyon, tous droits réservés

Filtering (SSF)).

La dernière contribution de la thèse est un générateur de séquences de métadonnées de requêtes dont l'objectif est d'évaluer la performance de RICE-Sy. Pour produire des séquences réalistes, nous identifions les comportements de requêtage en analysant les stratégies d'attaque par inférence et la nature des bases de données de capteurs. Sur la base de ces comportements, nous définissons trois archétypes : l'*attaquant non récurrent*, l'*employé honnête* et l'*attaquant trompeur*. Nous démontrons la validité des ensembles de données générés en fournissant des visualisations de séquences pour chaque archétype. Grâce au résultat du générateur, nous évaluons RICE-Sy en termes de temps de détection par requête et de taille de l'historique. Les résultats obtenus valident l'efficacité de QBF et SSF, et démontrent la faisabilité de la détection des IASDs au moment de la requête à l'aide de RICE-Sy.

Mots-clés : Confidentialité des données, Données de capteurs, Système de détection d'inférence, Modélisation des requêtes, Métadonnées

Cette thèse est accessible à l'adresse : https://theses.insa-lyon.fr/publication/2024ISAL0024/these.pdf
© [P. Lachat], [2024], INSA Lyon, tous droits réservés

Zusammenfassung

Die Erfassung personenbezogener Daten durch Organisationen ist für soziale Interaktionen immer wichtiger geworden. Dennoch müssen die Organisationen gemäß der GDPR (General Data Protection Regulation) die gesammelten Daten schützen. Zugriffskontrollmechanismen (Access Control, AC) werden traditionell eingesetzt, um Informationssysteme vor unberechtigtem Zugriff auf sensible Daten zu schützen. Die zunehmende Verfügbarkeit von persönlichen Sensordaten dank IoT-orientierter Anwendungen motiviert neue Dienste, die Erkenntnisse über Einzelpersonen bieten. Folglich wurden Data-Mining-Algorithmen vorgeschlagen, um aus gesammelten Sensordaten persönliche Erkenntnisse abzuleiten. Obwohl sie für echte Zwecke verwendet werden können, können Angreifer diese Ergebnisse ausnutzen, indem sie sie mit anderen Datentypen kombinieren und die Privatsphäre von Personen weiter verletzen. Ein konkretes Problem ist daher die Umgehung von Schutzmechanismen dank solcher Erkenntnisse.

In dieser Arbeit befassen wir uns mit diesem Problem, indem wir die von den Nutzern an eine Sensordatenbank gestellten Anfragen analysieren und feststellen, wann sie dank Data-Mining-Algorithmen genügend Informationen erhalten, um Rückschlüsse zu ziehen. Wir bezeichnen eine solche Art von Schlussfolgerung als "Inferenzangriff mit Sensordaten" (*Inference Attack Involving Sensor Data (IAISD)*). Deren Erkennung stärkt den Datenschutz des Einzelnen. Wenn Angreifer die Sensordatenbank abfragen, ist die wichtige Information nicht so sehr der genaue Wert der erhaltenen Datenpunkte, sondern vielmehr, ob die relevanten Informationen (z. B. die Art der Daten) gemäß den Bedingungen für die Offenlegung solcher Algorithmen erhalten werden. Um dieses Ziel zu erreichen, besteht diese Arbeit aus drei Beiträgen:

Das "auf Rohsensordaten basierende Inferenzkanalmodell" (*Raw sensor data based Inference ChannEl Model (RICE-M)*) modelliert die Abfragehistorie eines abfragenden Nutzers, die sowohl Informationen aus Abfragen als auch die mit einer Einsicht verbundenen Offenlegungsbedingungen enthält. RICE-M ermöglicht zunächst die Modellierung von Abfragen, die an eine Sensordatenbank gestellt werden, als eine Menge von Metadateneinheiten. Diese Einheiten werden aus den Abfrageparametern (z. B. ausgewählten Attributen), dem Abfragekontext (z. B. der Identität des abfragenden Benutzers) und den Metadaten des Abfrageergebnisses (z. B. der Anzahl der Datenpunkte) gebildet. Dieser Satz bildet die Abfrage-Metadaten. Zweitens modelliert RICE-M sowohl die Einschränkungen, die das Wissen eines Benutzers erfüllen muss, um den Data-Mining-Algorithmus anzuwenden, als auch die entsprechenden persönlichen Erkenntnisse. Diese Beschreibungen entsprechen den Inferenzkanälen, die Angreifer nutzen, um IAISDs durchzuführen.

Der zweite Beitrag dieser Arbeit ist ein auf "RICE-M basierendes Inferenzerkennungssystem" (*RICE-M based inference detection System (RICE-Sy)*). Unser System führt für jeden Benutzer ein Verlaufsprotokoll, in dem die Metadaten der von ihm an eine Sensordatenbank gestellten Abfragen festgehalten werden. Wenn ein Benutzer eine neue Anfrage stellt, werden die zugehörigen Metadaten der Anfrage extrahiert und vom System verarbeitet. Um den aktuellen Wissensstand des Benutzers korrekt zu berücksichtigen, ruft RICE-Sy aus dem Verlaufsprotokoll die Metadateneinheiten ab, die mit den neu gewonnenen Einheiten zusammengeführt werden können. Es stellt dann fest, ob diese Einheiten die Bedingungen eines beschriebenen Inferenzkanals erfüllen; in diesem Fall wird ein IAISD-Versuch erkannt. Andernfalls wird das Verlaufsprotokoll des Benutzers mit den neuen Abfrage-Metadaten aktualisiert. Um Einheiten aus dem Verlaufsprotokoll effizient zu filtern, statten wir RICE-Sy mit zwei konzeptionellen Optimierungen aus: dem "Abfragebasierte Filterung" (*Query Based Filtering (QBF)*) und dem "Filterung von Suchmengen" (*Search Set Filtering (SSF)*).

Der letzte Beitrag der Arbeit ist ein Generator für Abfrage-Metadaten-Sequenzen, dessen Ziel es ist, die Leistung von RICE-Sy zu bewerten. Um realistische Sequenzen zu erzeugen,

Cette thèse est accessible à l'adresse : https://theses.insa-lyon.fr/publication/2024ISAL0024/these.pdf
© [P. Lachat], [2024], INSA Lyon, tous droits réservés

identifizieren wir das Abfrageverhalten, indem wir die Strategien von Inferenzangriffen und die Natur von Sensordatenbanken analysieren. Basierend auf diesen Verhaltensweisen definieren wir drei Archetypen: den *einmaligen Angreifer*, den *ehrlicher Mitarbeiter* und den *betrügerischen Angreifer*. Wir demonstrieren die Gültigkeit der generierten Datensätze, indem wir die Sequenzen für jeden Archetyp visualisieren. Dank der Ergebnisse des Generators bewerten wir RICE-Sy in Bezug auf die Erkennungszeit pro Abfrage und die Größe des Verlaufsprotokolls. Die erzielten Ergebnisse bestätigen die Effizienz von QBF und SSF und zeigen die Machbarkeit der Erkennung von IAISDs zur Abfragezeit mit RICE-Sy.

Schlüsselwörter: Datenschutz, Sensordaten, System zur Erkennung von Inferenzen, Modellierung von Abfragen, Metadaten

Cette thèse est accessible à l'adresse : https://theses.insa-lyon.fr/publication/2024ISAL0024/these.pdf
© [P. Lachat], [2024], INSA Lyon, tous droits réservés

Acknowledgments

Foremost, I want to express my gratitude to my two supervisors, Lionel and Harald, for accompanying me during my PhD. Particularly for all the ideas, help, and feedback you gave me. A very special gratitude goes to my two advisors, Nadia and Veronika, for your patience and invaluable guidance. I can't stress enough how happy I am to have had you as my supervisors! Special thanks are also directed to the students I supervised, specifically Hugo Le Bon and Corentin Laharotte.

Next, I want to express my thanks to all the people I met in France and Germany, from the ones I encountered at LIRIS before the Covid lockdown, during my stay in the DIMIS chair of the University of Passau, to the new faces of the DRIM team I met once coming back to INSA Lyon. An obvious thanks goes to Felix and Ashish! I won't name anyone else because I'm too afraid to miss someone. Yet, thanks for the random political/ethical/metaphysical discussions, the diverse weekend events, the culinary exchanges, and your overall presence. Special mention to Jessy, my initial research partner, Jack, for your precious help, as well as Tom, Antoine, and Pierre.

Enfin, merci à ma famille à qui je n'ai rien de mieux à dire que je vous aime !

Cette thèse est accessible à l'adresse : https://theses.insa-lyon.fr/publication/2024ISAL0024/these.pdf
© [P. Lachat], [2024], INSA Lyon, tous droits réservés

Contents

List of Figures x

List of Tables xiii

List of Algorithms xiv

List of Acronyms xv

1 Introduction 1
- 1.1 Motivating example . . . 3
- 1.2 Problem statement . . . 4
- 1.3 Challenges and research questions . . . 5
  - 1.3.1 Inference channels involving sensor data . . . 5
  - 1.3.2 Sensor data queried from the database . . . 6
  - 1.3.3 Monitoring queries to detect IAISDs . . . 6
  - 1.3.4 Evaluating the system detecting IAISDs . . . 6
- 1.4 Contributions . . . 7
- 1.5 Thesis outline . . . 7

2 Background: Sensor data mining as inference channels 8
- 2.1 Taxonomy of data mining approches . . . 9
  - 2.1.1 Type of sensor deployment . . . 9
  - 2.1.2 Type of sensor data . . . 10
  - 2.1.3 Type of mined personal information . . . 11
  - 2.1.4 Type of constraints . . . 11
  - 2.1.5 Classified references . . . 12
- 2.2 Discussion . . . 14
- 2.3 Conclusion . . . 14

3 State of the art: The inference problem in databases 16
- 3.1 Type of attributes . . . 16
- 3.2 Types of disclosure . . . 17
  - 3.2.1 Identity disclosure . . . 17
  - 3.2.2 Membership disclosure . . . 18
  - 3.2.3 Attribute disclosure . . . 20
- 3.3 Inference attacks in databases . . . 22
  - 3.3.1 Inference strategies in databases . . . 23
  - 3.3.2 Approaches to prevent inferences in databases . . . 23
- 3.4 Taxonomy and classification of the inference detection system . . . 27

Cette thèse est accessible à l'adresse : https://theses.insa-lyon.fr/publication/2024ISAL0024/these.pdf
© [P. Lachat], [2024], INSA Lyon, tous droits réservés

3.5 Positioning 35

4 RICE-M: Raw sensor data based Inference ChannEl Model 38
4.1 Case studies description: mHealth & Orange4Home 39
4.1.1 mHealth case study 39
4.1.2 Orange4Home case study 40
4.2 Capturing information to constitute the user's knowledge 41
4.2.1 Query parameters 42
4.2.2 Query context 43
4.3 Modeling the user's knowledge 44
4.4 Modeling inference channels 46
4.4.1 Concept definitions 47
4.4.2 Constraints as filters for the user's knowledge 51
4.5 Discussion 52
4.5.1 Incorporating more constraints 53
4.5.2 Logical constraints over the user's knowledge 54
4.6 Conclusion 55

5 RICE-Sy: RICE-M based inference detection System 56
5.1 Generic workflow of RICE-Sy: The Reasoner & The Knowledge Storage 56
5.2 The Reasoner module 58
5.2.1 The detection module 59
5.2.2 The consolidation of users' knowledge 61
5.2.3 The consolidation module 62
5.2.4 Filtering only the relevant subset of users' knowledge 65
5.2.5 The filtering modules 66
5.3 Complete workflow of RICE-Sy 67
5.4 Discussion 68
5.4.1 Incorporating more constraints 68
5.4.2 Filtering module 69
5.5 Conclusion 69

6 Generator: Archetypes & Query metadata sequences 71
6.1 Assumptions about users' querying behaviors 72
6.1.1 Querying behaviors 73
6.1.2 Archetypes 74
6.2 Concept definitions 74
6.2.1 Regular groups of attributes 75
6.2.2 Blocks 75
6.2.3 Periods 75
6.2.4 Timelines 76
6.2.5 Sequences of queries metadata 76
6.2.6 Interactions of concepts 76
6.3 Workflow of the dataset generation 78
6.4 Archetype-based generation of query metadata sequences 79
6.4.1 The one-time attacker (OA) 80
6.4.2 The genuine user (GU) 82
6.4.3 The deceptive attacker (DA) 86
6.5 Parameters value of query metadata sequences 90

Cette thèse est accessible à l'adresse : https://theses.insa-lyon.fr/publication/2024ISAL0024/these.pdf
© [P. Lachat], [2024], INSA Lyon, tous droits réservés

6.6 Discussion . . . 92
6.7 Conclusion . . . 92

7 Evaluation of the conceptual optimizations **94**
7.1 Metrics: Detection overhead and ConsQHL size . . . 94
7.2 Implementation of RICE-Sy & the Generator . . . 95
7.3 Evaluation methodology and settings . . . 96
7.4 Evaluations of RICE-Sy . . . 97
7.4.1 Monitoring the impact of our proposed optimizations . . . 98
7.4.2 How the query issuing order impacts RICE-Sy? . . . 104
7.4.3 How the consolidation impacts RICE-Sy . . . 108
7.4.4 How multiple users impact RICE-Sy for a fixed number of queries . . . 112
7.5 Discussion . . . 113
7.6 Conclusion . . . 116

8 Conclusion and perspectives **117**
8.1 Conclusion . . . 117
8.2 Future research perspectives . . . 119
8.3 Publications . . . 121

Bibliography **123**

Cette thèse est accessible à l'adresse : https://theses.insa-lyon.fr/publication/2024ISAL0024/these.pdf
© [P. Lachat], [2024], INSA Lyon, tous droits réservés

List of Figures

1.1 Inference attack exploiting probabilistic dependencies as an inference channel in a profile database. . . . 3
1.2 Inference attack using a first inference channel depicted in Figure 1.1 and a second in between the sensor and the profile database. . . . 4

2.1 Publication trends for personal information mined from sensor data. . . . 8

3.1 Example of a probabilistic attribute disclosure inspired from Chen et al. [31]. **LAX** is an airport, **R1** a runway, and **C5** an aircraft. The sensitive node has a dash-dotted outline. The known nodes have a red and solid outline. The unknown nodes have a dotted outline. . . . 25
3.2 Example of queries transaction generated with a lattice [139]. . . . 31

4.1 Usage context of RICE-M: When a new query issued to the sensor database, the user's knowledge is modeled and processed by the InfDS w.r.t. the known modeled inference channels. Then, the data controller is notified when an inference is detected. . . . 38
4.2 Probability distribution of classifying human activities in the mHealth inference channel. The labels L01 to L12 correspond to the following human activities: standing still, sitting and relaxing, lying down, walking, climbing stairs, waist bends forward, frontal elevation of arms, knees bending, cycling, jogging, running, jumping forwards and backwards, respectively. . . . 39
4.3 Probability distribution of classifying human activities in the Orange4Home inference channel, at home level. The labels L01 to L17 correspond to the following human activities: cleaning, computing, cooking, dressing, eating, entering, going down, going up, leaving, napping, preparing, reading, showering, using the sink, using the toilet, washing the dishes, watching tv. . . . 40
4.4 Two individuals sharing their sensor data points via two distinct data streams. . . . 42
4.5 Query issued to $\mathtt{DB}_{sen}$. . . . 42
4.6 The query metadata is valid only for the individuals sharing their sensor data. . . . 43
4.7 Query metadata extraction workflow. For readability sake, the data stream individual$_1$_stream is denoted i$_1$_s. . . . 44
4.8 A pattern filtering from a QHL the MKUs referencing an attribute, for a duration of at least one seconds, and for a quantity of data points greater than five. . . . 47
4.9 The MKUs mku_2, mku_5, and mku_8 are identified by the constrained patterns filter as the subset of the patterns which satisfies the constraint of knowing three attributes for a common time interval with a duration of at least two seconds. . . . 49
4.10 During relationship in interval algebra [5]. . . . 49

5.1 Generic workflow of RICE-Sy. The knowledge extraction is depicted in Figure 4.7. . 57

Cette thèse est accessible à l'adresse : https://theses.insa-lyon.fr/publication/2024ISAL0024/these.pdf
© [P. Lachat], [2024], INSA Lyon, tous droits réservés

5.2 Example of an initial user's knowledge. Since all the MKUs refer the same environment, it is denoted by _ for the sake of simplicity. 57
5.3 Example of new queries metadata. Since all the MKUs refer the same environment, it is denoted by _ for the sake of simplicity. 58
5.4 New query metadata, denoted by QM_{Q_6}, an the QHL obtained after RICE-Sy has processed QM_{Q_4} in Figure 5.3a. The MKUs of QM_{Q_6} are displayed lower to differentiate them from the one within the QHL. 61
5.5 Overlapping relationship in interval algebra [5]. 62
5.6 Consolidation using the *during* relationship depicted in Figure 4.10. It discards mku_7 and keeps mku_4 depicted in Figure 5.4. 64
5.7 Consolidation using the *overlapping* relationship depicted in Figure 5.5. The mku_5 and mku_9 depicted in Figure 5.4 are consolidated into *new_mku*. 64
5.8 Example of filtering ConsQHL before the consolidation or the detection. 65
5.9 Complete workflow of RICE-Sy. 68

6.1 Theoretical queries metadata dataset obtained using existing benchmarks. 71
6.2 Illustration of the concepts. 77
6.3 Workflow of the dataset generation. The indice 0 (resp. 1) denotes a sequence generated for the inference channel mHealth (resp. Orange4Home). 78
6.4 Illustrations of sequences generated for a one-time attacker. 80
6.5 Sequence generated considering the mHealth case study. 83
6.6 Sequence generated considering the Orange4Home case study. 85
6.7 Sequence generated considering the mHealth case study. 86
6.8 Sequence generated considering the Orange4Home case study. 90
6.9 Example of queries metadata dispatched on the emission timeline. 92

7.1 Temporal sections of the data controller workflow. 95
7.2 Evolution of the overhead considering a single user following the GU archetype. Median of measurements performed over 10 datasets using the same parameters. . 99
7.3 Evolution of the overhead considering a single user following the DA archetype. Measurements performed over the first dataset, among the 10 generated for mHealth, and the 10 generated for Orange4Home. Attacks are shown as vertical dotted bar. . 100
7.4 Evolution of the overhead considering a single user following the GU archetype. Median of measurements performed over 10 datasets using the same parameters. . 103
7.5 Sum of access to MKUs in the ConsQHL, with and without storing the consolidated MKUs, for varying probabilities of consolidation considering the GU archetype. Measurements of the first dataset, among the 10 generated for mHealth and Orange4home.104
7.6 Impact of varying three time the order in which queries are issued by the GU archetype. Measurements of the first dataset, among the 10 generated for mHealth and Orange4home. 106
7.7 Impact of varying three time the order in which queries are issued by the DA archetype. Measurements of the first dataset, among the 10 generated for mHealth and Orange4home. Attacks are shown as vertical dotted bar. 107
7.8 Impact of varying probabilities of consolidation for the GU archetype. Measurements of the first dataset, among the 10 generated for mHealth and Orange4home. 110
7.9 Impact of varying the number of query metadata triggering the consolidation for the GU archetype. Measurements of the first dataset, among the 10 generated for mHealth.111

Cette thèse est accessible à l'adresse : https://theses.insa-lyon.fr/publication/2024ISAL0024/these.pdf
© [P. Lachat], [2024], INSA Lyon, tous droits réservés

7.10 Impact of varying the number of users following the GU archetype for a fix quantity of queries. Measurements of the first dataset, among the 10 generated for mHealth and Orange4home. 114
7.11 Impact of varying the number of users following the DA archetype for a fix quantity of queries. Measurements of the first dataset, among the 10 generated for mHealth and Orange4home. Attacks are shown as vertical dotted bar. The bottom plots display the overhead with and without the filtering modules. 115

Cette thèse est accessible à l'adresse : https://theses.insa-lyon.fr/publication/2024ISAL0024/these.pdf
© [P. Lachat], [2024], INSA Lyon, tous droits réservés

List of Tables

2.1 References describing the usage of data mining algorithms on sensor data in order to infer personal information. . . . 14
2.2 Distribution of the references in Table 2.1 by considering the constraint and the environment for which the inferred information is valid. Timestamp based Sliding Window (TSW), Sequence based Sliding Window (SSW), Aggregation (AGG), Sampling (SAM). . . . 15

3.1 Simplified re-identification example provided in [138]. The following column names stand for Security Social Number (SSN), Data of Birth (DOB), Marital Status (MS), and Health Problem (HP). . . . 19
3.2 Probabilistic attribute disclosure example proposed by Li et al. [107]. . . . 22
3.3 Example of techniques used to perform disclosure in different source of information. DB stands for database and ML for machine learning. . . . 22
3.4 Example of generalization inspired by [138]. . . . 24
3.5 Example of a dataset perturbation presented by [94]. . . . 24
3.6 Summary of differences of assumptions between profile database and sensor database. 35
3.7 Comparison of query-time IC mechanisms. E. S. Profile, E. U. Profile, S. B. Profile, and I. a. Sensor correspond to the facet value exact static profile data, exact updatable profile data, static boolean profile fact, exact sensor data points, information about sensor data points, respectively. . . . 37

4.1 Symbols defined to model user's knowledge in RICE-M. . . . 46
4.2 Symbols defined to model inference channels in RICE-M. . . . 53

5.1 Symbols defined to formalize RICE-Sy. . . . 69

6.1 Symbols defined to generate queries metadata. . . . 78

7.1 Values of parameters for the first evaluation. *wm* stands for washingmachine. . . . 96
7.2 Percentage of prevented accesses when saving consolidated MKUs in ConsQHL, for mHealth and Orange4Home and different probabilities of blocks consolidation. . . . 102
7.3 Median overhead for mHealth and Orange4Home, according to the three iterations. 105
7.4 Median overhead for mHealth and Orange4Home, according to different probabilities of blocks consolidation. . . . 108

Cette thèse est accessible à l'adresse : https://theses.insa-lyon.fr/publication/2024ISAL0024/these.pdf
© [P. Lachat], [2024], INSA Lyon, tous droits réservés

List of Algorithms

5.1 Detecting IAISDs based on a user's global knowledge and the ICR. 60
5.2 Consolidating the MKUs between a query metadata and ConsQHL. 63
5.3 Filtering ConsQHL for the consolidation module. 66
5.4 Filtering ConsQHL for the detection module. 67
5.5 Reasoner module of RICE-Sy. 68

6.1 Generation of a query metadata sequence for the OA archetype & mHealth 81
6.2 Part 1: Generation of a sequence for the GU archetype & mHealth 83
6.3 Part 2: Generation of a sequence for the GU archetype & mHealth 84
6.4 Part 1: Generation of a sequence for the DA archetype & mHealth 87
6.5 Part 2: Generation of a sequence for the DA archetype & mHealth 87
6.6 Part 3: Generation of a sequence for the DA archetype & mHealth 89

Cette thèse est accessible à l'adresse : https://theses.insa-lyon.fr/publication/2024ISAL0024/these.pdf
© [P. Lachat], [2024], INSA Lyon, tous droits réservés

List of Acronyms

AC Access Control. 1, 24

ConsQHL Consolidated Query History Log. 63, 69, 94

CPF Constrained Patterns Filter. 49, 55, 57

CVD Cardiovascular Disease. 3

DA Deceptive Attacker. 74, 79, 91

FD Functional Dependency. 2, 21, 27, 29

GPS Global Positioning System. 10

GU Genuine User. 74, 79, 91

IAISD Inference Attack Involving Sensor Data. i, ii, iv, 5, 8, 16, 22, 38, 56, 69, 94, 118

IC Inference Control. 3, 26

ICR Inference Channel Repository. 51, 56, 69

InfDS Inference Detection System. 27, 38

IoT Internet of Things. 8

MKU Metadata Knowledge Unit. 45, 46, 55, 69

ML Machine Learning. 18

MLP MultiLayer Perceptron. 12, 40

MRD Multilevel Relational Database. 26

OA One-time Attacker. 74, 79, 91

PAL Physical Activity Level. 3

PPT Privacy-Preserving Techniques. 24

QBF Query Based Filtering. i, ii, iv, 66, 96, 119

Cette thèse est accessible à l'adresse : https://theses.insa-lyon.fr/publication/2024ISAL0024/these.pdf
© [P. Lachat], [2024], INSA Lyon, tous droits réservés

QHL Query History Log. 45, 56, 72, 119

QID Quasi-Identifiers. 17

QS Quality of Service. 94

RICE-M Raw sensor data based Inference ChannEl Model. i, ii, iv, 7, 38, 55, 56, 69, 118

RICE-Sy RICE-M based inference detection System. i, ii, iv, 7, 56, 69, 71, 94, 116, 118, 119

SSF Search Set Filtering. i, ii, iv, 66, 96, 119

SSW Sequence based Sliding Window. xiii, 12, 15, 38, 53

SVM Support Vector Machine. 40

TSW Timestamp based Sliding Window. xiii, 12, 15, 38, 53

Cette thèse est accessible à l'adresse : https://theses.insa-lyon.fr/publication/2024ISAL0024/these.pdf
© [P. Lachat], [2024], INSA Lyon, tous droits réservés

Chapter 1

Introduction

With the advances in technology, the sharing of personal information with organizations has become increasingly essential for social interactions. According to [12], in 2019 more than 60% of US adults considered that they could have a daily life without sharing data with companies or with the government. We observe that 73% of US companies and 71% of European companies collected personal data in 2021 [73]. With customers' consent, organizations collect and store personal data to provide personalized services or to generate profit by selling that information to data brokers [39]. In 2021, according to [58], 87% of companies rely on third parties to process the collected data. Among them, only 14% perform an on-site audit to ensure the third party capability to protect data [58]. This situation leads to 79% [12] of US adults stating that, in 2019, they are "very or somewhat" concerned about the usage of their shared data by companies. To gain credibility and encourage the customers to share their data, organizations have to protect customers from attacks which aim to disclose personal data without their consent.

In Europe, the legal framework of *data controllers* is defined by the GDPR (General Data Protection Regulation) [69]. According to the GDPR, the "*personal data* are defined as any information relating to an identified or identifiable natural person (*data subject*); an identifiable natural person is one who can be identified, directly or indirectly, in particular by reference to an identifier such as a name, an identification number, location data, an online identifier or to one or more factors specific to the physical, physiological, genetic, mental, economic, cultural or social identity of that natural person". A data controller is a "natural or legal person, public authority, agency or other body" which defines for which purpose and how collected personal data are processed. They must ensure that, "[…] by default personal data are not made accessible without the individual's intervention to an indefinite number of natural persons".

To achieve the principle of protection by design and by default, solutions have been proposed to consider the consent provided by a data subject to share their[1] data with data controllers. For instance, Gerl et al. [70] have proposed a layered privacy language which incorporates the legal requirements in the policy to offer fine-grained access-control. Indeed, Access Control (AC) mechanisms [131] have been traditionally used to secure information systems against unauthorized access to sensitive data. Whereas those direct access are controlled (e.g., selecting a specific attribute in a database), sensitive data may be indirectly accessed. An *indirect access* [57] to a sensitive data occurs when one exploits an *inference channel* using non-sensitive data and information such as the existence of data dependencies, integrity constraints, etc.

Definition. An **inference channel** is the capability to infer sensitive data from non-sensitive data [150].

[1] In this thesis, we use the *singular they* as a generic pronoun.

Cette thèse est accessible à l'adresse : https://theses.insa-lyon.fr/publication/2024ISAL0024/these.pdf
© [P. Lachat], [2024], INSA Lyon, tous droits réservés

For instance, in a relational database a Functional Dependency (FD) can be combined with non-sensitive attributes (e.g., an employee's rank and name) to infer a sensitive attribute (e.g., this employee's salary). By exploiting an inference channel, one performs an *inference attack*. However, AC mechanisms are not capable of controlling indirect access to data [57], thus data subjects are not protected from inference attacks by AC mechanisms. Yet, this type of attack is identified as a privacy threat in systems such as databases [89] or the IoT [98].

Notorious examples of such attacks are the AOL search log release and the Netflix prize. In 2006, the New York Times reported [1] that journalists managed to infer the identity of an individual, based on their queries among the 20 million Web search queries publicly published by AOL. Similarly, in the context of a competition organized by Netflix, a training dataset containing film ratings was publicly published. Narayanan et al. [120] propose an algorithm to match the individuals' public record in IMDb[2] with the anonymized record in the Netflix dataset. By doing so, the authors are able to de-anonymize two individuals by correlating the rating of movies and the dates of ratings, respectively. The two records are 28 standard deviations and 15 standard deviations away from the second-best candidate, respectively. Apart from those two examples, we observe in the scientific literature that inference attacks are identified for information sources as diverse as computing devices [151] (e.g., the 4096-bit RSA decryption key can be inferred from a device sound), social networks [126] (e.g., the age and gender can be inferred based only on the link structure between friends), machine learning models [87] (e.g., the presence of a data instance in the training set of a model can be inferred), or databases [89] (e.g., the salary of an employee can be inferred via a functional dependency). Therefore, inference attacks are a critical privacy threat in information society, which are furthermore strengthened by the rise of a new category of data originating from sensors. Nowadays, the constantly expanding market of consumer electronic sensors [38] implies that more and more individuals own wearable sensors (e.g., embedded in smartphones), or ambient sensors (e.g., deployed in their homes).

The rising availability of personal sensor data motivates new services to offer insights about: customers' health via devices such as wrist-bands [60] or anomalies in elderly home activities [149]. In recent years, various data mining algorithms have been proposed to infer new personal data from collected sensor data. While those algorithms can be used for genuine purposes, they can also be leveraged by attackers to gain insights, and to breach individuals' privacy [99, 98]. For instance, Banos et al. [15] and Chikhaoui et al. [33] demonstrated using the *mHealth* and the *Opportunity* datasets, respectively, that one can infer human activities (e.g., walking, cycling, etc.) from specific wearable sensor observations, if the observation interval exceeds a minimum temporal duration. Cumin et al. [42] and Shahi et al. [140] demonstrated on the *Orange4Home* and the *CASA-Aruba* datasets, respectively, how daily human activities can be inferred from ambient sensors, thus considering the spatial environment in which sensors are deployed. Aside from human activities, sensor data can be exploited to infer the behavior of individuals [37], the PIN code of a smartphone [112, 61, 143] (e.g., using the microphone to infer the keystrokes), or even to infer the individual's identity for authentification purpose [2] (e.g., using keystrokes dynamics). In all those examples, the data are produced by wearable sensors, or ambient sensors deployed in a single inhabitant home. They are considered as personal data, since they are related to an identifiable data subject.

As a result, data controllers have to collect personal data related to the same individual, from different data sources. The first one is the individual themselves providing their information (e.g., when creating an account on a Website). The second one are the sensors related to this individual (e.g., the accelerometer in their smart-watch). To differentiate between the two types of data, in this thesis we refer to them as the *profile data* and the *sensor data*, respectively. Usually, attackers perform inference attacks on a single data source to infer a stored sensitive data. In presence

[2] An online database of information related to movies and other multimedia contents.

Cette thèse est accessible à l'adresse : https://theses.insa-lyon.fr/publication/2024ISAL0024/these.pdf
© [P. Lachat], [2024], INSA Lyon, tous droits réservés

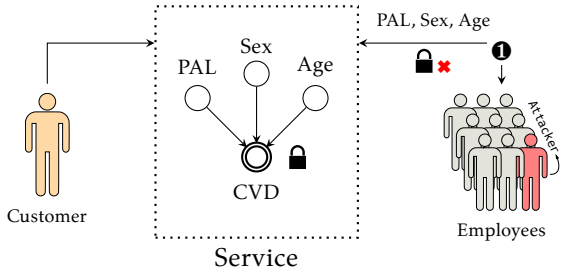


Figure 1.1: Inference attack exploiting probabilistic dependencies as an inference channel in a profile database.

of sensor data, they may exploit this new type of information to infer a personal information (e.g., an individual's activities), and then combine it with non-sensitive profile data to infer a profile sensitive data. In the following section we provide a complete example to illustrate this situation and to motivate the importance of controlling inferences stemming from sensor data.

1.1 Motivating example

Let us consider the following example where a recommendation service aims to enable its customers to stay healthy. According to Venkatachalam et al. [154], recommender systems provide goals or predict the needs of individuals who voluntarily use them "[…] as a practice or habit, for health reasons, or as a self-made goal". For instance, the habits may be related to eating more healthy dishes [22] or to perform adapted physical activities, as presented by Venkatachalam et al. In order for the service to recommend suitable activities, all the information constituting the customers' profile are assumed to be manually provided during subscription. To do so, the service collects and stores in a relational database ($\mathtt{DB}_{\mathrm{pro}}$) the following information: Age, Sex, Physical Activity Level (PAL) and Cardiovascular Disease (CVD) status. In parallel, to provide more context to the recommendations and let them deliver more accurate advises, sensor data are collected with the purpose of inferring customers' activities [15]. We assume that wearable sensors are provided to the service customers by an external company. The mission of this is to calibrate the customers' sensors based on their profile. The produced data are accessed via the sensor database [17] ($\mathtt{DB}_{\mathrm{sen}}$) managed by the service only. Based on a customer's profile data and inferred activities, the service recommends the most healthy and suitable exercises to them. In the following, we refer to: $\mathtt{DB}_{\mathrm{pro}}$ as the *profile data database* or profile database; $\mathtt{DB}_{\mathrm{sen}}$ as the *sensor data database* or sensor database.

Among all the collected data, only the CVD is considered as *sensitive personal data*. To protect this data, the service deploys an AC mechanism on $\mathtt{DB}_{\mathrm{pro}}$. This mechanism prevents unauthorized access to the CVD. Moreover, since the threat of inference attacks is well known for profile data [57], the service also deploys an Inference Control (IC) mechanism on $\mathtt{DB}_{\mathrm{pro}}$. This second mechanism monitors authorized access to non-sensitive data and prevents queries to exploit an inference channel. Both mechanisms are depicted as a lock in Figure 1.2. Like multiple companies [58], we assume that the service relies on a third party to manage some collected data. The service depends on the external company providing the sensors, to efficiently store sensor data ($\mathtt{DB}_{\mathrm{sen}}$). Employees of the external company all have authorized access to the $\mathtt{DB}_{\mathrm{sen}}$ and $\mathtt{DB}_{\mathrm{pro}}$, hosted by the external company and by the service, respectively. The sensitive personal data is not shared with the third party. According to the GDPR, in our example the service acts as the *controller*, the customers are the *data subjects*, and the third party's employees are

Cette thèse est accessible à l'adresse : https://theses.insa-lyon.fr/publication/2024ISAL0024/these.pdf
© [P. Lachat], [2024], INSA Lyon, tous droits réservés

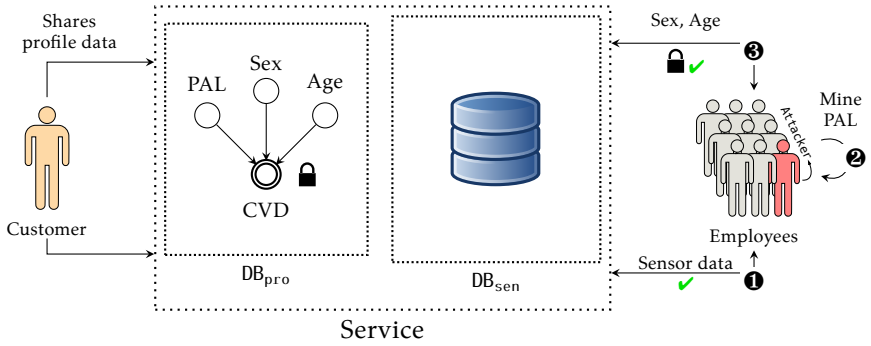


Figure 1.2: Inference attack using a first inference channel depicted in Figure 1.1 and a second in between the sensor and the profile database.

the *recipients*. In such a setting, the attacker is an employee aiming to breach the privacy of a customer, by obtaining the value of their CVD status. This type of threat is known as the *external insider* [85] in the literature. According to the European Union Agency for Cybersecurity (ENISA) [53], databases are the most vulnerable asset to the insider threat. In our example, their motivation here, can be to sell the customer's health information to data brokers.

Due to the AC mechanism implemented on $\mathtt{DB}_{\mathrm{pro}}$, the attacker is unable to directly query the CVD. Instead, they will infer the CVD status of a customer. To do so, the attacker uses an available scientific literature to determine that, based on the Age, Sex, and PAL of an individual, one can infer their cumulative incidence of having a CVD [100]. This corresponds to an inference channel that the attacker exploits by querying the three non-sensitive data in $\mathtt{DB}_{\mathrm{pro}}$. Assuming that this channel is known by the IC (i.e., the lock), the query is prevented to protect the customer's privacy ❶ in Figure 1.1.

The attacker now attempts to infer the CVD by combining the two source of data, i.e., $\mathtt{DB}_{\mathrm{pro}}$ and $\mathtt{DB}_{\mathrm{sen}}$. Once again, they consult a scientific article to determine that human activities can be recognized from sensor data (i.e., produced by accelerometers, gyroscopes, and magnetometers) [15]. As illustrated in Figure 1.2, the attacker queries from $\mathtt{DB}_{\mathrm{sen}}$ the relevant targeted customer's sensor data ❶, and infer their activities. Then, the attacker can exploit a last article to infer the customer's PAL ❷ based on known correlation with the duration of each deduced activities [23]. The attacker exploits this second inference channel between sensor data and the PAL. Finally, the attacker queries the Age and Sex information from $\mathtt{DB}_{\mathrm{pro}}$. The IC mechanism answers the query ❸, since the information are not sufficient to exploit the first inference channel. The attacker has all the required data to infer the sensitive CVD status.

1.2 Problem statement

The problem addressed in this thesis is associated to the inference of personal information using individual's sensor data stored in the sensor database. By combining this knowledge acquired from these data with personal data in profile databases to which access is allowed, attackers bypass the IC mechanism protecting a profile database, leading to a *personal data breach* [69]. Without this problem, data collectors could handle sensor data, while adhering to the GDPR. The escalation of services collecting sensor data, and the growing research aiming to infer personal information from sensors, amplifies the prominence of this form of privacy breach [98].

While the inferred personal information itself may not be sensitive (e.g., the PAL), attackers can combine them with other data. Detecting information inferred from sensor databases

Cette thèse est accessible à l'adresse : https://theses.insa-lyon.fr/publication/2024ISAL0024/these.pdf
© [P. Lachat], [2024], INSA Lyon, tous droits réservés

becomes imperative to enable IC mechanisms in other databases (e.g., the lock of DB_{pro} depicted in Figure 1.2) to effectively control how individuals' data is accessed. We refer to such an inference as Inference Attack Involving Sensor Data (IAISD). In this research, our focus lies on the detection process of IAISDs that should be set up on sensor data in order to prevent personal data breaches illustrated in the motivating example. Such process could be combined with a conventional IC to enhance the prevention capabilities of a data controller. The prevention part is out of scope of this thesis. The goal of this thesis is to present our contributions towards an IAISD detection system. Next, we describe the challenges and research questions associated to our goal.

1.3 Challenges and research questions

Before describing, in the following sections, the challenges that lead to each of our four research questions, we define the terms *individual*, *user*, and *attacker* used through this thesis:

- An **individual** is a person sharing the sensor data collected by data controllers.
- A **user** is a person having an authorized access to sensor data managed by data controllers (e.g., the employees in Figure 1.2).
- An **attacker** is a malicious user which aims to breach the privacy of an individual.

1.3.1 Inference channels involving sensor data

To perform an IAISD, the attacker exploits an inference channel. In our motivating example, the attacker queries the sensor database according to the data mining process they use. For instance, to infer the PAL of a customer as depicted in Figure 1.2, the attacker infers first this customer's human activities. To obtain this personal information, the attacker knows that, according to Banos et al's work [15], they have to query the data points produced by the customer's accelerometer, gyroscope, and magnetometer sensors, during the same time interval having a duration of at least 2 s. The attacker has queried the necessary data to replicate Banos et al's inference.

Given the wide range of mining algorithms proposed for various purposes, there is a large variety of approaches used to prepare data used to train ML models (e.g., during the pre-processing phase where data points are divided into sliding windows). For instance, Cumin et al. [42] demonstrate that, by considering vectors of 20 data points corresponding to multiple measurements (e.g., CO_2 level, luminosity, …) produced during the same time interval of 15 s, the human activities performed by a individual in an appartement can be inferred from ambient sensors. While our motivating example is based on the example of human activities, other personal information can be targeted. For example, Hart et al. [81] present how by computing the mean, the maximum of the absolute readings, and other aggregations from the data points generated by the accelerometer and gyroscope sensors embedded in their smartphone, the emotions (e.g., fatigue, mood, arousal, etc.) of an individual can be inferred.

Consequently, the system detecting IAISDs must know which inference channels the attackers can exploit by querying a sensor database. This entails representing under which conditions a channel can be exploited.

RQ1 How to model the inference channels exploited in IAISDs?

The detection system needs to be able to model the inference channel by capturing its characteristics: what sensor data is used, under what conditions personal information is disclosed, what information is disclosed, etc.

Cette thèse est accessible à l'adresse : https://theses.insa-lyon.fr/publication/2024ISAL0024/these.pdf
© [P. Lachat], [2024], INSA Lyon, tous droits réservés

1.3.2 Sensor data queried from the database

When the attacker queries the sensor database, the important information is not so much the exact value of the queried data points, but rather if the relevant type of data (e.g., acceleration, rotation, etc.), quantity of data points, … are obtained according to the condition of disclosure of an inference channel (e.g., the 2 s in Banos et al's work).

The information inferred when performing an IAISD is assumed to be personal, i.e., associated to a data subject. For instance, in the motivating example, the attacker queries sensor data related to the targeted individual (depicted on the left side of Figure 1.2) and infers their PAL. The attacker then queries the Age and Sex associated to the same individual to combine information related to the same data subject.

In the example of Section 1.1, we explain that the attacker queries all the necessary sensor data in one query. However, the same inference channel may be exploited differently by distinct attackers. For instance, another attacker than the one depicted in the motivating example may issue, in order,: (i) a query to select all the accelerometer data produced during a time interval of 2 s, (ii) another query for all the gyroscope data generated in this interval, (iii) a last query to select all the magnetometer data produced in the same interval. By combining the three results, the attacker can infer the human activities, similarly to the one depicted in the motivating example.

Hence, according to the modeled condition of disclosure of all known inference channels, for each user querying the sensor database, the system detecting IAISDs must capture the characteristics of the queried data and the individual associated to those sensor data.

RQ2 How to model the queried sensor data used in IAISDs?

The detection system needs to be able to model the knowledge acquired by users interrogating sensor data. The proposed model must capture the identity of the individual whose data was queried, as well as information about the queried data (e.g., what type of data, how many data points, …).

1.3.3 Monitoring queries to detect IAISDs

As shown in the motivating example, the queries issued to $\mathtt{DB}_{\mathrm{pro}}$ are monitored by the IC mechanism (see ❸ in Figure 1.2). At the opposite, the queries targeting $\mathtt{DB}_{\mathrm{sen}}$ are not monitored, the queries selecting sensor data are not monitored, hence enabling the attack described in the motivating example. The attacker can either obtain all the information they need for the attack using a single query (e.g., in the motivating example all the required sensor data are obtained via a singled query on $\mathtt{DB}_{\mathrm{sen}}$), or asking for the same data using several queries. In both cases, the system should be able to detect the IAIDSs on $\mathtt{DB}_{\mathrm{sen}}$.

RQ3 How can IAISDs be detected by a system?

The system must be able to detect an IAIDS, taking into account the modeled inference channels and the queried knowledge, regardless of the number of requests issued by the attackers.

1.3.4 Evaluating the system detecting IAISDs

As previously observed, the queries targeting $\mathtt{DB}_{\mathrm{sen}}$ are not monitored by a detection system. To the best of our knowledge, no existing datasets — used to evaluate the detection system protecting

Cette thèse est accessible à l'adresse : https://theses.insa-lyon.fr/publication/2024ISAL0024/these.pdf
© [P. Lachat], [2024], INSA Lyon, tous droits réservés

DB_{pro} (e.g., Toland et al. [150]), systems querying sensor databases (e.g., TPCx-IoT [127]), and so on — can be used to evaluate the proposed detection system. Besides the multiplicity of approaches attackers can use to query sensor data (i.e., via different combination of queries), the users that interact with the DB_{sen} can have genuine querying behaviors (e.g., some employees in Figure 1.2). The system detecting IAISDs needs to be evaluated with datasets that capture this diversity of approaches.

RQ4 What datasets can be used to evaluate our detection system?

A suitable dataset should contain query sequences targeting data points related to certain inference channels known to the system. These query sequences should correspond to realistic behaviors of genuine and malicious users.

Next, we present the contributions associated to our four research questions.

1.4 Contributions

In this thesis we present three contributions with the objective of tackling the problem exposed in Section 1.2. The first contribution is Raw sensor data based Inference ChannEl Model (RICE-M), a new model which is formalized in two parts associated to RQ1 and RQ2, respectively:

(i) The first one formalizes the representation of the personal information inferred via a channel and the condition for which this disclosure occurs.

(ii) The second part formalizes how the sensor data obtained by both attackers and non-attackers is represented based on the issued queries (e.g., the name of a selected attribute) and the queries results (e.g., the quantity of obtained data points).

The second contribution is called RICE-M based inference detection System (RICE-Sy). To address RQ3, we propose a detection system which implements the detection of IAISDs at the time queries are issued. The third contribution is a query generator which provides us with sequences of queries. This contribution tackles the lack of datasets satisfying the RQ4. We describe the methodology used to generate sequences of queries that correspond to different querying behaviors

1.5 Thesis outline

The remainder of this manuscript is organized as follows: as an background introduction, in Chapter 2 we briefly provides examples of inference channels used in IAISDs and describe their specificities. In Chapter 3, we present the existing IC mechanisms proposed to detect inference attacks in databases; highlight their limitation regarding our research questions; position our contribution w.r.t. those solutions. We provide in Chapter 4 the formal description of our first contribution, i.e., RICE-M. The proposed formalization is illustrated using the mHealth [15] and Orange4Home [42] case studies. In Chapter 5, we present the generic workflow of our second contribution, i.e., RICE-Sy, as well as a set of conceptual optimizations which limit the quantity of data considered at query-time, during the detection. To evaluate RICE-Sy, we describe in Chapter 6 the methodology used to generate synthetic query metadata. Based on those synthetic data, we evaluate in Chapter 7 the conceptual optimizations proposed in Chapter 5, using our case studies. Chapter 8 summarizes our contributions and presents future work.

Cette thèse est accessible à l'adresse : https://theses.insa-lyon.fr/publication/2024ISAL0024/these.pdf
© [P. Lachat], [2024], INSA Lyon, tous droits réservés

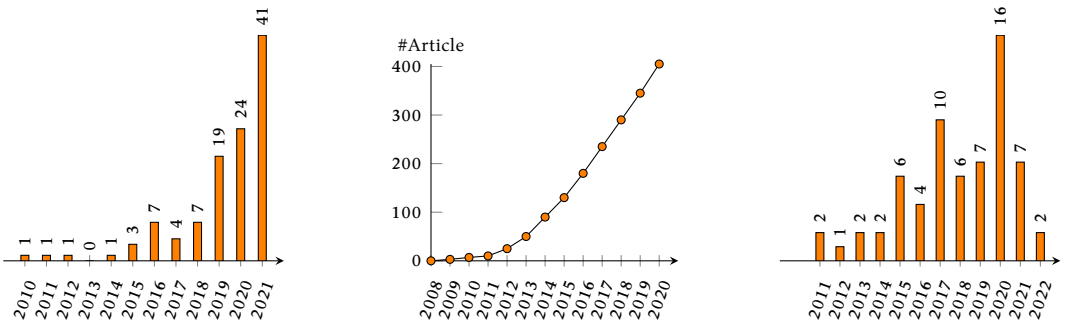
Chapter 2

Background: Sensor data mining as inference channels

As presented via the motivating example in Section 1.1, the attacker is assumed to be an *external insider* [85] which "consists of third-party personnel, such as contractors and suppliers, as well as internal employees with limited authorized access to various compartments inside an organization, such as, security guards, servicemen, and cleaners". In the literature, it is commonly assumed that the attacker possesses certain background knowledge concerning the dependencies between the data used to perform an inference attack [57]. In the context of an Inference Attack Involving Sensor Data (IAISD), there are differences in the assumptions associated with this knowledge acquisition process. It is worth describing how to mine personal information sensor data that are used as background knowledge by attackers, even if they are initially proposed for a genuine purpose.

With the increasing trend of Internet of Things (IoT), a lot of solutions have been proposed to extract meaningful information from sensor data. Data mining approaches have been widely proposed for various domains and objectives [147], such as:

- Smart-home, ambient assistant living, and smart healthcare [109]: These domains utilize



(a) Number of articles found by year of publication for the query *IoT and data mining* [170].

(b) Cumulative number of peer-reviewed articles on human activity recognition using smartphone sensors [145].

(c) Number of articles found by year of publication considered in this chapter.

Figure 2.1: Publication trends for personal information mined from sensor data.

Cette thèse est accessible à l'adresse : https://theses.insa-lyon.fr/publication/2024ISAL0024/these.pdf
© [P. Lachat], [2024], INSA Lyon, tous droits réservés

data mining techniques for human activity recognition and identification, energy conservation, safety and security measures, healthcare emergency detection, and detection of emotions or behavior deviations.

- Smart-transportation [141]: This domain leverages data mining for traffic sign detection and recognition, traffic prediction and forecasting, transportation service planning, and driver behavior analysis.

Since over more than 10 years, the availability of such works has been intensified, as depicted in Figure 2.1. Zhong et al. [170] observe in Figure 2.1a that, just from 2020 and 2021, the quantity of articles that focuses on data mining in IoT has doubled. Even by focusing on data mining using smartphone sensors, Straczkiewicz et al. [145] report a continuous increase of published articles since 2013, as illustrated in Figure 2.1b. This increasing availability and diversity of those new data mining solutions shows the relevance of detecting IAISDs.

In order to illustrate the specificities of the inference channels we have to model to tackle RQ1, we describe the diversity of approaches used to mine personal information from sensor data. To present a comprehensive set of references, we selected articles from recent surveys [164, 102, 145], as well as well-known sensor datasets used to train ML models (*UCI-HAR* [7], *mHealth* [15], *REDD* [97], etc.). We extracted the solutions presented as the most efficient to mine personal information, or the newest in case of similar performances. The final list contains a total of 85 references published between 2011 and 2022. Although this list is not exhaustive, we also notice in Figure 2.1c that more articles have been published in recent years.

This chapter is organized as follow: we present in Section 2.1 the facets we consider to classify those references and, for each, we describe the related works. At the end of this section, we compile this classification in Table 2.1. In Section 2.2, we discuss the families of references that we consider in the remainder of this thesis. We conclude this chapter in Section 2.3 by providing a summary of our classification.

2.1 Taxonomy of data mining approches

To analyze the specificities of the inference channels considered in IAISDs, we classify the selected references using four main facets, i.e., the type of: (i) sensors (ii) produced sensor data (iii) mined personal information (iv) constraints related to the processed data. In the following, we present those facets, we explain how we have defined their respective values, and we illustrate each of them with associated references. Furthermore, the information in bold and between parenthesis represents the value used to classify references in the table at the end of this section.

2.1.1 Type of sensor deployment

The first facet is related to the observation that sensors are either wearable or ambient sensors:

- **Wearable (W)** sensors. For instance, Lee et al. [106] use the *WISDM* dataset [158] produced using sensors from a smartphone and a smartwatch.

- **Ambient (A)** sensors. For instance, Shahi et al. [140] leverage the *CASA-Aruba* dataset[1] produced using motion sensors, door closure sensors, and temperature sensors deployed in a smart-home with a single inhabitant.

[1]Published the 07/07/2011 in casas.wsu.edu/datasets.

Cette thèse est accessible à l'adresse : https://theses.insa-lyon.fr/publication/2024ISAL0024/these.pdf
© [P. Lachat], [2024], INSA Lyon, tous droits réservés

A few works consider both **wearable and ambient (W & A)** sensors. For instance, Cumin et al. [43] leverage the *Opportunity* dataset [46] produced with 17 wireless sensors attached to the individual and 33 ambient sensors fixed to objects (e.g., a coffee machine) and furniture (e.g., a drawer) in a room representing a studio flat.

We consider this facet, since in IAISDs the relationship between individuals and the inferred information depends on the surroundings of sensors. For instance, a data mining process based on wearable sensor data yield an information related to the individual wearing the considered the sensors. Likewise, a process based on ambient sensor data implies that the information is linked to all the individuals present in the area where the sensors are installed. Considering the type of sensor is necessary to determine the *environment*[2] for which an inferred personal information is valid. Due to the focus on a specific type of sensors, the fact that the mined information is valid only for a environment is inherent in those works, even through it isn't explicitly addressed.

2.1.2 Type of sensor data

The second facet corresponds to the observation in our references, the following five types of physical observations are considered:

- **Inertial (In)** data. For instance, the WISDM dataset contains acceleration and rotation data on the x, y, and z axis obtained from a smartphone and a smartwatch. Fu et al. [64] exploit the step count measured by a stepometer.

- **Location (Lo)** data. Asim et al. [11] use the *Extrasensory* [152] dataset contains, among other data, the rotation data on three axis from a smartphone, the compass measurements of a watch, the Global Positioning System (GPS) data from location services, and the number of WiFi access points.

- **Ambient (Am)** data. The CASA-Aruba dataset contains pressure information to determine if a door is close and the temperature of the flat. Cumin et al. [44] propose the *Orange4Home* dataset which contains measurements such as the humidity, infra-red proximity, CO_2 level, luminosity, … in a smart-home. Liu et al. [111] use a private dataset containing the WiFi channel state information (i.e., how the signal scatters, fades, or decays) related to the signal emitted by a router and received by three smartphones.

- **Multimedia (Mu)** data. Lian et al. [109] have collected their own data containing the frequency modulated continuous wave signal produced by speakers.

- **Biological (Bi)** data. For example, Sabry et al. [136] have collected their own public dataset. It contains, among other data, the galvanic skin reaction (i.e., the electrical variation in the human skin) and the photoplethysmography (i.e., the blood volume changes) of individuals. Clarke et al. [37] have collected, among other data, the hearth rate of volunteers using wearable sensors.

We consider this facet to illustrate the diversity of sensor data exploited in the literature. Moreover, we show that the increasing availability of all kind of sensor data have a privacy-invading potential [98] which can be reinforced if IAISDs are not tackled.

[2]This concept is formalized in the following chapter.

Cette thèse est accessible à l'adresse : https://theses.insa-lyon.fr/publication/2024ISAL0024/these.pdf
© [P. Lachat], [2024], INSA Lyon, tous droits réservés

2.1.3 Type of mined personal information

The third facet corresponds to the type of personal information mined in the different references:

- **Human activities (Act)**. To enhance individuals' life and to enable healthcare at home for elderly and dependent people, a large body of work aim to infer human activities [102]. For instance, Akhavian et al. [3] infer construction workers' activities to control and manage more efficiently projects. Other works such as Arifogl et la. [8] recognize activities to identify abnormal one, which could be caused by dementia. To improve individuals' health, Fu et al. [64] propose to monitor fitness level via the inference of performed activities.
- **Appliance usage (App)**. The usage of smart metering to control electrical consumption in household creates concern about its privacy impact. As presented by Eibl et al. [51], those sensors are usually used to provides energy feedback to individuals via *non-intrusive appliance load monitoring analyze*. Hevesi et al. [83] infer the appliance operation mode (e.g., opened refrigerator, the usage of a water boiler, and so on). Knowing which appliance is used when can be exploited to infer further personal information, such as human activities.
- **Health**. With the worldwide increase of elderly population, the smart healthcare is a common technical answer proposed to support patients [147]. For instance, long term exposure to stress can lead to social isolation, help developing health problem, etc. Maxhuni et al. [114] infer individuals' stress level for the purpose of monitoring their well-being. Other information such as the emotion can help handling an individual's well-being. Hossain et al. [86] recognize emotions such as anger, disgust, fear, happiness, sadness, surprise, etc. to enable robotic patient therapy where robots provide feedback based on the patient's emotions. Further personal information, such as detecting: falls to reduce the rescue period [14]; the risk of cardiac arrest related to respiration disease [111]; the possibility of mental health condition [165]; drinking problems associated to heavy drinking episodes [13]; the dehydration level of elderly peoples that do not feel thirst [136].
- **Navigation (Nav)**. To facilitate the navigation of individuals in urban areas, transportation systems have to take into account how their users' move and what are their behavior. Sharma et al. [141] infer individuals' travel mode (i.e., stationary, walk, bicycle, car, bus, and subway) so that those systems can determine each user's movement. Bejani et al. [16] recognize the driving style of individuals to help them reduce the fuel consumption, for instance.
- **Localization (Loc)**. The location-based systems have a common requirement, to provide the current user positioning. The GPS may be sufficient for outdoor, but not sufficient for accurate indoor environment [54]. For instance, Zhang et al. [166] infer an individual's indoor location to improve the location-based systems. Yet, to determine which system to use, one needs to known if an individual is indoor or outdoor. Kelishomi et al [54] recognize the environment in which the users of such system is located.

This facet is important to illustrate both the multiplicity of motivation leading the discovery of new IAISDs attackers can leverage; the heterogeneity, and thus the challenge of modeling the personal information in inference channels, associated to RQ1.

2.1.4 Type of constraints

The last facet corresponds to the observed techniques used to segment sensor data and extract features to enable personal information inference [66, 65]:

Cette thèse est accessible à l'adresse : https://theses.insa-lyon.fr/publication/2024ISAL0024/these.pdf
© [P. Lachat], [2024], INSA Lyon, tous droits réservés

- **Aggregation (AGG)**. Statistical measures are used to summarize features of the generated sensor data. For instance, Arora et al. [9] use wearable sensors producing inertial data to infer if individuals' have Parkinson's disease. They extract 23 features an accelerometer data points, ranging from the mean and the standard deviation, to the entropy and the cross-correlation between the acceleration on the x and y-axis. Arora et al. are then able to discriminate individuals having Parkinson's disease with an accuracy of 98% using a random forest model.
- **Sampling (SAM)**. Data points are selected, deterministically or randomly, to be analyzed. For example, Anagnostopoulos et al. [6] use wearable sensor generating inertial, ambient, and location data to infer if an individual is indoor or outdoor. They use a deterministic sampling method which consist in retaining only 1 data point every x s. While this reduce the accuracy of detecting change of environments the larger x is, it enables them to reduce the energy consumption of the smartphone mining the personal data.
- **Sliding Window**. A sub-sets of data points generated in a window are analyzed. The main assumption which motivated the usage of sliding windows is that the recently generated sensor data are more relevant than historical data for the mining process. The two kinds of windows are:
  - **Sequence based Sliding Window (SSW)**. It is defined by the number of data points that it contains. For instance, Cumin et al. [44] use ambient sensor generating inertial, location, and ambient data to recognize human activities. They divide the generated data in windows containing 20 data points. The authors are able to infer activities with a F1 score of 93% using a MultiLayer Perceptron (MLP) model.
  - **Timestamp based Sliding Window (TSW)**. It is defined by a time interval. For example, Banos et al. [15] use wearable sensor generating inertial data to infer human activities. They segment the sensor data in non-overlapping windows having a duration of 2 s. This data preparation enables them to obtain a F1 score of 98% with a decision tree model.

We consider this facet, since in addition to selecting specific types of sensor data described in an article, to perform an IAISD, the attacker must query sensor data points so that they can replicate the techniques used to prepare data in the article. For instance, if the expected input of a data mining process is a window containing data points generated for a specific duration, then the attacker must query the required type of sensor data produced during a common interval of this duration. Hence, the techniques act as *constraints*[3] with regard to how attackers must query sensor data. In the following section, we show the final classification of our collected references.

2.1.5 Classified references

Table 2.1 compiles all our references. Each column corresponds to the presented facets: (i) **Depl.** describes the type of sensor deployment from Section 2.1.1. (ii) **Sensor data** describes the type of sensor data from Section 2.1.2. (iii) **Mined data** describes the type of mined personal information from Section 2.1.3. (iv) **Constraint** describes the type of constraints from Section 2.1.4 (v) **Dataset** is an additional facet describing the name of the public dataset used by a reference. Private and public untitled datasets are denoted by C+ and C-, respectively.

[3]This concept is formalized in the following chapter.

Cette thèse est accessible à l'adresse : https://theses.insa-lyon.fr/publication/2024ISAL0024/these.pdf
© [P. Lachat], [2024], INSA Lyon, tous droits réservés

	Depl.	Sensor data	Mined data	Constraint	Dataset
[15]	W	In	Act	TSW, AGG	mHealth
[93]	W	In, Lo	Act	AGG	mHealth
[43]	W, A	In, Lo, Am	Act	SSW	Opportunity
[44]	A	Am, Mu	Act	TSW, SSW	Orange4Home
[33]	W	In	Act	TSW	Opportunity
[95]	W	In	Act	TSW, AGG	UCI-HAR
[55]	A	Am	Act	AGG	CASA-Kyoto
[140]	A	Am	Act	TSW	CASA-Aruba
[56]	A	Am	Act	AGG	CASA-Multiresident
[8]	A	Am	Health	TSW	Van Kasteren
[51]	A	Am	App, Act	SSW	REDD
[83]	A	Am	App, Act	TSW, AGG	C-
[122]	A	In, Am	Act	TSW, AGG	C-
[37]	W	Bi	Health	TSW	C-
[4]	A	Am	Act	SSW	ARAS
[157]	W	In, Am, Mu	Act	AGG	C-
[86]	A	Mu	Health	TSW	RML, eNTERFACE
[6]	W	In, Am, Lo, Mu	Loc	SAM, TSW	C-
[9]	W	In	Health	AGG	C-
[29]	W, A	In, Lo, Mu	Act	TSW	C-
[32]	W	In	Act	TSW	HARUSDS
[79]	W	In	Act, Health	AGG	C-
[106]	W	In	Act	SSW	WISDM
[135]	W	In	Act	SSW	C-
[156]	W	Lo	Nav	AGG	C-
[159]	W	In	Health	SSW, AGG	WISDM
[137]	W	In	Health	TSW	C-
[3]	W	In	Act	SSW, AGG	C-
[82]	W	In	Act	TSW, AGG	SBRHAPTDS
[101]	W	Mu	Health	TSW	C+
[16]	W	In	Nav	SSW	C-
[114]	W	In, Lo, Mu	Health	SSW, AGG	C-
[64]	W	In, Mu	Act	TSW	C-
[115]	W, A	In, Am	Nav	AGG	C-
[171]	W	In, Lo	Nav	TSW, AGG	C-
[13]	W	In, Lo	Health	TSW	C-
[167]	W	In, Lo, Mu	Health	AGG	C-
[10]	W	In, Lo	Loc	SSW, AGG	C-
[14]	W	In	Health	TSW	C-
[165]	W	In	Health	TSW	C-
[108]	W	In	Act	TSW	UniMiB SHAR
[59]	W	In	Act	TSW	UniMiB SHAR
[72]	W	In	Act, Nav	SAM, TSW	SHL
[168]	W	In, Lo	Act, Nav	TSW, AGG	SHL
[119]	W	In	Act	TSW	MobiAct
[50]	W	In	Act	TSW, AGG	MobiAct
[24]	W	In, Lo	Act	TSW	WISDM, UCI-HAR, SHOAIB
[155]	W	In, Lo	Act	TSW, AGG	C-, SHOAIB

Cette thèse est accessible à l'adresse : https://theses.insa-lyon.fr/publication/2024ISAL0024/these.pdf
© [P. Lachat], [2024], INSA Lyon, tous droits réservés

[142]	W	In	Act	TSW	WISDM, Actitracker
[133]	W	In	Act	TSW	WISDM, Actitracker
[11]	W	In, Lo	Act	TSW, AGG	Extrasensory
[40]	W	In, Lo	Act	TSW, AGG	Extrasensory
[96]	W	In	Act	SSW	C-, RW, MS, SA
[27]	W	In, Lo	Act	TSW	Walking recognition
[68]	W	In, Lo	Act	TSW	Real-life HAR
[141]	W	In, Lo	Nav	SSW	SHL, TMD
[54]	W	In	Loc	TSW, AGG	HASC-2016
[116]	W	In, Lo	Act	TSW, AGG	Miao
[121]	A	Am, Mu	Act	AGG	ARAS
[81]	W	In	Health	AGG	C-
[136]	W	In, Lo, Bi	Health	SSW, AGG	C+
[109]	A	Mu	Loc	TSW	C-
[111]	W, A	Am	Health	TSW, AGG	C-
[166]	W	In, Lo	Loc	SSW, AGG	C-
[113]	W	In, Lo	Loc	AGG	C-

Table 2.1: References describing the usage of data mining algorithms on sensor data in order to infer personal information.

In the following section, we present the family of constraints that we consider in this thesis.

2.2 Discussion

We observe that, among all the constraints employed by the references studied above, the most pervasive ones are the usage of TSW and SSW (see Section 2.1.1). This comes from the necessity of processing continuous data points to discover personal information such as human activities. As a first step towards tackling the detection of IAISDs, we focus on those two types of sliding windows as constraints in this thesis. Furthermore, as presented in Section 2.1.4, the environment in which sensor are deployed defines the individual related to an inferred personal information.

In the references compiled in Table 2.1, datasets containing sensor data are used to demonstrate the feasibility of the proposed data mining process. To depict the environments considered in those references, we display in Table 2.2 the corresponding constraint. We observe that most references are divided between the *individual* and *area* environments. Only Cumin et al. [43, 44] explicitly consider that some inferred information (i.e., human activities) are specific to sub-area (i.e., rooms in a smart-home). Since both wearable and ambient sensors are equally used in the literature, we estimate that capturing this diversity of environments is important. In this thesis, we consider as environments: the individual, the area, and the sub-area.

2.3 Conclusion

As an introduction to our first contribution, we have observed in Table 2.1 that an increasing number of references inferring personal information from sensor data have been published in the last 10 years. To observe the diversity of approaches and heterogeneity of inferred personal information, we have proposed to classify those references with the following taxonomy: we consider the type of sensor deployment (i.e., on an individual or in an area such as a smart-home); the type of physical observations performed by sensors (i.e., inertial, location, ambient, multimedia, and biological data); the type of mined personal information (i.e., human activities,

Cette thèse est accessible à l'adresse : https://theses.insa-lyon.fr/publication/2024ISAL0024/these.pdf
© [P. Lachat], [2024], INSA Lyon, tous droits réservés

	Individual	Area	Sub-area
TSW	[15, 44, 33, 95, 37, 6, 29, 32, 137, 82, 101, 16, 64, 171, 13, 14, 165, 108, 59, 168, 119, 50, 24, 155, 142, 133, 11, 40, 27, 68, 54, 116, 111]	[140, 8, 83, 122, 86, 29, 109]	[44]
SSW	[43, 44, 106, 135, 159, 3, 114, 10, 72, 96, 141, 136, 166]	[44, 51, 4]	[43, 44]
AGG	[15, 93, 95, 157, 9, 79, 156, 159, 3, 82, 114, 115, 171, 167, 10, 168, 50, 155, 11, 40, 54, 116, 121, 81, 136, 111, 166, 113]	[55, 83, 122]	
SAM	[6, 72]	[55, 83, 122]	

Table 2.2: Distribution of the references in Table 2.1 by considering the constraint and the environment for which the inferred information is valid. Timestamp based Sliding Window (TSW), Sequence based Sliding Window (SSW), Aggregation (AGG), Sampling (SAM).

appliance usage, health, navigation, localization); and the type of constraints applied to sensor data (i.e., aggregation, sampling, and sliding windows). Finally, in this thesis, we will focus on the most common constraints Timestamp based Sliding Window (TSW) and Sequence based Sliding Window (SSW), and the environment corresponding to sensors deployed on an individual, in an area, or a sub-area. In the next chapter, we present the state of the art of inference detection system for profile databases, and position the work of this thesis according to the specificities of the references presented here.

Cette thèse est accessible à l'adresse : https://theses.insa-lyon.fr/publication/2024ISAL0024/these.pdf
© [P. Lachat], [2024], INSA Lyon, tous droits réservés

Chapter 3

State of the art: The inference problem in databases

As the concept of *inference attack* is generic, a variety of terms has been used in the scientific literature to denote such attacks. The usages we observe range from describing this concept as a problem (e.g., the inference problem, the data leakage problem, or the disclosure problem), as a privacy threat, an inference violation, etc. Harris-Kojetin et al. [80] define *disclosure* as the "[…] inappropriate attribution of information to a data subject, whether an individual or an organization". While the disclosed information varies depending on each setting, three main generic types of disclosure are identified in the literature [26, 91]: the *identity disclosure*, the *membership disclosure*, and the *attribute disclosure*. They are often analyzed according to the *information source* leveraged by the attacker (e.g., ML models [87] or databases [89]).

In the first part of this chapter, we position the problem of this thesis according to the categories used in the literature: the Inference Attack Involving Sensor Data (IAISD) is an attribute disclosure where the information source is a database containing sensor data. As presented by Woodall et al. [161], attackers exploits different inference strategies to disclose attributes from databases. An attacker performed the IAISD by exploiting the *External Dependency strategy*, since both information from the database (i.e., queried sensor data) and from an external source (i.e., the published data mining algorithm settings) are used. In the second part, we present the two families of solutions that tackle inference attacks in databases and described, using our taxonomy, the solutions in the state of the art related to the *Inference Control mechanisms*.

The outline of this chapter is the following: In Section 3.1, we present the terminology used to classify attributes leveraged to disclose individuals' information. Section 3.2 reviews the different types of disclosure studied in the literature. We illustrate each of them using examples of disclosure associated to different information sources. We put the focus on the inference strategies used in databases for the attribute disclosure in Section 3.3, and the two main research directions pursued to prevent this type of disclosure in databases. We introduce in Section 3.4 the taxonomy used to classify the state of the art according to our research questions. In Section 3.5, we present the limitations of those solutions w.r.t. to our research questions.

3.1 Type of attributes

According to Carvalho et al. [26], the collected attributes related to individuals can be classified using the following terminology:

- Identifiers: attributes which uniquely identify an individual, e.g., a social security number.

Cette thèse est accessible à l'adresse : https://theses.insa-lyon.fr/publication/2024ISAL0024/these.pdf
© [P. Lachat], [2024], INSA Lyon, tous droits réservés

- Quasi-Identifiers (QID): attributes which may, if combined, uniquely identify an individual, e.g., gender, birth date, ZIP code.
- Sensitives: attributes considered as confidential by individuals, laws, or regulations, e.g., religion, political orientation, etc.
- Non-Sensitives: all the other attributes, e.g., hashed email address or smartphone advertising ID [35].

According to the GDPR [69], the data controllers must "[…] implement appropriate technical and organisational measures, such as pseudonymisation, which are designed to implement data-protection principles, such as data minimisation, in an effective manner […]". Pseudonymizing (e.g., with k-anonymity [148]) or perturbing (e.g., with differential privacy [49]) individuals' data while preserving good utility is challenging. Although such processing removing or modifying identifiers attributes is essential, attackers are able to leverage available data to disclose individuals' personal information. Next, we present the different types of disclosure considered in the literature and we illustrate how attackers disclose data using different information sources.

3.2 Types of disclosure

In the scientific literature, the three main type of disclosures: *identity*, *membership*, and *attribute*. They are often studied by focusing on the information sources that attackers exploit. Some of those sources, such as *electronic devices* [151], *social networks* [126], *ML models* [87], and *relational databases* [89], contain personal data from which personal information may be inferred. In the following, we illustrate how attackers leverage those information sources to perform each of the three types of disclosure.

3.2.1 Identity disclosure

An identity disclosure occurs when an attacker is able to recognize that some data in the information source concerns an individual by matching QID attributes [26].

Electronic device

The wearable sensors and the sensors embedded in smartphones are often used to provide measurements about an individual. Those information can be exploited to disclose the identity of such an individual. For instance, Gohar et al. [74] propose a technique to re-identify peoples in an automated surveillance environment. The objective is to associate sensors data with a unique individual. Instead of relying on cameras to extract visual features, the authors leverage accelerometers and gyroscopes from both smartphone sensors and wearable sensors. Those measurements are used to analyze the gait of individuals', which is assumed to be unique. Using inertia data collected for 86 participants, Gohar et al. demonstrate that the walking signature of human gait enables a high accuracy of re-identification (i.e., 87%), using four distinct deep learning model. Since accelerometer and gyroscope sensors are often accessible on smartphone, even without the owner's authorization [98], an attacker may use such approach to disclose the identity of an individual using an publicly available dataset.

Social networks

The social graph data are valuable for diverse analysis. However, it can disclose the identity of individuals, thus breaching their privacy. Even when the identifiers attributes are removed

Cette thèse est accessible à l'adresse : https://theses.insa-lyon.fr/publication/2024ISAL0024/these.pdf
© [P. Lachat], [2024], INSA Lyon, tous droits réservés

from the release social graph data, an attacker can exploit the structure of the graph to disclose an individual's identity. Chen et al. [30] demonstrate how users can be re-identified from a pseudonymized social network graph periodically released. The authors assume that the attacker knows the identity of the targeted individual in the pre-release social network. The attacker's objective is to find which node in the released graph is the one related to this individual. The attack is performed by creating fake interconnected nodes (i.e., accounts), which are connected with the targeted individual's node. This pattern of connections is used as a fingerprint for this individual. Chen et al. describe that for each periodic release of the graph, the attacker retrieves first the set of fake accounts according to their structure of interconnection. Then, the attacker re-identifies the individual's node. They can improve the certainty by refining a previous re-identification upon a new release of the graph. The attacker can remove from the set of fake accounts of the previous release of the graph, a subset of account with a size equals to the number of node removed between these snapshots of the graph.

ML models

The availability of both large datasets and new tools (e.g., Torch, TenserFlow, and so on) has increased the usage of Machine Learning (ML) models [18]. To the best of our knowledge, the identity disclosure attacks that use ML models always follow the approach presented for Gohar et al. [74], i.e., identifying unique data records in a dataset. However, instead of having a model trained with inertial data, it can be trained with pictures [92] or other data. While for Gohar et al., the attacker have access to the sensor data directly to then train a model, here, the attacker has only access to the ML model. For instance, using a *model inversion attack*, one can leverage a pre-trained ML model to make it generate data similar to the training data, thus disclosing sensitive information. Khosravy et al. [92] show how such attack can be performed on an image classification model to re-identify an individual. The authors assume that the attacker has access to the black-box model which takes as input picture and classify them into a pre-registered individual. The attacker trains a new model which takes as input the identity of the black-box model and produces as output the corresponding pictures. The authors show that with a deep generative approach, they are able to reconstruct pictures with a rate of similarity of 96%.

Relational database

Relational database are the common approach to store and manage data. To illustrate the identity disclosure in this context, let us consider the example provided by Samarati et al. [138] and illustrated by Table 3.1. The patients' data in Table 3.1a have been de-identified by removing the identifier attributes, here the names and the Social Security Numbers (SSNs), to publish data without disclosing identities. However, some attributes such as the ZIP code, Day of Birth (DOB), race, sex, and the Marital Status (MS) may be released in another dataset, with the individuals' identity which can be linked with a patient. For instance, the ZIP, DOB, and sex present in Table 3.1a can be linked with the voter list, depicted by Table 3.1b, to identify a patient's name, address, and city. When some of the record is unique, it uniquely identifies the patient, e.g., pointed by ➤ in our example.

3.2.2 Membership disclosure

A membership disclosure occurs when an attacker is able to infer that the sensitive information regarding an individual is present or not in the information source [26].

Cette thèse est accessible à l'adresse : https://theses.insa-lyon.fr/publication/2024ISAL0024/these.pdf
© [P. Lachat], [2024], INSA Lyon, tous droits réservés

SSN	Name	Race	DOB	Sex	ZIP	MS	HP
		asian	09/27/64	female	94139	divorced	hypertension
		asian	04/18/64	male	94139	married	chest pain
		black	03/13/63	male	94138	married	hypertension
		black	09/13/64	female	94141	married	shortness of breath
		white	05/08/61	male	94138	single	obesity
	➤	white	09/15/61	female	94142	widow	shortness of breath

(a) Medical data released as anonymous.

Name	Address	City	ZIP	DOB	Sex	Party
…	…	…	…	…	…	…
Sue J. Carlson	900 Market St.	San Francisco	94142	9/15/61	female	democrat
…	…	…	…	…	…	…

(b) Voter list.

Table 3.1: Simplified re-identification example provided in [138]. The following column names stand for Security Social Number (SSN), Data of Birth (DOB), Marital Status (MS), and Health Problem (HP).

Electronic device

Besides the inertial sensors embedded in smartphone, the location service can be used to determine if an individual is present, or not, within an aggregate a set of locations. This disclosure can be sensitive by itself or a first step towards further disclosure targeting the same individual. Pyrgelis et al. [128] show that by using Principal Component Analysis (PCA), the attacker is able to capture how many times an individual has been in a location over time. Then, the PCA is combined with Logistic Regression (LR) classifier to perform membership disclosure. Besides this feature, the authors demonstrate that the volume of contributed data in aggregate location time-series, as well as the regularity and particularity of users' mobility patterns, enables improves this type of disclosure.

Social network

Zheleva et al. [169] identity the *group membership disclosure* as a threat where an attacker infer "whether a person belongs to a group relevant to the classification of a sensitive attribute". To the best of our knowledge, no attacks have been proposed in the literature for the membership disclosure in social network. Furthermore, while the membership disclosure is studied for social network data, the assumption is that they are always accessible via a ML model trained on them [110].

ML models

According to Carvalho et al. in a Federated Learning settings where several hospitals train a model for the COVID-19 diagnosis, an attacker can reveal which patient have been tested. Then, they can identify in which hospital the patient went, which leads to a discrimination risk, reinforced when the hospital is in a unsafe area (e.g., state or country) [47]. Another example is presented by Liu et al. [110]. They propose an attack which determines if the data of a specific individual have been used, or not, to train a ML model. The authors first generate synthetic data that have the same format than the dataset used to train the targeted model. Then, they classify the synthetic

Cette thèse est accessible à l'adresse : https://theses.insa-lyon.fr/publication/2024ISAL0024/these.pdf
© [P. Lachat], [2024], INSA Lyon, tous droits réservés

data according to the related prediction provided by the targeted model. For each prediction, the related data are clustered in two groups: *training set* for data that have a small variance and *testing set* for the other. A deep learning model is then trained with those data to mimic the prediction accuracy of the targeted model, until the output of both are indistinguishable. Finally, Liu et al. train the attacker model based on the prediction differences of the mimic model between the training and testing set. For a given record, the attacker collects the targeted model prediction and use the attacker model to predict if the record was part of the training set or testing set.

Relational Database

As presented by Woodall et al. [161], in relational databases the membership disclosure can occur via the exploitation of database constraints. Let us consider the following two relations: *MT(MissionID, Type)* and *SMD(Startship, MID, Destination)* [130]. The SMD relation describes missions. For instance, "Enterprise is on mission #101 to Rigel" corresponds to the tuple *(Enterprise, 101, Rigel)*. The MT relation determines the classification level of a mission. For instance, classifying the mission 101 has "top-secret" results in the tuple *(101, top-secret)*. All the sensitive value for the attribute MID refers to a sensitive or non-sensitive MissionID. If the database schema prevents deletion of a non-sensitive MissionID referred to by a sensitive MID tuple, the attacker can infer the existence of a sensitive MID by trying to delete a MissionID. Hence, insert, update and delete operations can be leveraged by attackers to exploits the system constraints and infer the presence of a sensitive value.

3.2.3 Attribute disclosure

An attribute disclosure occurs when an attacker is able to determine new information of an individual based on the data available in the information source [26]. The attacker may not precisely identify the record of a targeted individual, but could perform an attribute disclosure by leveraging the data related to the group in which the target belongs [88]. Hence, an attribute disclosure does not necessarily entail an identity disclosure.

Electronic device

The sensors embedded in smartphones can be leveraged to disclosure attribute such as an individual's transportation mode (e.g., still, run, bike, car, etc.). Sharma et al. [141] demonstrate that they are able to identify this information by using the accelerometer, gyroscope, and magnetometer data. The authors leverage multiple deep learning techniques (i.e., convolutional, LSTM, and fully connected blocks) to classify the transportation modes based on the sensors time-series data. They achieve an accuracy of 95% and 92% for the two considered datasets. While their objective is to provide context-aware assistance in an intelligent transportation system, an attacker may leverage the same approach for a nefarious purpose.

Social network

According to Piao et al. [126], in social networks, the attribute disclosure leverage three types of information:

- The *profile content*. For example, the gender, marital status, and relationship status of a Facebook user are inferred, with an accuracy of more than 90%, from their publicly available profile data (e.g., name, birthday, …) [34].

Cette thèse est accessible à l'adresse : https://theses.insa-lyon.fr/publication/2024ISAL0024/these.pdf
© [P. Lachat], [2024], INSA Lyon, tous droits réservés

- The *social links*. The hidden attributes of a user's profile (i.e., college, employment, and location) are inferred by [160]: using the publicly available attributes of other profiles; computing the community of profiles having a directed relationships (i.e., followers, following, and friends) which this user using the Louvain method [21]; leveraging their own cost function to compute the attribute vector of the initial user according to the profiles in the same circle.
- The *user behavior*. E.g., by matching the user profiles having similar musical interest, via the shared message or appreciated content, the age of a profile (i.e., the hidden information) is inferred from the age of the related users [28].

ML models

According to Hu et al. [87], ML models are exposed to different attribute disclosures. Frederikson et al. [62] present a model inversion attack which uses a model output to disclose sensitive features used for training. However, whereas Khosravy et al. [92] use this type of attack to re-identify an individual, the objective is here to disclose answers to a sensitive question. Indeed, to demonstrate their attack, they leverage the *FiveThirtyEight survey* [84] which contains answers to questions such as "Do you ever smoke cigarettes?", "Have you ever cheated on your significant other?", and so on. The authors assume that the attacker has access to a decision tree trained on this dataset to predict the sensitive attribute. This attribute is here assumed to be the answer "Yes" to the question about infidelity. Frederikson et al. show that by having access to the trained model and the features of an individual, the attacker is able to identify the individuals that have answered "Yes" to this question with a perfect accuracy.

Relational database

In the example of Samarati et al. [138] the attacker is able to know wether the Health Problem (HP) is related to the patient named Sue J. Carlson. An identity disclosure is here required to perform the attribute disclosure. However, in other situations, the identity disclosure is not necessary. Let us consider a database having the following relation *Employee(Name, Rank, Salary)*, a security rule preventing users to obtain the salary of an employee, and a Functional Dependency (FD) between the attribute rank and salary. To disclose the salary of an employee, an attacker exploits the inference channel (i.e., the FD) by first querying the name and rank of the targeted employee, and then querying all the rank and salary [150].

While the attribute disclosure provides a knowledge of an individual's information, it is possible to infer a sensitive attribute with high confidence from the released data. For instance, publicly available data with high correlation between income and purchase price of home can be leveraged to infer an individual's salary [134]. As described by Iyengar [88], some works consider that an "[…] attribute disclosure occurs when something about an individual is learnt from the released data". To illustrate how an attacker can increase their confidence w.r.t. an attribute, let us consider the example proposed by Li et al. [107]. A company wants to share its payment information, depicted by Table 3.2a, with sociology scientists. The salary attribute is considered as sensitive, and the age and ZIP are QIDs. Assuming that an attacker knows the age (e.g., 17) and ZIP (e.g., 12k) of an individual named Andy. According to Table 3.2a, the attacker deduces Andy's salary (i.e., 1000), since the first tuple matches certainly their information. Hence, to protect the individuals' privacy, the company generalizes the data before the release, by grouping QIDs as illustrated by Table 3.2b, so that it renders indistinguishable tuples w.r.t. the age and ZIP. The attacker cannot with certainty determine if Andy's is 1000, 1010, 1020, or 50000. However, the attacker can determine with a high confidence the limited interval in which Andy's salary

Cette thèse est accessible à l'adresse : https://theses.insa-lyon.fr/publication/2024ISAL0024/these.pdf
© [P. Lachat], [2024], INSA Lyon, tous droits réservés

Age	ZIP	Salary
17	12k	1000
19	13k	1010
20	14k	1020
24	16k	50000
29	21k	16000
34	24k	24000
39	36k	33000
45	39k	31000

(a) Original data.

Group	Age	ZIP	Salary
1	[17,24]	[12k,16k]	1000
1	[17,24]	[12k,16k]	1010
1	[17,24]	[12k,16k]	1020
1	[17,24]	[12k,16k]	50000
2	[29,34]	[21k,24k]	16000
2	[29,34]	[21k,24k]	24000
3	[39,45]	[36k,39k]	33000
3	[39,45]	[36k,39k]	31000

(b) Released data.

Table 3.2: Probabilistic attribute disclosure example proposed by Li et al. [107].

Type of disclosure	Information sources			
	Electronic devices	Social networks	ML models	Relational DB
Identity	Inertia sensors & Data mining techniques [74]	Graph structure analysis [30]	Camera sensors & Model inversion attack [92]	Record linkage technique [138]
Membership	Location service & PCA & LR [128]	—	Mimic model training [110]	Contraints violation [161]
Attribute	Inertia sensors & Data mining techniques [141]	Graph structure analysis [126]	Yes/No question & Model inversion attack [62]	Statistical inference [107]

Table 3.3: Example of techniques used to perform disclosure in different source of information. DB stands for database and ML for machine learning.

is. By knowing only Andy's QIDs, the attacker presumes that each record in the first group of Table 3.2b has the same chance to be Andy's tuple. Hence, they can conclude that Andy's salary is in the range [1000, 1020] with 75% probability, even if there is only a probability of 25% to infer the real salary. The attacker is able to infer with a high confidence that Andy's salary is approximately equal to 1000.

As summarized in Table 3.3, diverse approaches are utilized to disclose identity, membership, or attribute of an individual. As stated in Section 1.2, in this thesis we focus on the problem of detecting the inference of personal information from a sensor database. Since the attacker aims to obtain new information about an individual, we consider that an Inference Attack Involving Sensor Data (IAISD) is an attribute disclosure. Furthermore, we consider that the attacker uses as a information source the sensor database $\mathtt{DB}_{\mathtt{sen}}$. In the following section, we focus on the inference attacks that disclose attribute in databases.

3.3 Inference attacks in databases

In the following sections, we present the strategies used by attackers to query a database in order to infer non-authorized attributes, and the two main families of approaches that aim to control inferences performed by those attackers.

Cette thèse est accessible à l'adresse : https://theses.insa-lyon.fr/publication/2024ISAL0024/these.pdf
© [P. Lachat], [2024], INSA Lyon, tous droits réservés

3.3.1 Inference strategies in databases

To the best of our knowledge, Woodall et al. [161] is the only work that comprehensively reviews inference attacks strategies in databases. The authors identified the following strategies related to attribute disclosure:

(i) *Split Query strategy*: This strategy is used in a setting where an access control mechanism prevents attackers from querying two or more specific attributes together in the same query, while authorizing querying these attributes separately. The goal of an attacker is thus to bypass the access rule. To do so, they issue a query per targeted attribute and links together the attribute values coming from the same instance.

(ii) *Colluding Users strategy*: With this strategy, the attack follows the same process as the split query strategy, however the queries are issued by at least two users who then recombine their answers to infer data that they individually do not have complete access to.

(iii) *External Information strategy*: With this strategy, the attacker does not obtain all the information from the database. Instead, they leverage knowledge outside to obtain sensitive information from the database. For instance, Chen et al. [31] depict a context where the relation between the size of an aircraft and the landing track properties is assumed to be a domain expert knowledge. An attacker can use this knowledge as a dependency to infer sensitive information about the airport in which aircrafts land.

(iv) *Secondary Path strategy*: In relational databases, a secondary path is a way to join attributes of different tables which are not intended by the database designers (e.g., by not using the primary/foreign keys). With this strategy, like with the split query one, the attacker aims to bypass an access control rule preventing them from querying two or more attributes together. However, the difference is that a single query using the secondary path is issued here. For instance, selecting the primary key between the *Employee* and *EmpSalary* tables maybe denied to prevent attackers to select the salary and name of an employee. An attacker can use as a secondary path the employees' phone number, which is a unique attribute, to join the two tables and select the name and salary of an employee.

(v) *Hybrid strategy*: This last strategy is leveraged by chaining together multiple inference strategies. The attacker uses the output of a first used strategy as input for the second, and so on, until the targeted sensitive information is inferred.

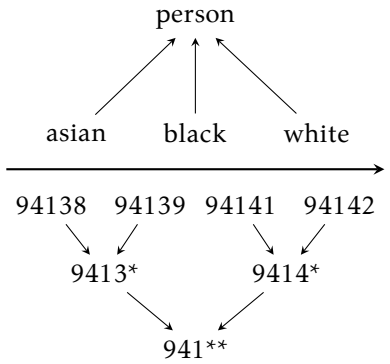
In the motivating example described in Section 1.1, the attackers leverage the Hybrid strategy by combining two External Information strategies corresponding to the two described inference channels. The PAL obtained from sensor data, thanks to the first external information [15], is used with the Age and Sex to infer the CVD, thanks to the second external information [100]. We classify them as External Information strategies, since the attacker leverages knowledge outside of the database (i.e., the works proposed by Kubota et al. [100] and Banos et al. [15]) to disclose the CVD. Hence, the IAISD is a type of inference attack where attackers use the External Information strategy in order to disclose an individual's attribute. In the remainder of this thesis, *inference* is used as a synonyme of *an attribute disclosure*. Next, we illustrate how proposed solutions prevent inference attacks on databases.

3.3.2 Approaches to prevent inferences in databases

To prevent the inference of sensitive personal information it is necessary to *control inferences*. The two protection approaches associated to databases are to either rely on *query restriction* or

Cette thèse est accessible à l'adresse : https://theses.insa-lyon.fr/publication/2024ISAL0024/these.pdf
© [P. Lachat], [2024], INSA Lyon, tous droits réservés

Race	ZIP
asian	94139
asian	94139
black	94138
black	94141
white	94138
white	94142



Race	ZIP
person	9413*
person	9413*
person	9413*
person	9414*
person	9413*
person	9414*

Table 3.4: Example of generalization inspired by [138].

ID	Ori.	Dest.	Time
1001	s1	s2	13:02
1002	s1	s2	13:15
1003	s1	s4	14:28
1004	s2	s1	14:55
1005	s2	s1	15:21
1006	s1	s2	15:30

(a) Credit card payment data for subway fares between an origin (Ori.) and destination (Dest.) subway stations.

Route	Freq.	Selection probability		
		$\varepsilon = 0.01$	$\varepsilon = 0.1$	$\varepsilon = 1$
s1 → s2	3	0.335	0.350	0.507
s2 → s1	2	0.333	0.333	0.307
s1 → s4	1	0.332	0.317	0.186

(b) Example of exponential mechanism with the dataset in Table 3.5a.

Table 3.5: Example of a dataset perturbation presented by [94].

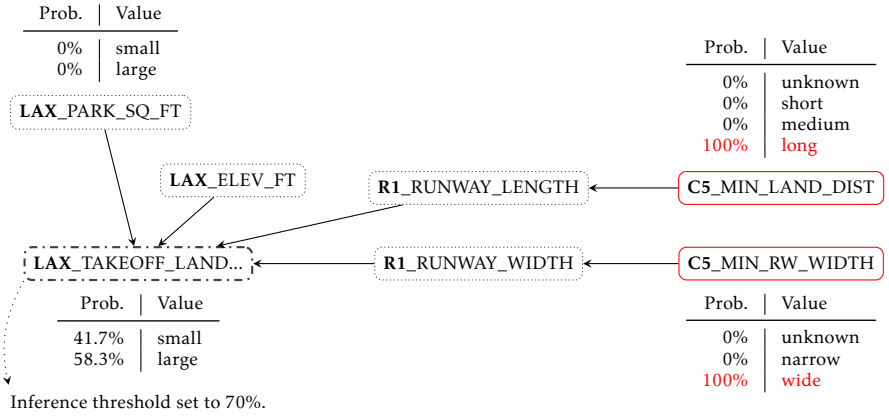
query perturbation [45]. The perturbation approach aims to modify the data on which the query is performed or the result of the query. The restriction approach determines if the query result must be returned or not. The two related families of solutions we have identified are the Access Control (AC) mechanisms and the Privacy-Preserving Techniques (PPT), respectively.

The purpose of PPT [26] is to guarantee privacy while preserving the utility of shared data (e.g., for prediction tasks). The purpose of those solutions is to undergo alterations on shared data (e.g., with k-anonymity [148] or differential privacy [49]) while preserving utility the shared data. Those approaches aim to release useful data, while preventing re-identification of individuals whose data have been used to compute the statistics.

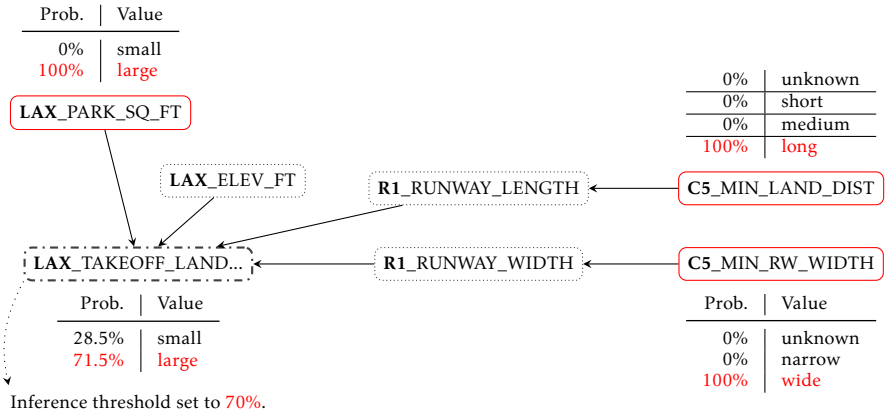
In Section 3.2.1, we have presented the example proposed by Samarati et al. [138], where an individual is re-identified by linking their anonymized data thanks to the use of a voter list. To address this issue, Samarati et al. propose to use generalization according to the data semantic. For instance, the proposed hierarchy to generalize the Marital Status attribute has three levels: (i) married, divorced, widow, single, or (ii) been married, never married, or (iii) not released. An attribute with the *married* value can be generalized first to *been married* and then to *not released*, if required. Hence, to prevent the identity disclosure relying on QIDs such as the race and ZIP code, those attributes can be generalized by one level, according to the hierarchies depicted in Table 3.4. The resulting data preserve utility, since one can still compute statistics about the area where each patient is located, for instance.

Let us consider another example provided by Kim et al. [94] and illustrated by Table 3.5. An analyst aims to determines the most frequent itinerary between subway stations. To do so, they

Cette thèse est accessible à l'adresse : https://theses.insa-lyon.fr/publication/2024ISAL0024/these.pdf
© [P. Lachat], [2024], INSA Lyon, tous droits réservés



(a) Bayesian network before querying LAX_PARK_SQ_FT.



(b) Bayesian network after querying LAX_PARK_SQ_FT.

Figure 3.1: Example of a probabilistic attribute disclosure inspired from Chen et al. [31]. **LAX** is an airport, **R1** a runway, and **C5** an aircraft. The sensitive node has a dash-dotted outline. The known nodes have a red and solid outline. The unknown nodes have a dotted outline.

issue a query to the data controller collecting the credit card payment data for subway fares, depicted by Table 3.5a. Here, an *exponential mechanism* is used to determine which answer is to be returned to the analyst. This type of mechanism is used for non-numerical query output. It uses a function to associate to each possible answer a score. The score is set so that it is higher for the true output of the query. Then, the mechanism uses the scores as probabilities to randomly choose the query answer. Let us consider Table 3.5b which contains the different routes and their frequencies, according to Table 3.5a. For instance, the route s1 → s2 is used three times. The selection probability corresponds to the score associated by the mechanism, to each possible answer to the analyst's query. The parameter ε corresponds to the *privacy budget*. A small (resp. higher) ϵ value leads to a strong (resp. weaker) privacy guarantee, but it incorporates more (resp. less) noise in the result. The exponential mechanism is parametrized with ε to adjust the score associated to each possible query output. For example, considering $\varepsilon = 0.001$, we observe that each possible route has approximately the same probability to be selected. When $\varepsilon = 1$, a higher probability is associated to the true output of the query (i.e., s1 → s2).

Cette thèse est accessible à l'adresse : https://theses.insa-lyon.fr/publication/2024ISAL0024/these.pdf
© [P. Lachat], [2024], INSA Lyon, tous droits réservés

On the other hand, AC mechanisms [131] are used to secure databases access according to policies, roles, attributes, etc. This ensures that only authorized users can directly access data. However, the standard AC mechanisms are not capable of detecting the inference of sensitive information via the access to non-sensitive data [57]. We refer to the AC mechanisms proposed to tackle this specific problem as Inference Control (IC) mechanisms.

In the following, we develop an example proposed by Chen et al. [31], illustrated by Figure 3.1. The authors propose to use Bayesian networks to represent the probabilistic dependencies between the attributes of a database, as well as each user queried data. Here, the graph makes references to three distinct instances: *LAX*, *R1*, and *C5* which correspond to an airport, a runway, and an aircraft, respectively. Each illustrated attribute is prefixed with its corresponding instance. The nodes of the graph describe the a priori value distribution of the attributes, at the moment the Bayesian network is learnt. In this illustration, the sensitive attribute is TAKEOFF_LANDING_CAPACITY (TLC) (i.e., the dash-dotted node). All the other attributes are considered as non-sensitive. The authors set an inference threshold equal to 70% for the TLC attribute. This indicate that an inference is detected when an attacker knows one of the value of TCL with a probability equal or greater than 70%. Assuming that the attacker knows (e.g., via previous queries) that *C5* is able to land on *R1* from *LAX*, and that they have queried C5_MIN_LAND_DIST (= *long*) and C5_MIN_RUNWAY_WIDTH (= *wide*) (i.e., both have red and solid outlines), then the attacker is able to infer the value of TLC with a confidence of 58.3% in Figure 3.1a. If they succeed to query LAX_PARKING_SQ_FT (= *large*), they would be able to infer that TCL is equal to *large* with a percentage of confidence of 71.5% in Figure 3.1b. Since it is greater than the defined inference threshold, an inference attack is detected for this last query and the authors propose to deny answering it.

A way of classifying the solutions related to the PPTs and the AC mechanisms is to consider the time at which they protect the database.

Time of the detection

According to Frigerio et al. [63], the approaches that prevent inferences can be classified as either *non-interactive* (i.e., by-design) or *interactive* (i.e., query-time) privacy technique/mechanism.

Detection by-design In a non-interactive setting, the PPT purpose is to sanitize the data before the database use, i.e., before a user queries the database.

In this setting, the IC mechanisms function is to remove inference channels during database design [57]. Then, users are able to issue infinite queries to the database without disclosing sensitive information. For instance, Qian et al. [129] describe a tool called *DISSECT* to assist with the discovery of inference channels, by considering only the information stored in a Multilevel Relational Database (MRD). The MRDs store data that have different classification levels and users are able to access data of a specific level. This solution searches existing sequences of foreign key relationships that enable joining the same tables. If for two tables, one or multiple sequences necessitate lower access levels than planned by the database designer, the related relationships may be eliminated by modifying the database schema.

Yet, Su et al. [146] show that completely removing the inference channels in an MRD is an NP-Complete problem. Moreover, the schema of databases is not always available and often not modifiable. The difficulty of this approach is thus to preserve the data utility or data availability without knowing which data will be selected.

Detection at query-time In the interactive setting, an entity (e.g., a middleware acting as a control point) is placed between the database and the users to analyze issued queries. Since the

Cette thèse est accessible à l'adresse : https://theses.insa-lyon.fr/publication/2024ISAL0024/these.pdf
© [P. Lachat], [2024], INSA Lyon, tous droits réservés

result of multiple queries can be combined by attackers (see Section 3.3.1) the entity performs the analysis by considering the information queried by users in the past. It relies on a history to log each user's query result. Moreover, those solutions represent the inference channels that can be leveraged by attackers. They are used to determine how a user's new query result, according to their history, must be restricted or perturbed.

The IC mechanisms goal is to detect when a query enables a user to disclose sensitive information, and then to either deny or modify the query response (i.e., by refusing or lying [20]). A special case corresponds to the detection of inference attacks at query-time, while considering updates in the accessed data. To the best of our knowledge, only Toland et al. [150] have considered this problem. In their solution, they represent FDs using Horn clauses. When an update occurs in the relational database, it is also applied on the history keeping track of each user's previously obtained data. The update-time approaches is able to reason on up-to-date information, without assuming that accessed database is static. The main drawbacks of this setting is that the computation time for data modification (PPT) or for the detection (IC mechanisms) impacts the query answer time. Similarly, both types of approaches have to keep track of the information previously queried by a user, and to consider it when processing a new query.

Discussion

In this thesis, we aim to detect IAISDs, i.e., non-sensitive personal information inferred from a sensor database (see Section 1.2). The prevention part is out of our scope. We focus on the detection part only.

The *query modification* approach associated to the PPTs is not suitable, since they do not detect when an inference occurs, but rather modify the query output result [49, 148] by default. On the other hand, according to Danezis et al. [45], the *query restriction* is the right approach when the users require query answers to be exact. In the scientific literature, sensor databases are queried to obtain the value of exact data points [17] or to obtain statistics computed over data points [117, 78, 127]. In personal databases, the inference channels usually correspond to dependencies between attributes (e.g., Functional Dependency (FD)). IC mechanisms detect inferences attacks by verifying if a user's issued query, combined with their history, enable them to exploit a channel. Those solutions always detect when an inference before preventing it.

Moreover, due to the focus of this thesis, the solutions with a design-time approach are not suitable, since the inferred personal data is non-sensitive. Removing the inference channel obtained via the External Information strategy would reduce the data availability of the sensor database. On the other hand, the query-time approach is commonly used by several works [89] to reason on the data queried by users only.

Based on those observations, in the rest of this chapter, we focus on solutions in the family of query-time IC mechanisms to describe how the detection of inferences is performed. In the following, we introduce our taxonomy and classify the state of the art.

3.4 Taxonomy and classification of the inference detection system

For each work proposing a query-time IC mechanism, we focus on its Inference Detection System (InfDS). To classify those systems, we propose a taxonomy guided by our research questions presented in Section 1.3. Then, we illustrate each value associated to each facet of the taxonomy with a representative InfDS. Table 3.7 synthesizes how each work is positioned. Since this chapter focuses on the problem of inferences in database, the RQ4 related to the datasets is not considered here, but in Chapter 6. In the following, we present each facet and describe the associated works.

Cette thèse est accessible à l'adresse : https://theses.insa-lyon.fr/publication/2024ISAL0024/these.pdf
© [P. Lachat], [2024], INSA Lyon, tous droits réservés

Information captured as user's knowledge

In order to model the knowledge acquired by users querying databases RQ2, we have to consider what type of data are stored in the databases and what types of query result are returned during the detection phase.

The most common type of data considered in the literature are profile data (e.g., salary, age, name, etc.) usually stored in relational databases. In those works, the query result is the *exact static profile data* selected. Those solutions also assume that the protected database is not updated. Once delivered to a user, the profile data is always assumed to be static (i.e., up-to-date) for the selected instance. For example, Pappachan et al. [125] consider as a use case a profile database with two tables describing employees (e.g., ZIP code, role, work hours) and wages (e.g., department name, salary) information. When a query selects attributes related to an instance (e.g., the role and name of an employee), the modeled knowledge is the exact static data returned as the query result (e.g., *Role* = *Student* and *EName* = *Alice Land*).

To the best of our knowledge, only two solutions [150, 163] explicitly consider updates in a relation database and how it impacts the detection of inference attacks. For those works, the query result is the *exact non-static profile data*. Some delivered profile data may become outdated after a database update. The attacker may not be able to use correctly this knowledge at some point in time (i.e., non-static). For instance, Toland et al. [150] illustrate their problem with a profile database containing employees' data (e.g., name, rank, salary, department). Upon issuing a query (e.g., the rank and salary of an employee), the result contains the exact selected data (e.g., *Rank* = *Clerk* and *Salary* = *38,000* for the employee named *John*). When an update occurs, the authors represent that the obtained data have been updated (e.g., *Clerk* and *38,000* are updated to *Manager* and *45,000* after *John* is promoted). The modeled knowledge is exact data that can be updated according to a new state of the database. For instance, Toland et al. propose to stamp the entry of the history containing the knowledge *Rank* = *Clerk* and *Salary* = *38,000* with the updated values (i.e., Manager and 45,000). The stamped values are considered when the user issues a new query.

We observe that a few solutions consider profile data stored in a logical information system. Users interact with the system via binary queries (i.e., yes/no queries). In a table containing the name of employees, an example of a binary query would be to search if the salary of a specific employee is greater than a given value. The query result would not be the exact salary, but a boolean value. In those works, the database is also considered as static. Consequently, the query result is a *static boolean profile fact*. As an example, Guarnieri et al. [77] consider a system with the following tables containing the names of (i) all the patients (ii) the patients that are smoking (iii) the patients which are the father of other patients (iv) the patients which is the mother of other patients. The result of a query such as *smokes(Carl)* is the value *true*, or yes, since the patient named Carl is within the smoking table. The knowledge is modeled as a static boolean value related to a profile data, e.g., *smokes(Carl)* = *true*.

To the best of our knowledge, only a single work proposed by Noury et al. [123] considers sensor data stored as time series. The query result is here the *exact sensor data points* queried for a specific time interval. However, in their context, an inference attack occurs when at least two conflicting access rules lead to some temporal intersection that enable attackers to infer values that must be protected by the access control mechanism. Consequently, they do not consider inference attacks exploiting the sensor data. We explain this lack of works related to inference problem in sensor database by the focus made on PPT [26] to protect data via modification applied by-design, and the ongoing research focusing on efficient querying of sensor data [162].

In this thesis, we assume that users issue open queries selecting sensor data. As presented in RQ2, we capture *information about sensor data points*, such as the type of selected data, how

Cette thèse est accessible à l'adresse : https://theses.insa-lyon.fr/publication/2024ISAL0024/these.pdf
© [P. Lachat], [2024], INSA Lyon, tous droits réservés

any data points are selected, and so on.

RQ2 – Facet **Data**:
(i) Exact static profile data (ii) Exact updatable profile data (iii) Static boolean profile fact (iv) Exact sensor data points (v) Information about sensor data points.

Type of dependency captured as inference channel

As stated in RQ1, in order to model the inference channel, we have to consider the condition of disclosure of personal information. In the state of the art, the solutions consider different types of dependencies that are exploited by an attacker. The three main kinds we have identified are *Functional Dependency (FD)* [150], *Probabilistic dependencies* [31], and *Linkage dependencies* [103].

In those works, FD represents a logical relationship between a set of one or multiple attributes and a single attribute (e.g., $\{X, Y, \ldots\} \rightarrow Z$). Usually, the attributes on the left side of the dependency (e.g., $\{X, Y, \ldots\}$) are considered as non-sensitive attributes, whereas the right side attribute (e.g., Z) is considered as a sensitive attribute. Since the sensitivity of an attribute depends on the stored data, some dependencies may have sensitive attributes in the left side and a non-sensitive attribute on the right side. The relationship models that, for the same instance, knowing the value of the set of attributes $\{X, Y, \ldots\}$ enables inferring the value of Z. For instance, as illustrated in Section 3.2.3, the following relation *Employee(Name, Rank, Salary)* with a FD between the rank and salary attributes is considered by Toland et al. [150] as the dependency that an attacker leverages to infer the salary of an employee.

A probabilistic dependency represents a probabilistic relationship between the value of one or multiple attributes and the value of a single attribute (e.g., $P(Z = a \mid X = b, Y = c) = 80\%$). This relationship models that, for the same instance, knowing that the value of X is b and Y is c enables inferring that the value of Z is a with a certainty of 80%. An illustration of such dependency is provided in Section 3.3.2. Chen et al. [31] consider the probabilistic relationship between data in the snapshot of a profile database as a graph constituting dependencies between attributes. The dependencies leading to sensitive attributes are captured as inference channels, since an attacker aims to exploit them.

The linkage dependency represents a relationship between two attributes in two instances in distinct databases. The two instances represent the same real world entity. This dependency models that an attacker can access the information from different databases. For instance, Lachat et al. [103] illustrate an example of two databases related to an online DVD store and a fictitious food company, respectively. They both contain the profile data of their customers. In case the same individual is a customer of both services, an attacker can exploit this link to obtain data such as postal code or contact name from different databases. In this thesis, we assume that the attacker exploits a single sensor database. While the linkage dependencies are out of scope of this thesis, they are combined with other dependencies (i.e., functional or probabilistic) to perform an inference attack.

As presented in RQ1, in this thesis, we aim to capture *constrained dependencies*, i.e., inference channels defining the expected type of selected sensor data, the specific quantity of data points to query and/or the specific time for which they are generated, and so on. Thus, this representation is depends on the requirements of RQ2.

RQ1 & RQ2 – Facet **Dependency**:
(i) Functional dependencies (ii) Probabilistic dependencies (iii) Linkage dependencies.

Cette thèse est accessible à l'adresse : https://theses.insa-lyon.fr/publication/2024ISAL0024/these.pdf
© [P. Lachat], [2024], INSA Lyon, tous droits réservés

Dependency representation level

To model the condition of disclosure associated to an inference channel, RQ1 implies that we consider if dependencies are represented between attributes (i.e., *schema level*) or between values of instances (i.e., *data level*). According to RQ2, this level of representation impacts how the information obtained by users must be modeled.

At the *schema level*, the dependencies are represented between attributes, without considering the exact value stored in the database. Jebali et al. [90] leverage a use case with an hospital database schema. It contains three relations *Patients(Patient_id, Diagnosis_id, Admission_date, Patient_details)*, *Drugs(Drug_id, Drug_code, Drug_name, Drug_cost)*, and *Consumption(Patient_id, Drug_id)*. They model the following dependencies, among other: FD_3: *Patient_id* $\rightarrow$ *Patient_details* in *Patients*; FD_6: *Drug_id* $\rightarrow$ *Drug_code* in *Drugs*; FD_9: *Patient_id*, *Drug_id* $\rightarrow$ *Patient_id* and FD_{10}: *Patient_id*, *Drug_id* $\rightarrow$ *Drug_id* in *Consumption*. When a user issues the following queries: $Q_1 = \{Patient_id, Drug_id\}$, $Q_2 = \{Patient_id, Patient_details\}$ and $Q_3 = \{Drug_id, Drug_code\}$, based on the queried attributes, the attacker is able to exploit those dependencies $\{FD_9, FD_3, FD_{10}, FD_6\}$, which then enables them to infer *Patient_details* and *Drug_code*, which violates the defined security rule. All this reasoning never considers the exact values of the queried attributes.

At the *data level*, the dependency considers the exact values stored in the database. Most solutions relying on the data level first represent dependencies at the schema level before instantiating them w.r.t. the stored data. A good example to illustrate this level of representation is depicted by Figure 3.1 in Section 3.3.2. Chen et al. [31] model first the dependencies between attributes (e.g., $PARK_SQ_FT, ELEV_FT, RUNWAY_LENGTH, RUNWAY_WIDTH \rightarrow TAKEOFF_LANDING$). Then, they are modeled between the instances of the database (e.g., the **LAX**, **R1**, and **C5** instances). When an attacker queries the attribute of a specific instance, based on the value of this attribute (e.g., *LAX_PARK_SQ_FT = large*), the attacker will be able to infer *LAX_TAKEOFF_LANDING*. All this reasoning considers the exact values of the queried attributes. Some works implicitly consider the representation of dependencies directly at the data level. Indeed, Staddon et al. [144] propose to represent dependencies between any information that can be queried from the database (i.e., fact, attribute, value, or relation). Hence, according to the domain of the stored information, an expert can directly represent a dependency between an instance's attributes values, without first modeling them at schema level.

In this thesis, we model constrained dependencies RQ1 to reason on information obtained about sensor data points RQ2, as presented for the two first facets. The representation level of dependencies is neither at the schema level, since we need information more precise than the name of selected attributes; nor at the data level, since it isn't the exact value of a data point which is important, but rather the metadata about queried information (e.g., the selected physical observations or interval during which data points are generated, the quantity of data points in the query result, and so on) enables one to satisfy the constraints of dependencies associated to data mining processes (see Section 2.1.4). Hence, we represent the dependencies at the *metadata level*.

> RQ1 & RQ2 – Facet **Level**: (i) Schema (ii) Data (iii) Metadata.

Inference channel model framework

According to the type of dependency and the representation level, the solutions leverage different types of model to represent and structure all inference channels considered by an IC mechanism. Hence, according to the RQ1 we have to consider those different approaches. The two main types of model we have identified are the *graph-based model* and the *set-based model*. To illustrate the

Cette thèse est accessible à l'adresse : https://theses.insa-lyon.fr/publication/2024ISAL0024/these.pdf
© [P. Lachat], [2024], INSA Lyon, tous droits réservés

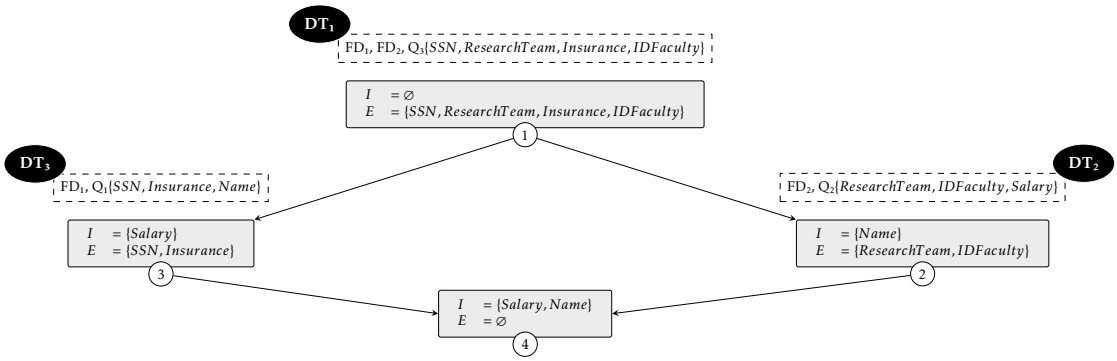


Figure 3.2: Example of queries transaction generated with a lattice [139].

diversity of approaches that exists in each type, we will describe the framework utilized by each work in the state of the work.

Graph-based model

We observe that this framework is leveraged by solutions which model probabilistic dependencies. This choice is explained by the need of representing conditional probabilities between attributes. The two models leveraged are Bayesian networks [31, 103, 77] and Petri nets [163].

As previously illustrated with Chen et al. [31]'s work, the Bayesian network enables the representation of dependencies at the data level. Each node corresponds to an instance's attribute and the distributions of probability, as well as the edges, are either automatically computed from a database snapshot, or manually defined by a domain expert. In the context of relational databases, the automatic computation of such models relies on the Probabilistic Relational Model [71]. While Guarnieri et al. [77] represent facts about individuals (e.g., a patient smokes or doesn't smokes), since they associate probabilities to those fact, they also leverage Bayesian networks to represent dependencies between profile data.

On the other hand, Yaseen et al. [163] use Petri nets to model the dependencies between attributes of a personal database. Similarly to the Bayesian network, a node of the Petri net corresponds to attributes in a personal database. The edges are defined manually by an expert. With Toland et al., the authors are the second one to consider updates in the database. They determine that the knowledge obtained by an attacker on the node of the Petri net changes when the values of the parent nodes reach a specific constraint (e.g., the value is greater than a threshold). Those conditions are encoded by the edge. The probabilities are also defined manually and correspond to the certainty that an attacker has on a node after having queried other nodes or changed an attribute value.

Set-based model

The latter model type structures dependencies as a set of non-connected elements describing each an inference channel. We have identified three groups of solutions based on the representation of (i) query transactions (ii) logical rules, and (iii) other approaches.

In the first group of works, the authors consider the dependencies that the attackers may leverage and generate all the queries which enable their exploitation. Hence, Jebali et al. [90], Sellami et al. [139], and Biskup et al. [20] represent inference channels as a set of query transactions where each transaction leads to some sensitive value. For instance, Sellami et al. rely on the Formal

Cette thèse est accessible à l'adresse : https://theses.insa-lyon.fr/publication/2024ISAL0024/these.pdf
© [P. Lachat], [2024], INSA Lyon, tous droits réservés

Concept Analysis (FCA) [67] framework to compute the transactions, instead of relying on an external entity. A lattice is built by iteratively considering all FDs leading directly, or indirectly, to an attribute in the access control rule. We consider as an access control rule the fact that the *salary* and the *name* of an instance cannot be known by a user. The two following FDs are modeled using FCA: (i) The first dependency (i.e., FD_1) leads to the name: $IDFaculty, ResearchTeam \rightarrow Name$. (ii) The second (i.e., FD_2) leads to the salary: $SSN, Insurance \rightarrow Salary$. The lattice depicted in Figure 3.2 contains four nodes. Each node describes some of the attribute of the control rule, denoted I and the queryable attributes, denoted by E. The node ① describes that, without knowing any attributes of the rule, if a user issues the query Q_3, they can then leverage the dependencies FD_1 and FD_2 to infer the salary and the name. This corresponds to a first disclosure transaction, denoted DT_1. The node ③ shows that, if the SSN and Insurance are available and the attacker's intent to infer the salary, then issuing Q_1 and exploiting FD_1 creates a new disclosure transaction DT_3. The node ② leads to DT_2, assuming the attacker focus on the name and issues Q_2. All the parents of the node ④ describe a transaction disclosure of the set I. While the two first works represent open queries (i.e., the standard SQL-like queries), Biskup et al. represent transactions of closed queries. Biskup et al. and Jebali et al. assume that an external entity provides this model to the detection system.

The second group of works represent dependencies as rules used to determine what knowledge attackers may obtain according to the queried information. Toland et al. [150], Pappachan et al. [125], and Biskup et al. [19] model inference channels as formula such as Horn clauses, first-order formula, or conjonction of logical facts (e.g., is the patient *Carl* smoking?), respectively.

Lastly, two unrelated approaches are used to model dependencies. Staddon et al. [144] consider dependencies as a set of objects in the database (e.g., attributes, values, relationships). The authors represent each inference channel as a set of cryptographic tokens, where each token is associated to an object and generated, and all the tokens associated to a channel are generated in accordance to each other. El Mokhtari et al. [52] model inference channels as weighted matrices. In each row, the first column is a sensitive data that an owner wishes to keep private, and all the remaining columns describe the attributes that must be known in order to infer the sensitive attribute. El Mokhtari et al. propose to affect weights to the non-sensitive attributes (i.e., the cells in the matrix) to reflect their influence on the sensitive data.

In this thesis, guided by RQ1, we structure inference channels using a set-based model. Each dependency is captured as a first-order formula encoding a data mining process constraints over the queries metadata obtained by users.

> RQ1 – Facet **Model**: (i) Graph-based model (ii) Set-based model.

Structure used by the system to keep track of the user's knowledge

The InfDSs need to keep track of the information queried by users, as stated in RQ3. The solutions performing the detection at query-time relies on a history which keeps track of information gained by users via queries.

The first approach we have observed is to *embed* the history directly within the model used to represent dependencies, thus reducing the system management burden. For instance, the solutions using Bayesian networks, such as Chen et al. [31], store the user's knowledge as evidences. In other words, when the user queries the value of C5_MIN_LAND_DIST and C5_MIN_RW_WIDTH, the network is updated as depicted in Figure 3.1a to keep track of that those two values are known by the user. Then, when they query the value of LAX_PARK_SQ_FT,

Cette thèse est accessible à l'adresse : https://theses.insa-lyon.fr/publication/2024ISAL0024/these.pdf
© [P. Lachat], [2024], INSA Lyon, tous droits réservés

the Bayesian network is further updated as illustrated in Figure 3.1b. The second type of approach is to represent the history as an *independent* structure. For instance, in the solution proposed by Toland et al. [150], a log is associated to each user where a line corresponds to the result of a query issued by a user. This log is then combined with the model of inference channel they use (i.e., Horn clauses) to logically infer all the data a user can obtain from their knowledge.

In this thesis, we keep track of a large user's knowledge via an independent structure which can be optimized without impacting the detection technique. Moreover, by decoupling the model and the history, inference channels can be updated without impacting the users' knowledge.

RQ3 – Facet **History**: (i) Embedded (ii) Independent

Detection technique used by the system

Based on RQ3, to tackle IAISDs, the detection system must reason on the conditions of disclosure related to known inference channels. We identify which techniques are used to detect inference attacks using a model representing inference channels.

Transaction search

Transaction search is used by the three works [90, 139, 20] modeling inference channels as a set of query transactions. Their approach is straightforward. To determine if a query enables an attacker to perform an inference, the InfDS searches if the query can be combined with queries in this user's history. If one of the candidate transaction (i.e., containing this new query) corresponds to a transaction already computed in the model. Then, this new query is detected as an inference attempt.

Logical inference

Some of the works modeling inference channels as logical formula perform *logical inference*. Pappachan et al. [125] present a system which reasons on the view the user has obtained on the database. The InfDS logically infers all the attributes that can be obtained by using the schema level inference channels. If the resulting set of values contains a sensitive information, the query is detected as leading to an attack. The system presented by Brodsky et al. [25] uses the model to compute all the disclosed information according to the knowledge obtained by a new query, and all knowledge in the history. In case a sensitive information is contained within the set of generated information, the new query is detected as an inference attempt.

Brodsky et al's solution can wrongly detect inference attacks associated to outdated information. In their example, the relation *Employee(Name, Rank, Salary, Department)* is queried by users. A FD between the rank and salary exists (i.e., $Rank \rightarrow Salary$). The mechanism needs to prevent users to obtain the salary associated to a given name. Consequently, when a user queries: all the names and ranks of employees working in the toy department $Q_1 = \{\langle John, Clerk\rangle, \langle Mary, Secretary\rangle\}$, and the salaries of all clerk working in the appliance department $Q_2 = \{\langle Clerk, \$38,000\rangle\}$. Then, the user can leverage the FD to infer the salary associated to John. The query Q_2 is detected as an inference attack attempt. Now, if between the two queries the following updates occur: the database is first updated to promote the instance of the employee named *John* to the rank of manager. Then, a second update increases the salaries of all clerk from 4%. Then, the result of Q_1 does not change and $Q_2 = \{\langle Clerk, \$39,520\rangle\}$. Brodsky et al's solution detects that the Q_2 breach the security rule. Because the disclosed information (i.e., Q_1) are not updated in the history, when processing Q_2 the system wrongly considers that

Cette thèse est accessible à l'adresse : https://theses.insa-lyon.fr/publication/2024ISAL0024/these.pdf
© [P. Lachat], [2024], INSA Lyon, tous droits réservés

the rank of John is still Clerk. Toland et al. [150] extend this work by proposing a method to propagate database updates to the gained user's knowledge. When an instance is updated, the user's knowledge that are referencing the attribute are stamped with either the newest attribute value, or a with a stamp describing that the instance has been deleted. The system is able to track the data which has been released to the user and thus to correctly reason on new queries which occur after database updates.

Probabilistic inference

Similarly to the logical inference, the *probabilistic inference* is leveraged by works that model probabilistic dependencies as inference channels. Guarnieri et al. [77] propose an InfDS which applied new issued queries as evidences on the Bayesian network. For instance, if the logical fact *smokes(Alice).* is queried, and hard evidence is applied on the associated random variable. Using Bayesian inference, the belief of the attacker is updated. An inference attempts is detected if the random variable corresponding to a sensitive value reaches the threshold specified by the system policy. For instance, if after having issued the previous query, Mallory's belief leads to a certainty of 75% that *cancer(Alice)*, an attack is detected. In their solution, since the Bayesian network is used to represent a user's belief, it also acts as an history by keeping track of the obtained evidences (i.e., the queried facts). Chen et al. [31], Lachat et al. [103] (which extend the work of Chen et al.), and Yaseen et al. [163] perform the detection similarly than Guarnieri et al.

On the other hand, El Mokhtari et al. [52] consider their weights matrix of accessed data, and propose to computes a percentage of completion towards an inference. When the completion reaches a given threshold, an attack is detected. Unlike the other works in the same category, the probabilistic representation of the inference channels relies on the weights affected to each attribute. It does not reflect the distribution of attribute values stored in the database that the mechanism is protecting. Yet, we place El Mokhtari et al's work in this category, since it estimate for each sensitive value, a probability interpreted as the confidence an attacker has to infer it.

Revocation

The two last works use *revocation* on a policy. The InfDS proposed by Biskup et al. [19] process queries of propositional atoms (i.e., logical facts denoted by φ_i) about individuals. The policy contains the inference channels modeled as conjunction of facts: $\varphi_i \wedge \ldots$. When a user queries information which appears within their policy, they are revoked from the related sentences. For instance, a policy defining the following sentence $\varphi_1 \wedge \varphi_2$ indicates that those two facts cannot be obtained together by a user. Querying the fact φ_1 does not violate the policy, which leads the system to answer the query. The sentence of the policy is updated and becomes φ_2. Then, querying the fact φ_2 is detected as an inference attacks attempt, since it violates the update sentence. Staddon et al. [144] present an InfDS where inference channels are loosely modeled by a set of tokens. When a user queries an object, they need to use the specific associated token which can be used once. The access to the objects of an inference channel are more and more restricted, until a user tries to query the last object of an inference channel which is detected as an inference attempt.

Guided by RQ3, we leverage in this thesis our inference channels modeled as a set of first-order formula, to logically determine when a new query enables an attacker to satisfy the constraints, thus performing an IAISD.

Cette thèse est accessible à l'adresse : https://theses.insa-lyon.fr/publication/2024ISAL0024/these.pdf
© [P. Lachat], [2024], INSA Lyon, tous droits réservés

	Background Knowledge	Constraint / Dependency	User's knowledge	Inferred Information
Profile database	Internal	Unique	Exact queried value	Value with uncertainty
Sensor database	External	Multiple	Query metadata	Probability distribution

Table 3.6: Summary of differences of assumptions between profile database and sensor database.

RQ3 – Facet **Detection**:
(i) Transaction search (ii) Logical inference (iii) Probabilistic inference (iv) Revocation

3.5 Positioning

In this chapter, we have observed the differences of assumptions between the profile and sensor databases synthetized in Table 3.6. Most of the solutions that can be found in the literature assume that attackers follow an inference strategy [161] where the background knowledge corresponds to some information contained within the database. However, for IAISDs, the attackers follow the *External Information strategy* by using as background knowledge references such as the one presented in Chapter 2. Moreover, we have observed that the knowledge gained by users is represented as the exact data returned by a query. For IAISDs, it is not the exact value of a sensor data point which is important, but rather if the knowledge gained by a user enables them to an inference channel. In IAISDs, the representation of both this user's knowledge and inference channels must be done at the *metadata level*. In addition, for profile databases, the existing works typically consider a single constraint for dependency between attributes (see Section 3.4). In IAISDs we encountered a variety of sensor data types and constraints that an attacker has to follow (see Section 2.1). The last difference is related to the information that an attacker infers when performing an attack. In profile databases, the inferred information is stored within the database. The inferred information can include the value of a sensitive attribute, the existence of a sensitive value for an instance, etc. Instead, in IAISDs the mined information does not correspond to data stored in the sensor database, but instead to a probability distribution over the personal information values (e.g., running, walking, … for the human activity).

Consequently, the work presented in this thesis is positioned as described in Table 3.7. To be able to perform an IAISD, we consider that an attacker exploits one of the known references like those presented in Chapter 2. Hence, they gather information from a sensor database guided by a known environment and known constraints. To tackle RQ1, we model what is guiding an attacker's queries as an *inference channel*. To address RQ2, we choose to model the information about selected sensor data as the *query metadata*, of each query issued to a $\mathtt{DB}_{\mathtt{sen}}$, by capturing:

- The *query parameters*. When using an article as an external source describing how to perform an IAISD, the attacker has to query specific types of sensor data. For instance, to leverage the references that use the *WISDM* dataset [158], an attacker needs to query data produced by an accelerometer and a gyroscope on three axis. Furthermore, the data stream selected by an attacker tells us which environment the attacker targets. Capturing the parameters of a query provides information about the attacker's intent.
- The *query context*. When an attacker performs an IAISD, the sensor database is notified with the inferred personal information. To determine for which individual this information is valid, we need to know precisely the identity of individuals for each targeted environment. Moreover, knowing the identity of the user issuing a query enables keeping track of all

Cette thèse est accessible à l'adresse : https://theses.insa-lyon.fr/publication/2024ISAL0024/these.pdf
© [P. Lachat], [2024], INSA Lyon, tous droits réservés

knowledge they have already gained. Capturing the context of a query provides information about the identity of the user (i.e., a potential attacker) and the individuals (i.e., potential victim of an IAISD).

- The *query result metadata*. The constraints force an attacker to obtain a specific data points to exploit an inference channel. For instance, SSW or TSW implies that an attacker obtains as a query result, a specific number of data points, or data generated during a specific time duration. Capturing metadata (e.g., duration and quantity) about data points in a query result provides information about the capability of an attacker to satisfy the constraints of an inference channel.

In the next chapter, we present our first contribution which precises how the query metadata and the inference channels an attacker can exploit are modeled.

Cette thèse est accessible à l'adresse : https://theses.insa-lyon.fr/publication/2024ISAL0024/these.pdf
© [P. Lachat], [2024], INSA Lyon, tous droits réservés

	Data	Dependency	Level	Model	History	Detection
Biskup (2012)	S. B. Profile	Functional	Data	Set-based	Embedded	Revocation
Biskup (2020)	S. B. Profile	Functional	Data	Set-based	Independent	Transaction search
Chen and Chu (2008)	E. S. Profile	Probabilistic	Schema & Data	Graph-based	Embedded	Bayesian inference
el Mokhtari et al. (2021)	E. S. Profile	Functional	Schema	Set-based	Independent	Probabilistic inference
Guarnieri, Marinovic, and Basin (2017)	S. B. Profile	Probabilistic	Schema & Data	Graph-based	Independent	Bayesian inference
Jebali et al. (2022)	E. S. Profile	Functional, Linkage	Schema	Set-based	Independent	Transaction search
Lachat, Rehn-Sonigo, and Bennani (2020)	E. S. Profile	Probabilistic, Linkage	Schema & Data	Graph-based	Embedded	Bayesian inference
Pappachan et al. (2022)	E. S. Profile	Functional	Schema	Set-based	Independent	Logical inference
Sellami, Hacid, and Gammoudi (2019)	E. S. Profile	Functional, Linkage	Schema & Data	Set-based	Embedded	Transaction search
Staddon (2003)	E. S. Profile	Functional	Data	Set-based	Embedded	Revocation
Toland, Farkas, and Eastman (2010)	E. U. Profile	Functional	Schema & Data	Set-based	Independent	Logical inference
Yaseen and Panda (2012)	E. U. Profile	Probabilistic	Schema & Data	Graph-based	Independent	Probability inference
RICE-Sy	I. a. Sensor	Functional	Metadata	Set-based	Independent	Logical inference

Table 3.7: Comparison of query-time IC mechanisms. E. S. Profile, E. U. Profile, S. B. Profile, and I. a. Sensor correspond to the facet value exact static profile data, exact updatable profile data, static boolean profile fact, exact sensor data points, information about sensor data points, respectively.

Cette thèse est accessible à l'adresse : https://theses.insa-lyon.fr/publication/2024ISAL0024/these.pdf
© [P. Lachat], [2024], INSA Lyon, tous droits réservés

Chapter 4

RICE-M: Raw sensor data based Inference ChannEl Model

As shown in Chapter 2, inference channels implying sensor data are numerous and have each specific constraints to exploit sensor data in order to infer some personal data (e.g., humant activities). Consequently, to be able to detect an Inference Attack Involving Sensor Data (IAISD), we should be able to model these specific constraints (RQ1). Moreover to let the IAIDS detection system be able to check if an attacker has exploited an inference channel (RQ2), we need a model that captures (i) which information and how many time their values are obtained, as we are focusing on the constraints Timestamp based Sliding Window (TSW) and Sequence based Sliding Window (SSW) (ii) what is the identity of the individual whose information has been queried.

We present in this chapter our first contribution called Raw sensor data based Inference ChannEl Model (RICE-M). As depicted in Figure 4.1, we assume that the sensor database is protected by an Inference Detection System (InfDS) against IAISDs. An administrator models the *inference channels* deemed as important using the first part of RICE-M. Then, when a user issues a query to the sensor database, we extract and model the corresponding *user's knowledge* using the second part of RICE-M. In this thesis, for the sake of simplicity, we assume that users are not colluding, i.e., that they are not sharing the user's knowledge they obtain.

The structure of this chapter is organized as follows: We begin by introducing in Section 4.1 the two case studies taken from the references presented in Section 2.1.5 to illustrate the formalization of our proposed model. Then, we introduce the formalization of the user's knowledge and the

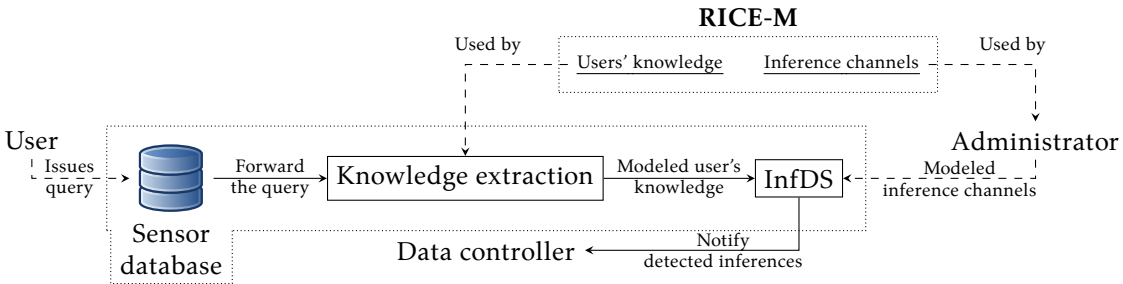


Figure 4.1: Usage context of RICE-M: When a new query issued to the sensor database, the user's knowledge is modeled and processed by the InfDS w.r.t. the known modeled inference channels. Then, the data controller is notified when an inference is detected.

Cette thèse est accessible à l'adresse : https://theses.insa-lyon.fr/publication/2024ISAL0024/these.pdf
© [P. Lachat], [2024], INSA Lyon, tous droits réservés

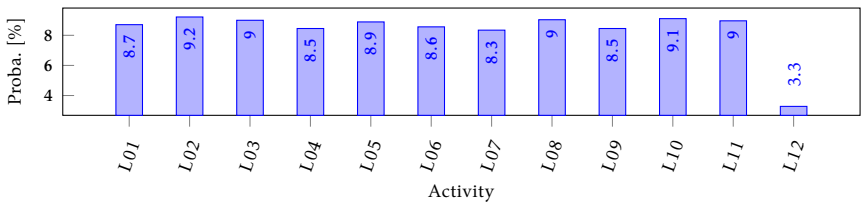


Figure 4.2: Probability distribution of classifying human activities in the mHealth inference channel. The labels L01 to L12 correspond to the following human activities: standing still, sitting and relaxing, lying down, walking, climbing stairs, waist bends forward, frontal elevation of arms, knees bending, cycling, jogging, running, jumping forwards and backwards, respectively.

inference channels in Section 4.3 and Section 4.4, respectively. We discuss the choice and limits of RICE-M in Section 4.5 and conclude this chapter in Section 4.6.

4.1 Case studies description: mHealth & Orange4Home

To illustrate how our proposed model represents both inference channels and user's knowledge for IAISDs, we propose to use two examples of data mining algorithms applied to sensor data as our case studies: *mHealth* [15] and *Orange4Home* [42]. We chose those two references because they both use a publicly available dataset and provide information about their mining process (e.g., type of model used and the parameters of the windows). They enable use to consider the two constraints TSW and SSW. Furthermore, via mHealth, we cover both the individual environment with wearable sensors, and via Orange4Home, the area and sub-area with ambient sensors (see Section 2.2).

4.1.1 mHealth case study

mHealth is a public dataset published by Banos et al. [15] for training ML models to perform human activity recognition. The authors provide detailed descriptions of the training process, which allows to extract the information required to model the inference channel. To do so, we analyze the dataset the authors leverage to determine the considered physical phenomenons and the setting in which observations are made by sensors. Then, we analyze the data processing section presented by Banos et al. [15] to determine how they prepare the sensor data before the training. The two following paragraphs summarizes the dataset properties and how data are processed to train the ML model.

The dataset contains observations (such as acceleration, turn rate, etc.) from three wearable sensors placed on the chest, the right wrist, and the left ankle of a human being. The measures are conducted with a total of 10 volunteers. The median number of data points per volunteer is 120 960, for a total of 1 215 745 data points measured. Each sensor records observations at a fixed frequency while a volunteer performs each of the 12 human activities depicted in Figure 4.2. Each activity is either performed during 1 min, or repeated 20 times for the waist bends forward, frontal elevation of arms, knees bending (crouching), and jump front & back activities. Figure 4.2 displays the complete list of activities. The collected sensor data [124] are represented as a vector of 23 attributes (e.g., x, y, and z axis for the acceleration) and the 24^{th} attribute represents the label of the activity performed by the volunteer during the measurements.

The authors aim to train a decision tree on this dataset to classify the measures into one of the learned activities. To do so, they consider 21 among the 23 attributes, since they judge that the two electrocardiogram measures from the chest sensor are not needed for a first evaluation.

Cette thèse est accessible à l'adresse : https://theses.insa-lyon.fr/publication/2024ISAL0024/these.pdf
© [P. Lachat], [2024], INSA Lyon, tous droits réservés

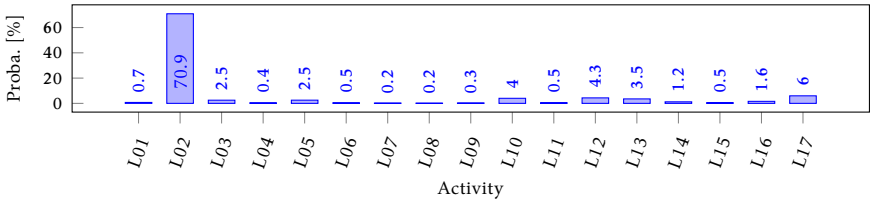


Figure 4.3: Probability distribution of classifying human activities in the Orange4Home inference channel, at home level. The labels L01 to L17 correspond to the following human activities: cleaning, computing, cooking, dressing, eating, entering, going down, going up, leaving, napping, preparing, reading, showering, using the sink, using the toilet, washing the dishes, watching tv.

They employ a non-overlapping sliding window with a duration of 2 s to pre-process the data. They obtain a median F1 score of 98.5% when evaluating the classification performance of their approach. Thus, the constraint that must be satisfied to be able to exploit this inference channel is to know the values of the 21 attributes for a common duration of at least 2 s. The constraint corresponds here to a TSW. By leveraging this dataset, it is possible to extract information about the activities performed by the volunteers wearing the sensors. In this setting, the inferable knowledge is represented by the random variable *activity*. It corresponds to the probability distribution among the 12 identified actions as depicted in Figure 4.2.

4.1.2 Orange4Home case study

The *Orange4Home* dataset, proposed by Cumin et al. [42], is also employed for training ML models in the field of human activity recognition. Similarly to mHealth, we extracted from Cumin et al's article the information used to model Orange4Home as an inference channel. To do so, we follow the same process than for our first case study. The two following paragraphs summaries the dataset properties and how data are processed to train the ML model.

This dataset contains observations (e.g., temperature, CO_2 level, luminosity, etc.) from 236 ambient sensors installed in nine rooms (i.e., entrance, kitchen, living room, toilet, staircase, walkway, bathroom, office, and bedroom) within the same apartment. A total of 746 767 data points [41] are conducted in the presence of a single inhabitant, which means that the mined information is only related to this individual. Those measurements are done while the inhabitant follows a schedule of tasks (e.g., computing, reading, eating, and so on). Figure 4.3 depicts the list of activities performed in different rooms, and repeated for 20 days (except for the *using the toilet* task which is unpredictable). A total of 493 instances of activities are measured in approximately 180 h. The inhabitant indicates manually when they start or stop a task and in which room they perform it. The collected sensor data are represented as a vector containing:

- As a first field, either the timestamp of a measurement or the start of an activity
- The name of the sensor or the indication that the tuple represents a label
- The measurement value or the status and label of the activity. The data collected between the two entry labels `2017-01-30 08:02:25,label,START:Bathroom|Showering` and `2017-01-30 08:18:12,label,STOP:Bathroom|Showering` correspond to the observations performed while the inhabitant performs the *showering* activity in the *bathroom*.

Based on this dataset, Cumin et al. demonstrate in [44] how they train MultiLayer Perceptron (MLP) and Support Vector Machine (SVM) models for human activities recognition. They

Cette thèse est accessible à l'adresse : https://theses.insa-lyon.fr/publication/2024ISAL0024/these.pdf
© [P. Lachat], [2024], INSA Lyon, tous droits réservés

demonstrate that recognition can be performed at two levels: at home level, where sensor data are considered independently from the room where sensors are deployed; and at room level, where models are trained for each room and fusion techniques are utilized to make a final decision. To do so, they consider all sensor data in their dataset, and re-sample each activity instance so that it is 20 sample long, thus forming feature vectors used as a training set. Cumin et al. obtain a F1 score of 89.61% at home level, with the SVM, and 93.05% at room level, with the MLP. The first constraint corresponds here to a SSW with a sequence of size 20. As described in the previous paragraph, when starting an activity, the inhabitant manually records the `[...],label,START:Bathroom|Showering` entry in the dataset. All entries recorded afterwards correspond to data points measured during the activity, until the inhabitant stops the activity by creating a `[...],label,STOP:Bathroom|Showering` entry. Therefore, re-sampling this activity into a sample of 20 data points implies that each sample contains data points in the temporal order they are generated. An attacker needs to query 20 data points which are not generated randomly, but rather 20 data points which have been generated in a common temporal duration. This second constraint corresponds to a TSW. At home level, a single classifier model takes as input 20 data points produced by all sensors to determine the correct class among all the possible activities. At room level, a classifier model takes as input 20 data points produced by sensors deployed in that room only, and decides the correct class among the activities related to the room (e.g., *showering* in the *bathroom*). Thus, the chosen *environment level* (e.g., the home or the room) impacts the duration of the TSW that needs to be satisfied to leverage the mining algorithm. The duration at home level (resp. room level) is determined by computing the median duration of feature vectors over all the activities (resp. over all the activities in a specific room only). The median duration is equal to 15 s at home level (resp. 15 s for the bathroom, 16 s for the bedroom, 10 s for the entrance, 11 s for the kitchen, 15 s for the living room, 17 s for the office, 9 s for the staircase, 19 s for the toilet, 15 s for the walkway).

If both of these constraints are met by the user's queried data, w.r.t. the selected environment level (e.g., the home, the bathroom, etc.), the attacker is able to obtain some knowledge about the activities performed by the inhabitant. Assuming that this attacker queried sensor data without considering a specific room (i.e., home level), the inferrable knowledge in this case is represented by the random variable *activity* defining the probability distribution among the 17 identified activities, as shown in Figure 4.3.

With this second case study, we consider both a new constraint, i.e., SSW, and the fact that sensor data can be generated from different spatial sub-areas (e.g., rooms) within a given area (e.g., a home). In the following section, using our two case studies as illustration, we present how we model the queried information and the inference channels.

4.2 Capturing information to constitute the user's knowledge

To describe which information constitute the user's knowledge, and how they are extracted as depicted in Figure 4.1, let us first describe our example setting. In the following, the name of each new concept is *emphasized* and its formal definition is provided below by using the same name. Figure 4.4 illustrates that the Individual$_1$ shares all data points generated from their wearable sensors as a single *data stream*. Those data points correspond to measurements of three physical phenomenons (e.g., the acceleration). For the sake of simplicity, in our figures we represent those phenomenons using squares with color and pattern instead of labels. Individual$_2$ proceeds the same way with ambient sensors and three new physical phenomenons. The *sensor database* exposes those two data streams to the *users*, and thus *attackers*.

Cette thèse est accessible à l'adresse : https://theses.insa-lyon.fr/publication/2024ISAL0024/these.pdf
© [P. Lachat], [2024], INSA Lyon, tous droits réservés

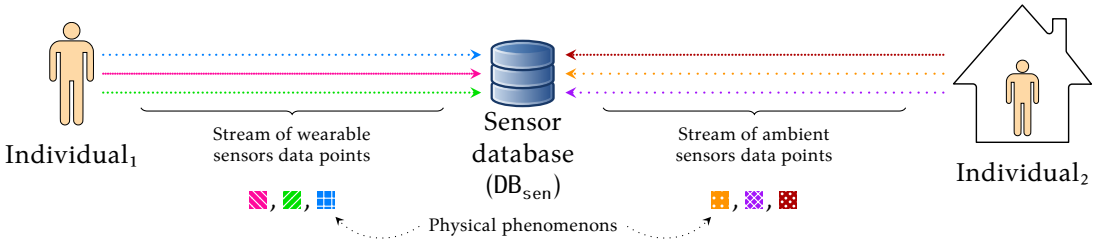


Figure 4.4: Two individuals sharing their sensor data points via two distinct data streams.

$$
\begin{aligned}
\langle \text{query} \rangle &\vdash \textit{SELECT}\ \langle \text{attributes} \rangle\ \textit{FROM}\ \langle \text{streams} \rangle\ \textit{WHERE}\ \langle \text{conditions} \rangle \\
\langle \text{attributes} \rangle &\vdash a \in A \mid a \in A\ ,\ \langle \text{attributes} \rangle \\
\langle \text{streams} \rangle &\vdash s \in S \mid s \in S\ ,\ \langle \text{streams} \rangle \\
\langle \text{conditions} \rangle &\vdash \langle \text{condition} \rangle \mid \langle \text{condition} \rangle\ \textit{AND}\ \langle \text{conditions} \rangle \\
\langle \text{condition} \rangle &\vdash \textit{INTERVAL}\ t \in T
\end{aligned}
$$

(a) Theoretical grammar of queries issued on data streams.

```
SELECT ▨, ▨, ▦
FROM individual1_stream
WHERE INTERVAL (1, 2)
```

(b) $\mathbf{Q_1}$

```
SELECT ▩, ▩, ▩
FROM individual2_stream
WHERE INTERVAL (1, 2)
```

(c) $\mathbf{Q_2}$

Figure 4.5: Query issued to $\mathtt{DB}_{sen}$.

Definition 4.2.1 (Data Stream)**.** A **data stream** $s \in \mathcal{S}$ is a temporally ordered sequence of sensor measurements.

Definition 4.2.2 (Sensor Database)**.** A **sensor database** $\mathtt{DB}_{sen} \subseteq \mathcal{S}$ is a set of data streams on which queries can be issued.

Definition 4.2.3 (User)**.** A **user** $u \in \mathcal{U}$ has an authorized access to $\mathtt{DB}_{sen}$.

Definition 4.2.4 (Attacker)**.** An **attacker** is a user $u \in \mathcal{U}$ which attempts to perform an IAISD.

As illustrated in Figure 4.5, for the sake of simplicity we assume that queries follow the SQL-inspired theoretical grammar of Figure 4.5a. A user issues queries such as Q_1 in Figure 4.5b and Q_2 in Figure 4.5c. They select the data points associated to some phenomenons and generated during some interval in a data stream.

4.2.1 Query parameters

As stated in Section 3.5, the first information we capture are the *query parameters*, since they provide information about the users' intent. Hence, for each query we extract:

- The *attributes*, i.e., the name of the queried physical phenomenon, from the SELECT clause. The attributes of Q_1 (respectively Q_2) are ▨, ▨, and ▦ (respectively ▩, ▩, and ▩).
- The data streams from the FROM clause. The data stream of Q_1 (respectively Q_2) is individual$_1$_stream (respectively individual$_2$_stream).

Cette thèse est accessible à l'adresse : https://theses.insa-lyon.fr/publication/2024ISAL0024/these.pdf
© [P. Lachat], [2024], INSA Lyon, tous droits réservés

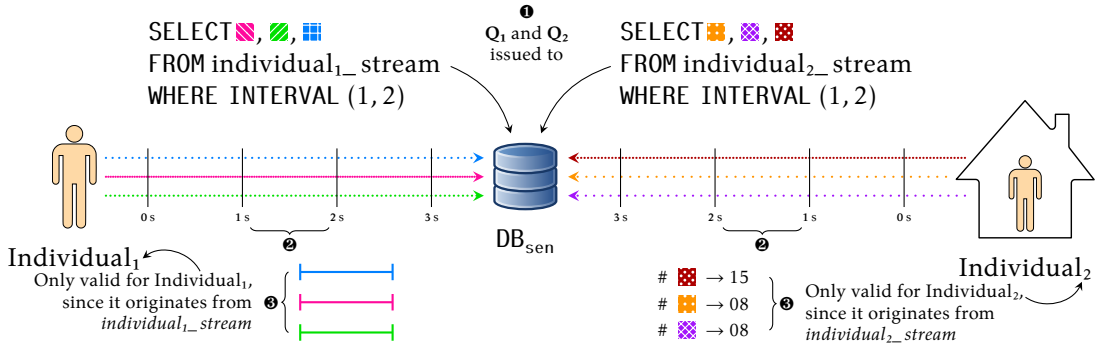


Figure 4.6: The query metadata is valid only for the individuals sharing their sensor data.

- The *time interval* from the `INTERVAL` clause. Sensor data are usually queried by users according to the time they are generated [17]. In this thesis we employ Allen's definition of the time interval [5]. The time interval of both Q_1 and Q_2 are $(1, 2)$. A timestamp is the elapsed number of seconds since the creation of the sensor data stream. Considering TSW, this tells us the beginning timestamp (e.g., 1) and the ending timestamp (e.g., 2) between which generated data points are selected.

Definition 4.2.5 (Attribute)**.** An **attribute** $a \in \mathcal{A}$ is a physical phenomenon measured by sensors. The fixed set of attributes provided by a data stream $s \in \mathcal{S}$ is denoted by $A_s \subseteq \mathcal{A}$.

Definition 4.2.6 (Time Interval)**.** A **time interval** $t \in \mathcal{T}$ is defined as a pair of timestamps which delimits the beginning and end of a query selection condition, denoted by $t = (t^-, t^+)$, $t^- < t^+$, where t^- is the begin timestamp and t^+ is the end timestamp [5].

4.2.2 Query context

The second information we capture are the *query context* provided by the DB_{sen} to enable our model to capture information that enables us to reason on the extracted query parameters. Hence, for each query, we collect:

- The user issuing the processed query. This enables us to keep track of the knowledge associated with each user. Let us consider that both Q_1 and Q_2 in Figure 4.5 are issued by the same user, then the knowledge extracted from both queries will be incorporated in this user's knowledge.

- The *identities of individuals* sharing sensor data in a data stream. The information we capture from a query is relevant to the individuals associated to the selected data stream only. As illustrated in Figure 4.6 for Q_1, the knowledge that data points of the attributes , , and are known during the interval $(1, 2)$ is valid for $Individual_1$ only, since the queried data stream (i.e., $individual_1$_stream) contains only data generated by sensors worn by this individual. We assume that the data controller managing DB_{sen} knows the identities of individuals sharing data for each data stream. We assume that this information is not provided to users, for the sake of protecting individuals' privacy. Instead, we assume that an attacker knows which data stream to query based on external knowledge they obtained. The individuals' identity is captured to later reason on knowledge related to the same

Cette thèse est accessible à l'adresse : https://theses.insa-lyon.fr/publication/2024ISAL0024/these.pdf
© [P. Lachat], [2024], INSA Lyon, tous droits réservés

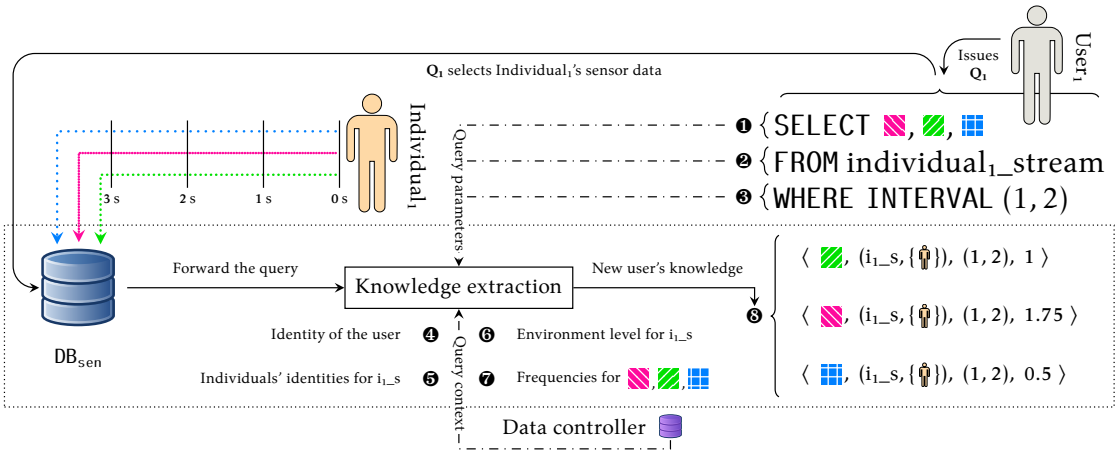


Figure 4.7: Query metadata extraction workflow. For readability sake, the data stream individual$_1$_stream is denoted i$_1$_s.

individual, as well as determining to which individuals an inferred personal information is related.

- The *environment level* in which the sensors of a data stream are deployed. The environment level of a stream providing only data from wearable sensors corresponds to the individual wearing those sensors. In case a stream provides data from ambient sensors, selected data points originate from a specific area or sub-area (e.g., the home or the bathroom in Orange4Home). The environment level is captured to later reason on knowledge associated to different data streams, but originate from the same individual, area, or sub-area.
- The *frequency* at which the data points of an attribute are generated. Considering SSW, we have to examine the quantity of data points obtained in a selected time interval. As illustrated in Figure 4.6 for Q_2, the frequencies of sensors is necessary to compute how many data points generated during the interval $(1, 2)$ for each of the attribute , , and .

Definition 4.2.7 (Individuals' identity)**.** A contextual **identity of an individual** $d \in \mathcal{D}$ is any value uniquely identifies an individual (e.g., a auto-increment integer for each individual).

Definition 4.2.8 (Environment level)**.** An **environment level** $el\colon \mathcal{S} \mapsto \mathcal{L}$ is a function which specifies at which level sensors of a data stream are deployed.

Definition 4.2.9 (Frequency)**.** Each attribute $a \in \mathcal{A}$ has a **frequency** (in Hz), denoted as $fq_a \in \mathbb{R}$, at which a sensor generates data points.

Next, we show how those information are combined to model the user's knowledge.

4.3 Modeling the user's knowledge

The information extracted from a query is used to determine if an attacker is able to perform an IAISD or not. To reason over some information, we have to ensure that they corresponds to data produced in the same *environment*. Indeed, the attacker needs to combine information generated from the same individuals and originating from data streams with the same environment level.

Cette thèse est accessible à l'adresse : https://theses.insa-lyon.fr/publication/2024ISAL0024/these.pdf
© [P. Lachat], [2024], INSA Lyon, tous droits réservés

We propose to represent this environment as a data stream and the individuals' identities. Hence, two environments are similar if they share the same identities and data streams with the same environment level. To verify that all the required attributes are queried and that the data points associated to each of those attribute satisfy the required duration or quantity for TSW or SSW, respectively, we have to organize the extracted information as attribute-level unit of knowledge. We propose the *Metadata Knowledge Unit (MKU)* which models that the queried data points related to an attribute, originating from an environment, are generated over a time interval and at a specific frequency.

Definition 4.3.1 (Environment)**.** An **environment** $env = (s \in \mathcal{S}, D \subseteq \mathcal{D}) \in \mathcal{E}$, $|D| \geq 1$, is a pair defining the relationship between a data stream and the individuals' identities sharing their data in the stream. Two environments are *similar*, denoted by $\simeq$, if they share the same set of identities and reference data streams having the same level. Formally, for $env = (s, D) \in \mathcal{E}$ and $env' = (s', D') \in \mathcal{E}$, $env \neq env'$: $env \simeq env' \Leftrightarrow D = D' \wedge el(s) = el(s')$.

Definition 4.3.2 (Metadata Knowledge Unit (MKU))**.** A **metadata knowledge unit** $mku = \langle a \in \mathcal{A}, env \in \mathcal{E}, t \in \mathcal{T}, fq_a \in \mathbb{R} \rangle \in \mathcal{MK}$ represents the information captured from a query selecting: the physical phenomenon described by an attribute a; deployed in the environment $env \in \mathcal{E}$; for a time interval t; and a frequency fq_a.

All the MKUs extracted from a query constitute the *query metadata*, i.e., the new knowledge obtained by a user. As illustrated by Figure 4.7, three MKUs are extracted from Q_1. To do so, the attributes ❶, the data stream ❷, and the time interval ❸ are extracted from the query. The data controller provides the identity of the user issuing Q_1 ❹, the identity of individuals ❺ and the environment level ❻ associated to the selected data stream, the frequencies associated to each selected attribute ❼. Since only one data stream is selected in the query, a single environment is modeled here. Finally, the first modeled MKU is associated to the attribute ▨, reference the single environment, the selected time interval, and its specific frequency (i.e., 1) at which data points are generated. The second and third MKUs are similar to the first one, except that they are related to the attribute ▧ and ▦, and the frequency 1.75 and 0.5, respectively. The three MKUs form the *query metadata* of the query Q_1.

Definition 4.3.3 (Query Metadata)**.** A **query metadata** $QM_Q \subseteq \mathcal{MK}$ is a set representing the information captured from a query issued to $\mathtt{DB}_{\mathtt{sen}}$, denoted by Q. The set associated to a query issued by a user $u \in \mathcal{U}$ is denoted by QM^u_Q. The ith query issued by u is denoted by $QM^u_{Q_i}$.

We model the query metadata of each query issued to the $\mathtt{DB}_{\mathtt{sen}}$. To detect if a user attempts to perform an IAISD, we reason on all their query metadata. In consequence, we keep track of all MKUs a user obtains via the *Query History Log (QHL)*. The QHL of a given user represents their user's knowledge.

Definition 4.3.4 (Query History Log (QHL))**.** A **query history log** is a function, denoted by $QHL\colon \mathcal{U} \mapsto \mathcal{MK}$, which associates to each user $u \in \mathcal{U}$, the user's knowledge after having issued n queries, denoted by $\{Q_1, \ldots, Q_n\}$, is $QHL(u) = \bigcup_{i=1}^{n} QM^u_{Q_i}$.

Let us show how the user's knowledge is modeled with our two case studies. An attacker exploiting *mHealth* as an inference channel has to obtain a user's knowledge such that MKUs are referring to the 21 required attributes during a common metadata duration of 2 s (see Section 4.1.1). Hence, issuing a query denoted by Q such as `SELECT` $a_1, \ldots, a_{21}$ `FROM` mhealth_individual$_1$_stream `WHERE INTERVAL` $(1, 4)$ provides attacker u with the sufficient data to infer individual$_1$'s activities. The environment from which the queried data originate is here $env =$

Cette thèse est accessible à l'adresse : https://theses.insa-lyon.fr/publication/2024ISAL0024/these.pdf
© [P. Lachat], [2024], INSA Lyon, tous droits réservés

Symbol	Description
$s \in \mathcal{S}$	A sensor data stream.
$DB_{sen} \subseteq \mathcal{S}$	A sensor database $\mathtt{DB}_{\mathtt{sen}}$ containing data streams.
$u \in \mathcal{U}$	A user issuing queries to DB_{sen}.
$a \in \mathcal{A}$	An attribute, i.e., the name of a physical phenomenon.
$A_s \subseteq \mathcal{A}$	Attributes which can be selected in data stream s.
$t \in \mathcal{T}$	A time interval.
$d \in \mathcal{D}$	An individual's identity.
$el\colon \mathcal{S} \mapsto \mathcal{L}$	The environment level of a data stream.
$fq_a \in \mathbb{R}$	The generation frequency of attribute a.
$env \in \mathcal{E}$	An environment in which sensor data are generated.
$mku \in \mathcal{MK}$	A metadata knowledge unit extracted from a query.
$QM^u_{Q_i} \subseteq \mathcal{MK}$	A query metadata extracted from a query Q_i issued by user u.
$QHL\colon \mathcal{U} \mapsto \mathcal{MK}$	The users' knowledge (Query History Log).

Table 4.1: Symbols defined to model user's knowledge in RICE-M.

$(mhealth_individual1_stream, \{individual1\})$. The corresponding query metadata QM^u_Q represents the following set of MKUs: $QM^u_Q = \{\langle a_k, env, (1,4), fq_{a_k}\rangle \mid 1 \leq k \leq 21\}$. In the *Orange4Home* case study, an attacker has to obtain from a specific environment (e.g., the home) a quantity of 20 or more data points, generated by the one or multiple sensors in this environment, over a specific temporal duration (see Section 4.1.2). Issuing a query, denoted by Q', such as `SELECT` a_{15}, a_{123} `FROM` orange4home_stream `WHERE INTERVAL` $(10, 20)$ provides the attacker u with the required data to infer the home inhabitant's activities. The environment from which the queried data originate is here $env = (orange4home_stream, \{individual2\})$. The resulting query metadata correspond to $QM^u_{Q'} = \{\langle a_{15}, env, (10,20), fq_{a_{15}}\rangle, \langle a_{123}, env, (10,20), fq_{a_{123}}\rangle\}$.

To determine if the queries Q and Q' lead to an inference, we have to reason on the modeled MKUs. Let us first consider the query metadata QM^u_Q related to the mHealth case study. We can see that each of the required attributes (i.e., all the a_k), generated for the same individual $individual_1$, are selected over a common interval having a duration of 2 s. For the Orange4Home case study, let us consider that in $QM^u_{Q'}$, the frequencies $fq_{a_{15}} = 0.5$ and $fq_{a_{123}} = 1.25$. Considering the time interval selected in Q', we see that the quantity of obtained data points is equal to 18 for a_{15} and 6 for $fq_{a_{123}}$, hence more that 20 data points are selected in total. In both cases, the modeled MKUs enable us to detect that each query performs an IAISD exploiting a different inference channel.

Our proposed representation, i.e., the Metadata Knowledge Unit (MKU), enables representing all the required information to detect if a user's knowledge enables them, or not, to perform an IAISD. To facilitate future references to our formalization of the user's knowledge, Table 4.1 gives an overview of all the defined symbols. To perform this detection, an InfDS has to be provided with the descriptions of the inference channels attackers can exploit. In the following section, we formally present how we model the inference channels, based on the proposed user's knowledge representation.

4.4 Modeling inference channels

Attackers query the sensor database and are able to gather enough sensor data guided by their background knowledge. Besides modeling the user's knowledge, an InfDS has to model the core

Cette thèse est accessible à l'adresse : https://theses.insa-lyon.fr/publication/2024ISAL0024/these.pdf
© [P. Lachat], [2024], INSA Lyon, tous droits réservés

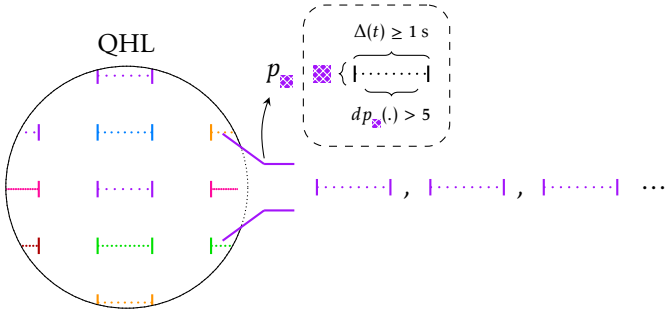


Figure 4.8: A pattern filtering from a QHL the MKUs referencing an attribute, for a duration of at least one seconds, and for a quantity of data points greater than five.

of the attackers' background knowledge, i.e., the description of those constraints. In RICE-M, we capture those descriptions as *inference channels* to endow an InfDS with the background knowledge that an administrator expects the attackers to leverage in IAISDs, as depicted in Figure 4.1. In the following section, we define the concepts used to formally model those inference channels.

4.4.1 Concept definitions

To describe the second part of our model focusing on inference channels, we define, illustrate, and formalize: how the MKUs related to an inference channel are filtered from the user's knowledge based on the data they reference (e.g., attributes); the conditioned filtering which determines if a subset of a user's knowledge enables them to exploit an inference channel; how the personal information inferred when exploiting an inference channel is modeled.

Filtering the user's knowledge

Let us consider a given user $u \in \mathcal{U}$ such as user u has the user's knowledge $QHL(u)$, containing MKUs which capture both: a diversity of metadata related to the inferences that the administrator of the InfDS aims to detect, and sensor data originating from different environment levels, e.g., generated from the wearable sensors of a single individual in mHealth, or the ambient sensors within a smart-home in Orange4Home (see Section 4.1.2). Before checking if a $QHL(u)$ meets the constraints of a modeled inference channel, an InfDS has to filter from the user's knowledge, the MKUs originating from similar environments and capturing information related to an inference channel only. In the following, the name of each new concept is *emphasized* and its formal definition is provided below by using the same name.

Let us illustrate this concept using Figure 4.8. The QHL on the left side contains different MKUs (e.g., referencing different attributes). A hypothetical inference channel requires knowledge of the single attribute ■. We aim to reason on all MKUs that reference this attribute for time intervals with a *duration* $\Delta(t)$ of at least 1 s and a *quantity* of obtained data points greater than 5. To do so, we use the *pattern* $p_{\blacksquare}$ depicted as a purple bottleneck, which provides the set of all MKUs that met our required constraints.

Definition 4.4.1 (Duration)**.** The **duration** is a function $\Delta\colon \mathcal{T} \mapsto \mathbb{R}$ such that $t = (t^-, t^+) \in \mathcal{T}$, $\Delta(t) = t^+ - t^-$.

Cette thèse est accessible à l'adresse : https://theses.insa-lyon.fr/publication/2024ISAL0024/these.pdf
© [P. Lachat], [2024], INSA Lyon, tous droits réservés

Definition 4.4.2 (Quantity)**.** The **quantity** is a function $dp\colon \mathcal{T} \times \mathbb{R} \mapsto \mathbb{Z}$ such that $dp_a(t, fq_a) = \lfloor \Delta(t)/fq_a \rfloor$ for an attribute $a \in \mathcal{A}$. For the sake of simplicity, and without loss of generality, we assume that in an interval $t = (t^-, t^+) \in \mathcal{T}$, t^- corresponds to the timestamp of the first selected data point.

Definition 4.4.3 (Pattern)**.** A **pattern** $p \in \mathcal{P}$, $p\colon \mathcal{MK} \times \mathcal{E} \mapsto \mathcal{MK}_p$, is a function which filters MKUs referencing similar environments, the same attribute, and have specific metadata duration or metadata quantity. The filtered set is denoted by $\mathcal{MK}_p \subseteq \mathcal{MK}$.

For example, in our first case study mHealth, the MKUs to consider must: refer to one of the 21 attributes (see Section 4.1.1) originating from the environment of a given data stream $s \in \mathcal{S}$, with a time interval having a non null duration, and a non null quantity of selected data points (otherwise, the user did not obtain any data). Considering the first attribute of this case study, i.e., the *acceleration from the chest sensor (X axis)* (see Section 4.1.1) denoted by a_1 for the sake of readability, the pattern defining the expected MKUs from a given set $MK \subseteq \mathcal{MK}$ is:

$$p(MK, env') = \{\, mku \mid \exists\, mku = \langle a, env, t, fq_a \rangle \in MK\colon a = a_1 \ \wedge\ env \simeq env' \wedge \Delta(t) > 0 \ \wedge\ dp_a(t, fq_a) > 0 \,\} \tag{4.4.1}$$

Similarly, considering our second case study Orange4Home, the MKUs to consider must: refer to any attribute, originate from the environment of a given data stream s, with a time interval having a non null duration, and a non null quantity of selected data points. Considering a random attribute of this case study, e.g., *bathroom heater temperature* [42] denoted by a_{bht}, the related pattern for a given set MK is Equation 4.4.1, where a_1 is replaced by a_{bht}.

By defining patterns for each attribute required in an inference channel, we can filter from a user's knowledge all the MKUs that have to be analyzed. To determine if a subset of those MKUs enables an attacker to exploit an inference channel, we need to represent the constraints that they must satisfy.

Constrained filtering over the patterns

As illustrated in Figure 4.1, upon receiving from a user $u \in \mathcal{U}$ a new query Q, the InfDS detects if the newly modeled query metadata, denoted by QM_Q^u, and their previously gained knowledge, denoted by $QHL(u)$, enable them to perform an IAISD. Let us illustrate this detection by using Figure 4.9. The set $\mathrm{MK}_{p_{\blacksquare}, p_{\blacksquare}, p_{\blacksquare}}$ on the left side is formed by the three patterns (from left to right) describing MKUs that reference the attribute depicted by ■, ■, and ■ (from top to bottom), respectively. It represents the user's knowledge $QM_Q^u \cup QHL(u)$ considered for the detection. The right side represents the same MKUs as time intervals aligned according to the referenced attributes. For instance, we see that mku_1 to mku_4 are related to the attribute ■. In this example, we consider that an inference channel is exploited if it exists three MKUs referencing each a distinct attribute, and sharing the same time interval t_{ref} with a duration of at least 2 s. Such a situation exists since the MKUs mku_2, mku_5, and mku_8 reference each one of the three required attributes and share a common interval t_{ref} with a suitable duration. In this situation, attacker u's knowledge enables them to perform an IAISD.

Since the pre-processing steps performed before training a ML model are constraining the input sensor data, we consider them as mandatory steps that guide the queries issued by an attacker to obtain suitable sensor data (see Section 2.1.4). In RICE-M, we model constraints as filters over the patterns. The filters determine if a subset of the patterns contains MKUs

Cette thèse est accessible à l'adresse : https://theses.insa-lyon.fr/publication/2024ISAL0024/these.pdf
© [P. Lachat], [2024], INSA Lyon, tous droits réservés

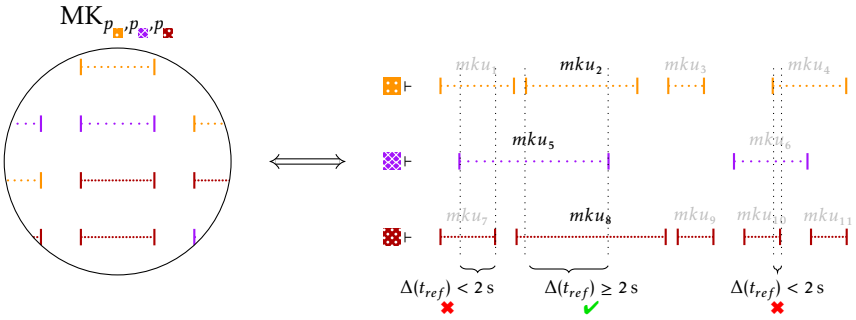


Figure 4.9: The MKUs mku_2, mku_5, and mku_8 are identified by the constrained patterns filter as the subset of the patterns which satisfies the constraint of knowing three attributes for a common time interval with a duration of at least two seconds.

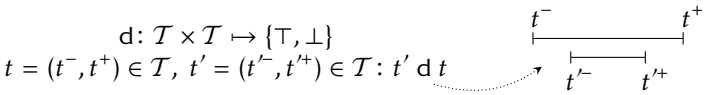


Figure 4.10: During relationship in interval algebra [5].

describing that they have obtained data points related to the required attributes, generated during a common time interval, with a suitable duration (e.g., TSW), or with a required number of points (e.g., SSW). Hence, the resulting set of MKUs is constrained according to the modeled constraints. We refer to this filter as a *Constrained Patterns Filter (CPF)*.

Definition 4.4.4 (Constrained Patterns Filter (CPF))**.** A **constrained patterns filter** $f \in \mathcal{F}$, $f: MK_P \mapsto \mathcal{MK}_f$, is a fonction over a set of MKUs, denoted by $MK_P \subseteq \mathcal{MK}$, described by the set of patterns, denoted by P, in the inference channel. In case a subset of those MKUs satisfies the encoded constraint, it extracts the subset $MK_f \subseteq MK_P$. Otherwise, it returns an empty set.

Modeling TSW and SSW as a CPF implies the definition of logical goals parametrized by a specific quantity of collected data points or a duration during which they are generated. Those values can change depending on the environment in which sensor data are collected from. In Orange4Home, the value associated to the quantity is fixed to 20 data points. The value associated to the duration depends on the considered environment (at home level or at room level). If two distinct users query sensor data from the *bathroom*, and the *bedroom*, respectively, then, the duration to meet w.r.t. the TSW is different (see Section 4.1.2). Consequently, the modelization of constraints for inference channels has to be generic to cover this diversity of settings. To reason about the time interval, in this thesis we employ Allen's interval algebra [5]. For the sake of readability, we introduce the different relationships when using them.

Considering once again mHealth, the set $MK_{p_1,\ldots,p_{21}}$ corresponds to all the MKUs referencing one of the 21 attributes, defined in this case study, and associated to the set of individuals' identities D. In this inference channel, the constraints are related to the TSW. Informally, the constraint encodes the fact that, if it exists a time interval with a duration equal or greater than 2 s, common to a subset of $MK_{p_1,\ldots,p_{21}}$ containing MKUs referencing all the required attributes, then the user's knowledge enables the attacker to perform an attack. Here, we employ from

Cette thèse est accessible à l'adresse : https://theses.insa-lyon.fr/publication/2024ISAL0024/these.pdf
© [P. Lachat], [2024], INSA Lyon, tous droits réservés

Allen's interval algebra [5] the relationship *during*, denoted by the operator d and illustrated by Figure 4.10, to verify that an interval occurs during the interval referenced in an MKU. In other words, it means that the attacker has gathered the necessary sensor data, during the right duration, to exploit a mining algorithm. For mHealth, the required duration is equal to 2 s, since it is independent from the environment. Indeed, the constraint can be defined for this set of MKUs, without considering from which data stream they are related, because all sensors are wearable, hence only a single individual is referenced in those MKUs. Formally, this constraint is defined as:

$$f(MK_{p_1,\dots,p_{21}}) = \begin{cases} MK' \subseteq MK_{p_1,\dots,p_{21}}, & \text{if } \exists\, t_{ref} \in \mathcal{T} : \Delta(t_{ref}) \geq 2\,\text{s}, \\ & \quad \exists\, a \in \{a_1, \dots, a_{21}\}, \exists!\, \langle a, _, t, _\rangle \in MK' : t_{ref}\ \mathsf{d}\ t \\ \varnothing, & \text{otherwise} \end{cases} \tag{4.4.2}$$

Now considering Orange4Home, like for mHealth, the set $MK_{p_1,\dots,p_{256}}$ corresponds to all the MKUs referencing one of the 256 attributes of this case study, and associated to the set of individuals' identities D. Here p_1 denotes the MKUs related to the first attribute in Orange4Home, whereas in the previous paragraph, p_1 denotes the first attribute of mHealth. Informally, the CPF encodes the fact that, if a time interval exists, having a duration specific to the environment where data are queried, common to a subset of $MK_{p_1,\dots,p_{256}}$ where the total quantity of data points is equal or greater than 20, then an attack is detected. In other words, it means that the attacker has gathered the necessary amount of data points, for the right period of time, to exploit a mining algorithm. For instance, at the *home* level (see Section 4.1.2), the required duration is set to 15 s. The required quantity is set to 20, since it is independent from any environment level. Formally, for the home level, the constraint is defined as:

$$f(MK_{p_1,\dots,p_{256}}) = \begin{cases} MK' \subseteq MK_{p_1,\dots,p_{256}}, & \text{if } \exists\, t_{ref} \in \mathcal{T} : \Delta(t_{ref}) \geq 15\,\text{s}, \\ & \quad \exists\, a \in \{a_1, \dots, a_{256}\}, \exists!\, \langle a, _, t, _\rangle \in MK' : t_{ref}\ \mathsf{d}\ t \wedge \\ & \quad \sum_{\langle a,_,t,fq_a\rangle \in MK'} dp_a(t, fq_a) \geq 20 \\ \varnothing, & \text{otherwise} \end{cases} \tag{4.4.3}$$

The constraints are modeled as constraints over MKUs. Yet, when an attacker has the required user's knowledge to perform an IAISD, they are able to infer some personal information related to an individual. The final concept to introduce is the representation of the inferrable knowledge obtained when exploiting an inference channel.

The inferred personal information

Upon the exploitation of an inference channel, an attacker infers some personal information about an individual. This personal information is not stored within the sensor database used by the attacker. Since in IAISDs, attackers exploit data mining algorithms as inference channels, the knowledge they obtain on an individual usually corresponds to a distribution of probabilities over values. For instance, in mHealth the knowledge corresponds to human activities performed by individuals. For the reasons explained in Section 3.5, in our model, we represent both inference channels and user's knowledge at the metadata level. It implies that an InfDS cannot determine the exact inferred distribution. Consequently, via the description of inference channels provided to an InfDS by its administrator, as illustrated in Figure 4.1, they have to define the distribution

Cette thèse est accessible à l'adresse : https://theses.insa-lyon.fr/publication/2024ISAL0024/these.pdf
© [P. Lachat], [2024], INSA Lyon, tous droits réservés

of probabilities that an attacker obtains upon exploiting a channel. This process is illustrated in Section 4.1 for our two case studies. Both distributions of probabilities are obtained by reproducing the training proposed by Banos et al. [15] and Cumin et al. [42], for mHealth and Orange4Home, respectively. Then, by extracting the learned distribution from the models. The difficulty of this process relies on the reproducibility of the training. The obtained distribution is as exact as the model is accurate. The *inferrable knowledge* is modeled as a random variable describing an *inferrable attribute* (e.g., human activities), having a given domain of values (e.g., cycling, climbing stairs, and so on), and the distribution of probabilities over this domain. In the following definition, we formally define the inferrable knowledge obtained by attackers.

Definition 4.4.5 (Inferrable attribute)**.** An **inferrable attribute** $a_{inf} \in A_{inf}$ denotes a personal information that can be inferred by an attacker via an inference channel. Such an attribute cannot be queried from a sensor database, formally $A_{inf} \cap A = \emptyset$. Each attribute has a non-empty domain of values, denoted by $dom(a_{inf})$.

Definition 4.4.6 (Inferrable knowledge)**.** An **inferrable knowledge** $X \in \mathcal{X}$ is a random variable related to an inferrable attribute. It assigns a probability to each value in the domain of an inferrable attribute. Formally, $\forall\ a_{inf} \in A_{inf},\ X\colon dom(a_{inf}) \mapsto [0,1]$

Considering mHealth, the inferrable attribute is denoted by $activity_{inf}$ = {standing still,…,jumping forwards and backwards} as depicted in Figure 4.2. For Orange4Home, the inferrable attribute is denoted $activity_{inf}$ = {cleaning,…,watching tv} as depicted in Figure 4.3.

In mHealth, the corresponding inferrable knowledge, denoted by $X_{activity_{inf}}$, is the distribution over the activities illustrated in Figure 4.2. Similarly, for Orange4Home, the corresponding inferrable knowledge, also denoted by $X_{activity_{inf}}$, is illustrated in Figure 4.3. This distribution thus describes a generic knowledge that the attacker obtains about a targeted individual.

In the next section, we formally present how those concepts interact to enable the detection of IAISDs.

4.4.2 Constraints as filters for the user's knowledge

Now that we have defined each concept, we formalize the definition of *inference channel* provided at the beginning of Chapter 1. Each inference channel is made of:

- The patterns defining the expected subset of MKUs that an attacker must have obtained
- The CPFs that filter from the patterns a subset of MKUs describing that this attacker has obtained all the required sensor data
- The inferrable knowledge obtained by the attacker, in case such a subset if filtered from the attacker's knowledge.

Once defined by an administrator, all those descriptions are registered by an InfDS in the *Inference Channel Repository (ICR)*, as illustrated in Figure 4.1, and designated by a unique *inference channel identifier*.

Definition 4.4.7 (Inference Channel)**.** An **inference channel** $ic = \langle P \subseteq \mathcal{P},\ F \subseteq \mathcal{F},\ X \in \mathcal{X} \rangle \in \mathcal{IC},\ |P| > 0,\ |F| > 0$ is a tuple whose members are: a set of patterns P describing the MKUs that the constraints have to consider; a set of constraints F defining the constraints that must be validated to exploit the channel; the inferrable knowledge X obtained upon validation of the constraints.

Cette thèse est accessible à l'adresse : https://theses.insa-lyon.fr/publication/2024ISAL0024/these.pdf
© [P. Lachat], [2024], INSA Lyon, tous droits réservés

Definition 4.4.8 (Inference Channel Repository)**.** The **inference channel repository** $ICR: \mathcal{I} \mapsto \mathcal{IC}$ is a function representing all known descriptions of inference channels registered by an InfDS.

Definition 4.4.9 (Inference Channel Identifier)**.** An **identifier** $i \in \mathcal{I}$ designates uniquely an inference channel. The description of an inference channel identified by i is retrieved via $ICR(i) = \langle P_i, F_i, X_i \rangle$. For the sake of readability, we identify an inference channel ic with its identifier i.

To conclude the description of our two case studies, as inference channels, we summarize the instantiated concept. For the sake of readability, mHealth and Orange4Home are associated to the identifiers $i = 0$ and $i = 1$, respectively.

Considering mHealth first, the patterns are defined as $P_0 = \{p_1, \ldots, p_{21}\}$, where each pattern describes the MKUs referencing one of the 21 required attributes (see Section 4.1.1) and a time interval having a non null duration, as formalized in Equation 4.4.1 for the attribute a_1. This channel is materialized by $F_0 = \{f\}$ which encodes the CPF used to check if among the set of all MKUs described by P_0, it exists a subset referencing all required attributes and having a shared time interval having a duration of at least 2 s, as formalized in Equation 4.4.2. The inferrable knowledge X_0, obtained in case the channel is exploited, corresponds to the distribution of probabilities over all human activities considered in mHealth, illustrated by Figure 4.2 and formalized as the first random variable $X_{activity_{inf}}$ in Section 4.4.1. Hence, we describe the inference channel related to mHealth as $\langle P_0, F_0, X_0 \rangle$.

Similarly, for Orange4Home the patterns are defined as $P_1 = \{p_1, \ldots, p_{256}\}$, where each pattern characterizes the MKUs referencing one of the 256 required attributes (see Section 4.1.2), with a time interval having a non null duration, and a non null number of selected data points. The single constraint $F_1 = \{f'\}$ which enables verifying that in the MKUs described by P_1, it exists a subset which contains a common temporal interval with a duration associated to the data stream environment (e.g., 15 s in the bathroom, see Section 4.1.2), and at least 20 selected data points. This constraint is formalized by Equation 4.4.3. The inferrable knowledge X_1 corresponds to the distribution of probabilities over the human activities considered in Orange4Home, illustrated by Figure 4.3 and formalized as the second random variable $X_{activity_{inf}}$ in Section 4.4.1. Hence, we describe the inference channel related to Orange4Home as $\langle P_1, F_1, X_1 \rangle$.

Our proposed representation of inference channels used in IAISDs enables the capture of the constraints associated with the data mining algorithm corresponding each to a distinct inference channel, as well as the knowledge that an attacker obtains for individuals once performing the inference. To facilitate future references to the formalization of the inference channels, we have compiled all the defined symbols in Table 4.2. In the following section, we first discuss how RICE-M can be extended to further types of constraints and why we model constraints as logical goals.

4.5 Discussion

In this section, we discuss the open issues related to the incorporation of new types of constraints and the problems stemming from the direct usage of mining algorithms by a system to detect IAISDs.

Cette thèse est accessible à l'adresse : https://theses.insa-lyon.fr/publication/2024ISAL0024/these.pdf
© [P. Lachat], [2024], INSA Lyon, tous droits réservés

Symbol	Description
$p \in \mathcal{P}$	A pattern filtering the expected MKUs.
$MK_p \subseteq \mathcal{MK}$	A set of MKUs filtered by the patterns $P = \{p_a, \dots, p_z\}$.
$f \in \mathcal{F}$	A CPF filtering MKUs according to a constraint.
$a_{inf} \in A_{inf}$	An attribute which can be inferred from an inference channel.
$dom(a_{inf})$	The domain of value of an inferrable attribute.
$X_{a_{inf}} \in \mathcal{X}$	A random variable over the values of an inferrable attribute.
$i \in \mathcal{I}$	The identifier of an inference channel.
$ic = \langle P_i, F_i, X_i \rangle \in \mathcal{IC}$	The description of the inference channel identified by i.
$ICR: \mathcal{I} \mapsto \mathcal{IC}$	The repository of associating identifiers and descriptions.

Table 4.2: Symbols defined to model inference channels in RICE-M.

4.5.1 Incorporating more constraints

In this thesis, we focused on two constraints related to sliding windows: (i) Timestamp based Sliding Window (TSW) and (ii) Sequence based Sliding Window (SSW). They are illustrated via the two case studies *mHealth* and *Orange4Home*, respectively (see Section 4.1). Those two types of constraints are chosen since they are the most used approaches, based on our study summarized in Table 2.1. However, other references mining personal information from sensor data describe other kinds of constraints than the two we consider. As depicted by Table 2.1 in Chapter 2, constraints such as the *Aggregation* and the *Sampling* are used by solutions either alone or explicitly combined with one of the two constraints we have considered (TSW and SSW). As a future work, a more generic version of RICE-M needs to consider those two new constraints. To do so, we could assume that the users query the sensor database using an extended version of the theoretical grammar illustrated in Figure 4.5a.

Integrating aggregation-constrained inference channels

For the Aggregation, the sensor database can propose a set of aggregation functions that can be leveraged by a user to directly compute the aggregation of selected data points.

Query structure Let us consider the query Q:

SELECT f(▩), y(▩)
FROM data_stream
WHERE INTERVAL (t^-, t^+)

Query Q selects data points generated within a given time interval for two attributes. Two different aggregation functions are applied to the selected data points and their outputs are returned as the query result. The functions f and y are, for instance, one of the following $\{AVG, MEAN, SUM, COUNT, MIN, MAX, \dots\}$.

MKU structure The used aggregation functions could be directly extracted from the query, during the extraction of other query parameters shown in Figure 4.7. In addition to the attributes and the time interval, the resulting MKUs would have a new field, which would either be empty if no aggregation function is used in the query, or as a value describing each of the used functions and the corresponding values. The patterns can then filter from the user's knowledge, MKUs having an aggregation function. The CPFs can filter those MKUs according to the used aggregation functions and the attributes.

Cette thèse est accessible à l'adresse : https://theses.insa-lyon.fr/publication/2024ISAL0024/these.pdf
© [P. Lachat], [2024], INSA Lyon, tous droits réservés

Integrating sampling-constrained inference channels

For the Sampling, the sensor database can offer the possibility to define how to sample data points in a selected data stream.

Query structure Let us consider the query Q:

```
SELECT ▩, ▩
FROM data_stream
WHERE INTERVAL (t⁻, t⁺) AND SAMPLE 0.75
```

Query Q selects with an equal probability of 75%, the data points generated during a given time interval for two attributes. The sensor database can propose more complex functions to define how selected data points are sampled.

MKU structure Here, the new query parameter corresponds to the sampling technique used. Like for the Aggregation, the patterns and constraints of an inference channel could refer to this new parameter to enforce the description of MKUs and the constraints related to the sampling approach (e.g., the probability threshold value). To be consistant, the descriptions of inference channels should refer to sampling approaches which could be used to query the sensor database managed by the data controller. Considering this new type of query parameter implies that it should be captured by MKUs. In addition to the previous query parameters, the MKUs would have a new field which is either empty if the sampling clause is not used in the processed query, or set to one of the type of sampling functions offered by the sensor database. In both cases, the open challenge is to comprehensibly study which aggregation functions and sampling techniques are used, as well as the related parameters, to determine the most general representation to incorporate them in the MKUs, patterns, and constraints.

4.5.2 Logical constraints over the user's knowledge

Besides the type of constraints considered, in RICE-M the constraints of inference channels are modeled as logical constraints over the user's knowledge. It means that, for an attacker to perform an IAISD, we assume that they need to obtain the exact query metadata described by those constraints. For instance, in mHealth, data points for 21 attributes must be queried for a common time interval having a duration of 2 s. In Orange4Home, a quantity of at least 20 data points, generated during a common time interval having some duration (e.g., 15 s at home level, see Section 4.1.2). Yet, the attacker may query all required attributes during only 1.75 s, for mHealth, or 18 data points, for Orange4Home, and still be able to obtain knowledge about a targeted individual with a high enough certainty. Reasoning over logical goals implies a binary decision w.r.t. the detection of attacks, and not a computation of the "risk" that a given user performs an attack based on their issued queries. This stems from the fact that it is difficult to quantitatively estimate how such a risk evolves, depending on the information obtained by a user. In other words, is it when querying sensor data for 1.7 s, or 1.75 s, that a user starts to obtain a "high enough" certainty about the inferrable attribute of the channel?

Using those sensor data as input, one could compare the output of some data mining algorithm (e.g., the one proposed by Banos et al. [15] or Cumin et al. [44]), and tell when the probability associated to one of the values of the inferrable attribute is high enough to count as an inference. However, it implies that: (i) The detection system could run all data mining algorithms and perform the detection considering all data points previously queried by a user, before answering an issued query. (ii) To do so, the administrator of the detection system should guarantee that

Cette thèse est accessible à l'adresse : https://theses.insa-lyon.fr/publication/2024ISAL0024/these.pdf
© [P. Lachat], [2024], INSA Lyon, tous droits réservés

each data mining algorithm deployed, for each known inference channel, matches the settings they assume attackers to use when leveraging the same algorithm. While such an approach would be more precise than modeling constraints as logical goals, it could be quite complicated to maintain, due to those the two requirements we have exposed. Instead, with RICE-M, the administrator task is limited to defining the description of inference channels that must be registered. Even if it relies on an external entity to do so, the descriptions have to be created once and do not require maintenance. Reasoning on the query metadata, instead of the exact values of each data point, implies that: the InfDS keeps track of less information, thus reducing the memory burden of detecting IAISDs; furthermore the system has to process fewer information when performing this detection.

To model inference channels, besides the representation of constraints, a crucial aspect is to quantitatively determine the distribution of probabilities of inferrable knowledge. Focusing on an inference channel, one could keep track of the exact value of data points queried by users, and use the data mining algorithm associated to this channel to obtain the exact distribution of probabilities related to the queried data. Yet, running an algorithm for each issued query, and after the detection, could highly increase the query answer time. Those computation could be performed in the background to limit its potential overhead. However, if after detecting an IAISD, the query is answered before estimating the distribution of probabilities, it would lead to situations where an attacker bypasses other detection systems which did not have time to consider this distribution. This comes back to the setting depicted by our motivating example in Section 1.1. Instead, one can model a more generic representation of the inferrable knowledge. By modeling inference channels with RICE-M, the administrator of an InfDS can leverage any article that demonstrates how a data mining algorithm is applied to a sensor dataset to infer personal information. They can reproduce the training of the model presented in the article, and extract the learned distribution of probabilities. We used this approach to obtain the distributions of mHealth and Orange4Home, depicted in Figure 4.2 and Figure 4.3, respectively. Otherwise, the administrator could use other background knowledge to manually define the suitable distribution.

4.6 Conclusion

In this chapter we have proposed the Raw sensor data based Inference ChannEl Model (RICE-M), which tackles the challenges related to RQ1 and RQ2. It formalizes how the *user's knowledge*, obtained when querying the sensor database, is represented as a set of metadata units called Metadata Knowledge Unit (MKU). We illustrate, thanks to the *mHealth* and *Orange4Home* case studies, how different information and environments, in which sensors are deployed, can be modeled adequately. We present the second part of our model which focuses on modeling *inference channels* based on the *patterns* of MKUs, the set of Constrained Patterns Filter (CPF) which filters patterns based on the contraints, and the inferrable knowledge that a user obtains when a non-empty subset of their knowledge satisfies the CPFs. Similarly to the first part, we illustrate how to model our two case studies, thus demonstrating the expressiveness of RICE-M. We have discussed the open challenges related to the modelization of other types of constraints and the logical approach we propose to model sliding windows constraints. To the best of our knowledge, we propose the first model at metadata level for the task of detecting IAISDs. Yet, our model does not provide any reasoning capability for such a detection. In the following chapter, we present our detection system based on RICE-M.

Cette thèse est accessible à l'adresse : https://theses.insa-lyon.fr/publication/2024ISAL0024/these.pdf
© [P. Lachat], [2024], INSA Lyon, tous droits réservés

Chapter 5

RICE-Sy: RICE-M based inference detection System

Based on the knowledge modeling presented in the previous chapter, in this chapter, we will present RICE-M based inference detection System (RICE-Sy) which is the system detecting an Inference Attack Involving Sensor Data (IAISD). RICE-Sy is made of two main parts, i.e., the *Knowledge Storage* and the *Reasoner*, which respectively: stores both the Query History Log (QHL), for each known user, and the description of inference channels to consider registered in the Inference Channel Repository (ICR), using the representation proposed in Raw sensor data based Inference ChannEl Model (RICE-M); processes the query metadata to detect if it leads to an IAISD. To formalize our system, we reuse the symbols defined in Chapter 4. To present our contribution, we describe the workflow of RICE-Sy and, by using an example we illustrate and motivate the need of each module constituting the Reasoner.

This chapter is organized as follows: We introduce the general workflow of our proposed system in Section 5.1. In Section 5.2, we focus on detailing the Reasoner module. We explain how the detection of IAISDs is performed and we formalize the *detection module* of RICE-Sy in Section 5.2.1. We motivate in Section 5.2.2 the need of preparing the user's knowledge to perform the detection in every situation. In Section 5.2.3 we present how this preparation is performed by merging the information referenced by MKUs, and we formalize the *consolidation module* of our system. We explain in Section 5.2.4 how only a subset of previously obtained MKUs can be considered for both the consolidation and the detection modules. In Section 5.2.5, we formalize the two *filtering modules* of the Reasoner. We present in Section 5.3 the complete workflow of our proposed system. We discuss our proposed contribution in Section 5.4 and conclude this chapter in Section 5.5.

5.1 Generic workflow of RICE-Sy: The Reasoner & The Knowledge Storage

To formally present how the detection of IAISDs is performed by RICE-Sy, we first provide a global overview of its architecture and workflow illustrated by Figure 5.1. When a user issues a query to the sensor database ❶, the data controller managing this DB_{sen} extracts the query knowledge ❷. This process is described in Section 4.3 of the previous chapter. The resulting query metadata corresponds to the first input of the Reasoner module ❸. The second input corresponds to the QHL of the user issuing the query ❹. The third input of the Reasoner is the description of all the known inference channels ❺. Then, by considering the new user's knowledge extracted from the query and the user's knowledge previously obtained by issuing

Cette thèse est accessible à l'adresse : https://theses.insa-lyon.fr/publication/2024ISAL0024/these.pdf
© [P. Lachat], [2024], INSA Lyon, tous droits réservés

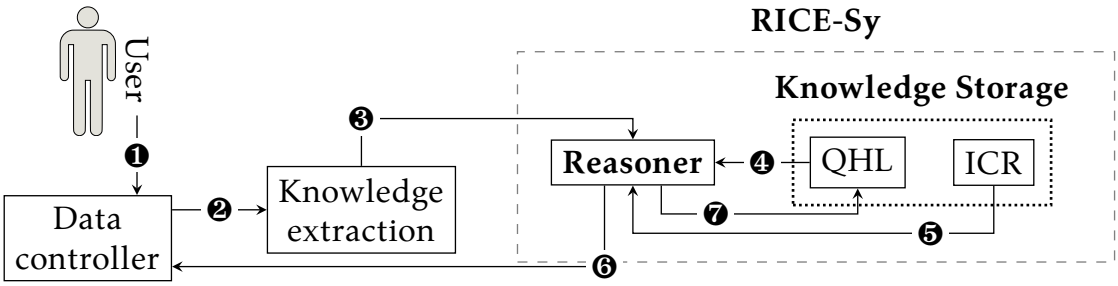
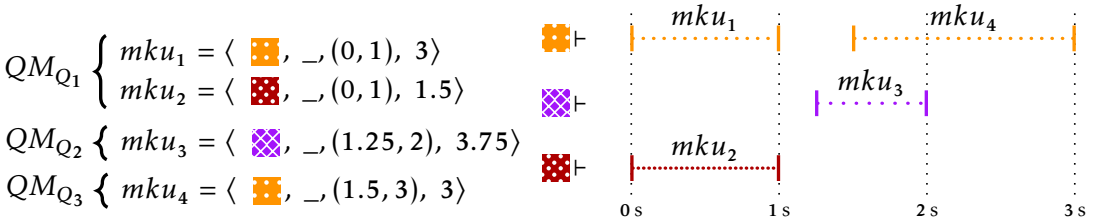


Figure 5.1: Generic workflow of RICE-Sy. The knowledge extraction is depicted in Figure 4.7.



(a) The initial queries metadata.

(b) The initial QHL, after having processed the initial queries metadata.

Figure 5.2: Example of an initial user's knowledge. Since all the MKUs refer the same environment, it is denoted by _ for the sake of simplicity.

queries, the Reasoner determines if one of the inference channels is exploited, or not, by the user. Based on this result, the Reasoner notifies the data controller to indicate if the user has gained some knowledge, or not ❻. In case no IAISD is detected, the Reasoner inserts the new query metadata in the user's QHL to keep track of their newly obtained knowledge ❼.

The purpose of the Knowledge Storage module is to maintain two of the Reasoner inputs. The QHL is automatically updated during each detection. We assume that the administrator of a RICE-Sy instance regularly performs a technological monitoring. Each time a new inference channel is identified, the administrator models it using RICE-M, and enriches the ICR with its representation. Let us illustrate the information in the Knowledge Storage.

We assume that the administrator has described a single inference channel identified by $i = 2$ in the ICR. The set of patterns $P_2 = \{p_{\blacksquare}, p_{\blacksquare}, p_{\blacksquare}\}$ (see Table 4.2) define the MKUs in $QHL(u)$ referencing: the attribute , , or ; similar environments; and a time interval with a non null duration. The set of Constrained Patterns Filter (CPF) F_2 contains a single CPF f defining that all those attributes must be referenced by MKUs during the same common interval having a duration of 1 s. The inferrable knowledge X_2 is not relevant in this example.

Moreover, we assume the following hypothesis about our example. A single user $u \in \mathcal{U}$ is issuing queries to the same data stream, denoted by $s \in \mathcal{S}$. The following attributes are accessible via the data stream $A_s = \{$, , $\}$. The environment related to this data stream contains the identity of a single individual d, i.e., $env = (s, \{d\})$. For the shake of readability, env is denoted by _, since it is not used in this example. The queries Q_1, Q_2, and Q_3 do not lead to an IAISD according to the inference channel $i = 2$. As illustrated in Figure 5.2, the three extracted query metadata depicted in Figure 5.2a lead to the $QHL(u)$ in Figure 5.2b. The process of updating the QHL ❼ is straightforward. Let us consider the empty QHL, $QHL(u) = \varnothing$. Processing the query metadata (i) QM_{Q_1} leads to the insertion of mku_1 and mku_2, i.e., $QHL(u) = \varnothing \cup \{mku_1, mku_2\}$ (ii) QM_{Q_2} leads to

Cette thèse est accessible à l'adresse : https://theses.insa-lyon.fr/publication/2024ISAL0024/these.pdf
© [P. Lachat], [2024], INSA Lyon, tous droits réservés



(a) A new query metadata, denoted by QM_{Q_4}, which does not lead to an IAISD.



(b) A new query metadata, denoted by QM_{Q_5}, which leads to an IAISD.

Figure 5.3: Example of new queries metadata. Since all the MKUs refer the same environment, it is denoted by _ for the sake of simplicity.

$QHL(u) = \{mku_1, mku_2\} \cup \{mku_3\}$ and (iii) QM_{Q_3} leads to $QHL(u) = \{mku_1, mku_2, mku_3\} \cup \{mku_4\}$ In this Figure, the time interval of mku_1 describes that sensor data generated between 0 s and 1 s in the stream s have been selected. At any point in time, we assume that the QHL contains only queries metadata which does not enable the exploitation of an inference channel in the ICR. In the following section, we present how the Reasoner module detects IAISDs.

5.2 The Reasoner module

To intuitively illustrate how the Reasoner module works, let us consider the two queries metadata illustrated in Figure 5.3. We focus first on an example which does not lead to an IAISD, then we present the two cases where a query metadata leads to an IAISD.

From this initial state depicted by Figure 5.2, user u issues the new query Q_4 selecting the single attribute ▩ for a single time interval. As depicted in Figure 5.3a, it results in the query metadata QM_{Q_4}. To perform the detection, RICE-Sy has to consider the mku_5 obtained via QM_{Q_4} and all MKUs within $QHL(u)$. Then, the system has to search within the set of all those MKUs to check if a subset of this user's knowledge matches the set of patterns P_2 and satisfies the CPF f. In our example, we observe how mku_5 is temporally positioned w.r.t. the other MKUs in Figure 5.2b. We see that it shares the common interval $t' = (1.5, 2.25)$ with the mku_4 of QM_{Q_3} and $t'' = (1.25, 2)$ with the mku_3 of QM_{Q_2}. However, for the inference channel identified by $i = 2$, both t' and t'' are: common to MKUs referencing only two of the three attributes in A_s; less than the expected duration, i.e., $\Delta(t') < 1$ s and $\Delta(t'') < 1$ s. Considering both the state of $QHL(u)$ in Figure 5.2b and the QM_{Q_4} content (i.e., mku_5), the new user's knowledge obtained via QM_{Q_4} does not enable user u to exploit the inference channel. The Reasoner notifies the data controller that user u did not infer personal information about the individual d. Finally, the mku_5 is inserted in $QHL(u)$ to track user u's gained knowledge.

Let us consider that a new user u' queries $\mathtt{DB}_{\mathtt{sen}}$. The two cases where a query metadata QM_Q enables user u' to exploit the inference channel occur when:

- Put together with $QHL(u)$, QM_Q allows to exploit an inference channel.
  Let us consider that user u' has the QHL depicted by Figure 5.2b. Then, they issue a new query metadata $QM_{Q_5} = \{mku_6\}$ illustrated by Figure 5.3b. mku_6 shares a common time interval $t' = (0, 1)$ with both mku_1 and mku_2 of QM_{Q_1}. The interval t' is common to MKUs referencing each of the three attributes in A_s and it has a suitable duration $\Delta(t') = 1$ s. Hence, considering the initial $QHL(u')$, the new user's knowledge obtained via QM_{Q_5} enables the user to exploit the inference channel identified by $i = 2$.

Cette thèse est accessible à l'adresse : https://theses.insa-lyon.fr/publication/2024ISAL0024/these.pdf
© [P. Lachat], [2024], INSA Lyon, tous droits réservés

- QM_Q contains itself the required user's knowledge to exploit an inference channel. For instance, let us consider that user u' has an empty QHL. They obtain a first query metadata $QM_{Q_{1'}} = \{mku_1,\ mku_2,\ mku_6\}$, selecting some of the MKUs depicted in Figure 5.2. Then, those three MKUs enable user u' to exploit the inference channel identified by $i = 2$ without leveraging MKUs from $QHL(u')$.

Hence, at this generic level, the only task of the Reasoner is to perform the detection of IAISDs for each query issued by each user. In the following section, we formalize how the detection of IAISDs is performed by the Reasoner.

5.2.1 The detection module

In our system, the Reasoner incorporates the *detection module* whose task is to detect IAISDs. For a given query, this task is formalized as a search problem among a set of MKUs. Users have the capability to query multiple data streams through a query (see Figure 4.5a). The query metadata may contain MKUs referencing different identities of individuals. This depends of the environment in which the selected data points have been generated. The *detection* task is formally defined in Definition 5.2.1.

Definition 5.2.1 (Detection)**.** The **detection** is a function $detection : \mathcal{MK} \times \mathcal{I} \mapsto \mathcal{D} \times \mathcal{X}$. It searches within a set of MKUs, if a known inference channel can be exploited according to the environment of the queried sensor data. It returns a set of pairs containing the inferrable knowledge of each exploited channel, associated with the individuals' identity referenced by the MKUs enabling the inference.

The detection module is formalized by Algorithm 5.1. The first main function, denoted by *detection*, implements Definition 5.2.1. Since we assume that users are not colluding during a detection, the considered MKUs are related to a single user $u \in \mathcal{U}$. The search set of this problem is thus the set MK_{search}. The module performs the detection by considering the environments referenced in the query metadata QM_Q^u, since the patterns need to ensure that we reason on MKUs, from the $QHL(u)$, referencing similar environments (see Section 4.4.1). The detection is thus independent from how sensor data are available either as a single data stream or via multiple streams. Moreover, only the environments referenced in the MKUs of the query metadata are considered, since it is the query metadata which provides new knowledge which may lead to an inference. Hence, *detection* iterates through each environment referenced in the new query metadata, and each inference channel in ICR.

The second function, denoted by *detection_for*, is called for each known inference channel in the ICR and each environment referenced in the QM_Q^u. It first applies the patterns to filter from user u's global knowledge, denoted by MK_{search}, the MKUs that have the required attribute, metadata duration, … considered for this channel. The resulting set is denoted by $MK_i \subseteq MK_{search}$. It corresponds to the input of the CPFs in F_i. If a non-empty subset of the input, denoted by MK', satisfies all CPFs, it implies that user u has some knowledge originating from the same data stream, or from similar data streams, which enables them to perform an IAISD for those individuals. Consequently, the function $detection_for$ returns the inferrable knowledge X_i that user u infers for the individuals' identities, denoted by D. Otherwise, in case a CPF is not validated, it means that user u has no, or not enough, knowledge to exploit the channel identified by i. The function then returns an empty set. To illustrate how RICE-Sy performs this detection, let us instantiate the example presented previously.

Assume user u issues a new query to the data stream s which provides sensor data shared by the individual d. Then, the function *detection* in Algorithm 5.1 is executed with the following

Cette thèse est accessible à l'adresse : https://theses.insa-lyon.fr/publication/2024ISAL0024/these.pdf
© [P. Lachat], [2024], INSA Lyon, tous droits réservés

Algorithm 5.1: Detecting IAISDs based on a user's global knowledge and the ICR.

Inputs: QM_Q^u User u's new query metadata.
$QHL(u)$ User u's previously obtained MKUs.
ICR The inference channel repository.

Output: A set of pairs, each containing individuals' identities and the inferrable knowledge inferred by user u.

1 **Function** *detection_for*$(i,\ MK_{search},\ env = (_, D))$
2 $MK_i \leftarrow \bigcup_{p \in P_i} p(MK_{search}, env)$ // MKUs suitable for the channel i
// If a subset of those MKUs satisfies all the CPFs of the channel i
3 **if** $MK_i \neq \varnothing \ \wedge\ \exists!\ MK' \subseteq MK_i, MK' \neq \varnothing \ \wedge\ \forall\ f \in F_i \colon MK' \subseteq f(MK_i)$ **then**
4 **return** $\{(D, X_i)\}$ // The user u infers X_i for D
5 **return** $\varnothing$
6 **Function** *detection*$(QM_Q^u,\ QHL(u),\ ICR)$
7 $detected \leftarrow \varnothing$
8 $MK_{search} \leftarrow QM_Q^u \cup QHL(u)$
9 $envs \leftarrow \{\, env \mid \exists\ \langle _, env, _, _ \rangle \in QM^u \,\}$
// Checks if, considering one of the query metadata environment
10 **foreach** $env \in envs$ **do**
11 **foreach** inference channel identifier i registered in ICR **do**
// The user u exploits, or not, the inference channel identified by i
12 $detected \leftarrow detected \cup detection_for(i,\ MK_{search},\ env)$
13 **return** *detected*

parameters: the new query metadata QM_{Q_4} depicted in Figure 5.3a; user u's QHL, denoted by $QHL(u)$, in the state depicted in Figure 5.2b; the ICR contains a single inference channel identified by $i = 2$. Hence, $ICR = \{\langle P_3, F_3, X_3 \rangle\}$. The only environment in QM_{Q_4} is $env = (s, \{d\})$. The function *detection_for* is executed once with the following parameters: the inference channel identifier $i = 2$; the search set $MK_{search} = QM_{Q_4} \cup QHL(u)$; the environment env. The set of patterns applied to the search set MK_{search} results in three distinct subsets: $p_{\blacksquare}$ defines the subset $\{mku_1, mku_4\}$, $p_{\blacksquare}$ defines the subset $\{mku_3\}$, and $p_{\blacksquare}$ defines the subset $\{mku_2, mku_5\}$. The union of those three subsets is denoted by MK_3. It is equal to MK_{search}, since all MKUs reference: one of the three attributes in A_s, the same environment env, and a strictly positive duration and quantity metadata. As presented previously, MK_3 does not satisfy the single CPF in F_3: $f(MK_3) = \varnothing$. Consequently, both *detection_for* and *detection* return an empty set, thus notifying the data controller that QM_{Q_4} does not lead to an IAISD.

Let us assume that instead of QM_{Q_4}, the function *detection* receives the query metadata QM_{Q_5} depicted in Figure 5.3b. Once again, only the environment env is captured. It thus calls *detection_for* using the previous parameters values, except for the search set, which is now $MK_{search} = QM_{Q_5} \cup QHL(u)$. The patterns applied to the search set result in: $p_{\blacksquare}$ defines the subset $\{mku_1, mku_4\}$, $p_{\blacksquare}$ defines the subset $\{mku_3, mku_6\}$, and $p_{\blacksquare}$ defines the subset $\{mku_2\}$. As in the previous example, only a CPF is applied to MK_3: $f(MK_3) = \{mku_1, mku_2, mku_3\}$. Consequently, *detection_for* and then *detection* returns $\{(\{d\}, X_2)\}$. Since the set is not empty, it notifies the sensor database $\mathtt{DB}_{\mathtt{sen}}$ that QM_{Q_5} leads to an IAISD.

Now, let us assume that a user u' has never issued any queries. Their QHL is thus empty,

Cette thèse est accessible à l'adresse : https://theses.insa-lyon.fr/publication/2024ISAL0024/these.pdf
© [P. Lachat], [2024], INSA Lyon, tous droits réservés

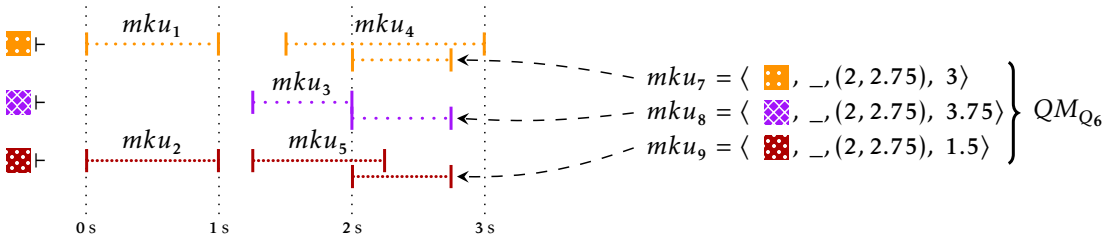


Figure 5.4: New query metadata, denoted by QM_{Q_6}, an the QHL obtained after RICE-Sy has processed QM_{Q_4} in Figure 5.3a. The MKUs of QM_{Q_6} are displayed lower to differentiate them from the one within the QHL.

$QHL(u') = \emptyset$. The function *detection* receives the query metadata $QM_{Q_1'} = \{mku_1, mku_2, mku_6\}$ (see the MKUs in Figure 5.2). Only the environment *env* is captured. The function *detection_for* uses the previous parameters values, except for the search set, which is now $MK_{search} = QM_{Q_1'}$. The patterns filter all the MKUs within the search set, formally $MK_3 = MK_{search}$. As in the previous example, only a CPF is applied to MK_3: $f(MK_3) = MK_{search}$. Consequently, *detection_for* and then *detection* returns $\{(\{d\}, X_2)\}$. Since the set is not empty, it notifies the sensor database $\mathtt{DB}_{\text{sen}}$ that $QM_{Q_1'}$ leads to an IAISD.

Finally, we consider that an individual, with the identity d, has sensor data shared via multiple data streams, instead of a single one. The environment level in the patterns enables RICE-Sy to reason over MKUs associated to different data streams. To illustrate this situation, we assume that two data streams s_1 and s_2 exists. Each stream provides the following attributes: $A_{s_1} = \{$ $\}$ and $A_{s_2} = \{$, $\}$. The two environments are here $env_1 = (s_1, \{d\})$ and $env_2 = (s_2, \{d\})$, which have the same environment level, i.e., the individual wearing the sensors. The MKUs $\{mku_1, mku_4\}$ reference env_1 and the MKUs $\{mku_2, mku_3, mku_5, mku_6\}$ reference env_2. Since both the individuals' identity and the environment level are equal, for QM_{Q_4} and QM_{Q_5}, the patterns P_3 define a set MK_3 containing the same MKUs than presented in the two previous examples.

Those examples show how, thanks to the detection module, the Reasoner enables RICE-Sy to detect IAISDs. However, an implicite situation depicted here is that, a user never queries the $\mathtt{DB}_{\text{sen}}$ on attributes over time intervals that do not overlap. By doing so, the attacker put together the data they obtained which leads to an inference situation. In the following section, we illustrate this problem to motivate the introduction of a new process that we call the *consolidation*.

5.2.2 The consolidation of users' knowledge

Let us consider that user u has the $QHL(u) = \{mku_1, mku_2, mku_3, mku_4, mku_5\}$ depicted on the left side of Figure 5.4. This results from the $QHL(u)$ in Figure 5.2b and QM_{Q_4} in Figure 5.3a. In this example, QM_{Q_5} in Figure 5.3b is ignored. Assume now that user u issues the query Q_6 where the three attributes A_s are selected during the interval $(2, 2.75)$. The resulting query metadata is denoted by QM_{Q_6}. The three MKUs are displayed next to MKUs in $QHL(u)$ and with a vertical offset to signify that they come from QM_6^u. No new sensor data are queried for the attribute , since they have already been obtained via mku_4. Instead, for the attributes and , mku_8 and mku_9 extends to the selected time interval the queried data points of their respective attribute. Considering the set of CPFs F_2 from our previous example, if we focus on the attribute , the module proceeds as follows:

Cette thèse est accessible à l'adresse : https://theses.insa-lyon.fr/publication/2024ISAL0024/these.pdf
© [P. Lachat], [2024], INSA Lyon, tous droits réservés

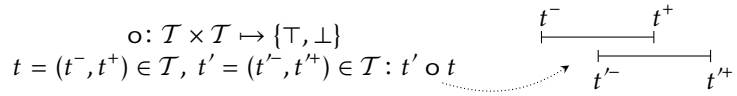


Figure 5.5: Overlapping relationship in interval algebra [5].

1. It checks if mku_2 contains an interval t such that $\Delta(t) \geq 1$ s, which is true, since $t = (0, 1)$. It checks if the MKUs referencing the other attributes, *during* (see Figure 4.10) the same interval as mku_2, share the same interval. Here, mku_1 is the only candidate having an interval that occurs during t. There is not combination of three MKUs referencing all the attributes of A_s. The CPF in F_3 is not satisfied.
2. Then it repeats the same process for mku_5, which has a suitable interval (1.25, 2.25). Here, the possible combinations are: $\{mku_5, mku_3, mku_4\}$, $\{mku_5, mku_8, mku_4\}$, $\{mku_5, mku_3, mku_7\}$, and $\{mku_5, mku_8, mku_7\}$. We observe for each set that there is no interval t with a duration greater than or equal to 1 s that has a *during* relationship (see Figure 4.10) with the intervals of all the MKUs in a set.
3. Finally, the module stops, since there is no other subset of MKUs referencing different attributes and having overlapping time intervals.

Yet, we observe that the interval (1.5, 2.75) has a duration greater than 1 s, and that user u has a continuous user's knowledge over this time interval for each of the three attributes in A_s. Therefore, equipping the Reasoner with the detection module only is not sufficient. A new module should enable merging into a single MKU $\{mku_3, mku_8\}$ and $\{mku_5, mku_9\}$ to allow the detection module to correctly reason over user u's knowledge. In the following section, we present and formalize the consolidation module which is dedicated to tackle the problem identified through the example.

5.2.3 The consolidation module

The Reasoner incorporates the *consolidation module* which ensures that, upon receiving a query metadata QM_Q^u, its MKUs are consolidated with the MKUs in $QHL(u)$. The detection module is then capable of properly performing the detection. The *consolidation* task is formally defined in Definition 5.2.2.

Definition 5.2.2 (Consolidation)**.** The **consolidation** is a function $consolidation\colon \mathcal{MK} \times \mathcal{MK} \mapsto \mathcal{MK} \subseteq \mathcal{MK}$. It is applied to two MKUs $mku = \langle a, env, t = (t^-, t^+), fq_a \rangle$ and $mku' = \langle a', env', t' = (t'^-, t'^+), fq'_a \rangle$ if $a = a' \wedge env \simeq env' \wedge t \circ t'$. The result is a MKU $new_mku = \langle a, (min(t^-, t'^-), max(t^+, t'^+)), fq_a \rangle$. For readability sake, we define the operator $\oplus$ as the application of the function *consolidation* between two MKUs.

Intuitively, the role of the consolidation is to prepare the query metadata used in the search set of the detection module. It ensures that this search set does not contain two MKUs where the time interval of one (i) occurs during the time interval of the other or (ii) overlaps the time interval of the other. By doing so, the consolidation tackles the problem illustrated in Section 5.2.2. We assume that the knowledge extraction process produces a query metadata with consolidated MKUs. The consolidation can occur when considering, in addition, the MKUs originating from the QHL. The consolidation module precedes the detection module in the Reasoner module. Considering the generic workflow illustrated in Figure 5.1, this new module takes as input the query metadata ❸ and consolidates its MKUs with the MKUs in the QHL ❹. The QHL contains

Cette thèse est accessible à l'adresse : https://theses.insa-lyon.fr/publication/2024ISAL0024/these.pdf
© [P. Lachat], [2024], INSA Lyon, tous droits réservés

only consolidated MKUs. Hence, we refer to this QHL as the Consolidated Query History Log (ConsQHL). Performing the consolidation ahead of the detection has two advantages (i) unlike a just-in-time approach during the detection, MKUs are consolidated only once and (ii) when no inference is detected, RICE-Sy caches the consolidated MKUs for future use by updating the ConsQHL.

Algorithm 5.2: Consolidating the MKUs between a query metadata and ConsQHL.

Inputs: $\begin{cases} QM^u_Q & \text{Query metadata} u \in U. \\ ConsQHL(u) & \text{Consolidated QHL of user } u. \end{cases}$

Output: $\begin{cases} MK_{new} & \text{Set of newly consolidation MKUs.} \\ QM_{new} & \text{Set of MKUs, from } QM^u_Q, \text{ not consolidated with MKUs in } ConsQHL(u). \\ MK_{old} & \text{Set of MKUs from } ConsQHL(u), \text{ consolidated with some MKUs in } Q^u_Q. \end{cases}$

1 **Function** *consolidation_process*$(QM^u_Q, ConsQHL(u))$
2 $QM_{new} \leftarrow QM^u$
3 $MK_{new}, MK_{old} \leftarrow \emptyset, \emptyset$
4 **foreach** $(a, env) \in \{(a, env) \mid \exists \langle a, env, _, _\rangle \in QM^u_Q\}$ **do**
5 $MK_{used} \leftarrow \emptyset$ // MKUs used for the consolidation
// MKUs referencing the same attribute and similar environments
6 $MK \leftarrow \{mku \mid mku = \langle a', env', _, _\rangle \in QM^u_Q \cup ConsQHL(u),\ a' = a \ \wedge\ env' \simeq env\}$
// MKUs with an interval which occurs during the interval of another MKU
7 $MK_d \leftarrow \{mku \mid \exists \{mku = \langle _, _, t, _\rangle, mku' = \langle _, _, t', _\rangle\} \subset MK,\ t \operatorname{d} t'\}$
8 $MK_{used} \leftarrow MK_{used} \cup MK_d$ // Those MKUs are used for the consolidation
// All clusters CL of ordered MKUs having overlapping intervals
9 **foreach** $CL \subseteq MK,\ \forall \{mku = \langle _, _, t, _\rangle, mku' = \langle _, _, t', _\rangle\} \subset CL,\ t < t' : t \operatorname{o} t'$ **do**
// Are consolidated into a new MKU
10 $new_mku \leftarrow$ applies Definition 5.2.2 to CL until a single MKU is remaining
11 $MK_{new} \leftarrow MK_{new} \cup \{new_mku\}$
12 $MK_{used} \leftarrow MK_{used} \cup CL$ // The cluster is used for the consolidation
13 $QM_{new} \leftarrow QM_{new} \setminus (MK_{used} \cap QM^u_Q)$ // Remove used MKUs $\subseteq$ the query metadata
14 $MK_{old} \leftarrow MK_{old} \cup (MK_{used} \cap ConsQHL(u))$ // Remove used MKUs $\subseteq$ the ConsQHL
15 **return** $MK_{new}, QM_{new}, MK_{old}$

The *consolidation module* is formalized by Algorithm 5.2. The function *consolidation_process* considers as input, the MKUs originating from user u's new query metadata, denoted by QM^u_Q, and $ConsQHL(u)$. The output corresponds to three sets of MKUs denoted by MK_{new}, QM_{new}, and MK_{old}. Respectively, they represent the newly created MKUs; the MKUs originating from the query metadata which are not consolidated; and the MKUs originating from the QHL, which have been consolidated with MKUs from the query metadata. In case the query metadata does not lead to an IAISD, the set MK_{old} corresponds to old consolidated MKUs to remove from the ConsQHL. The consolidation module provides to the detection module the prepared query metadata $QM^u_Q = MK_{new} \cup QM_{new}$. To illustrate the construction of those set, let us consider the two examples which reuse some of the MKUs depicted in Figure 5.4.

For the first case, we assume that an MKU denoted by mku_7, references the same attribute and a similar environment as a second MKU denoted by mku_4, as depicted in Figure 5.6. If the interval t' of mku_7 occurs during the interval t of mku_4, according to Allen's relationship

Cette thèse est accessible à l'adresse : https://theses.insa-lyon.fr/publication/2024ISAL0024/these.pdf
© [P. Lachat], [2024], INSA Lyon, tous droits réservés

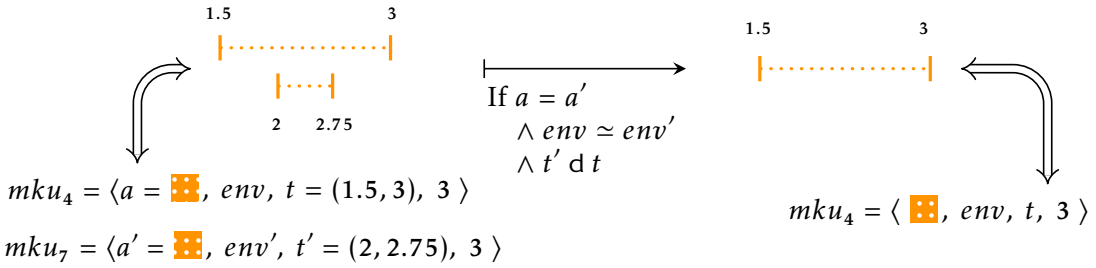


Figure 5.6: Consolidation using the *during* relationship depicted in Figure 4.10. It discards mku_7 and keeps mku_4 depicted in Figure 5.4.

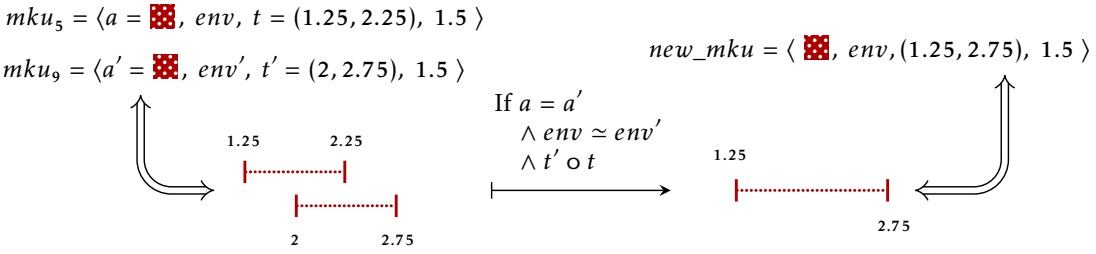


Figure 5.7: Consolidation using the *overlapping* relationship depicted in Figure 5.5. The mku_5 and mku_9 depicted in Figure 5.4 are consolidated into *new_mku*.

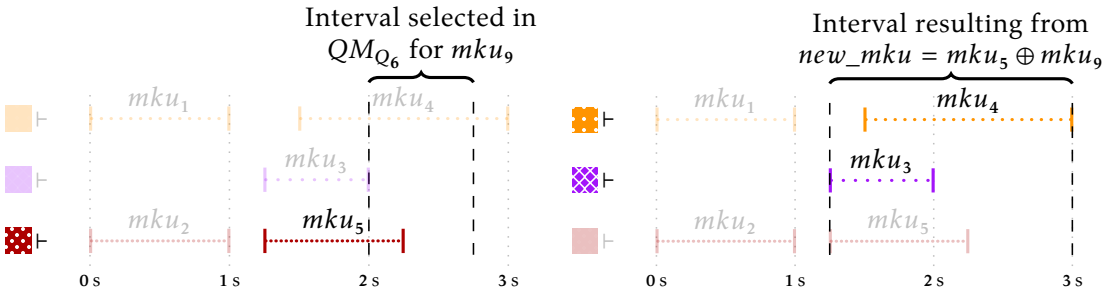
depicted in Figure 4.10, then mku_7 is discarded. This process is done in Line 7 of Algorithm 5.2. All the MKUs that are in a situation similar to mku_7 in our example are put aside in Line 8. They are removed from QM_{new} in Line 13 or added to MK_{old} in Line 14 in case they originate from the query metadata QM^u_Q or $ConsQHL(u)$, respectively. No new MKUs are consolidated in this case. Considering that $QM^u_Q = \{mku_7\}$ and $ConsQHL(u) = \{mku_4\}$, the process returns $MK_{new} = \emptyset$, $QM_{new} = \emptyset$, and $MK_{old} = \emptyset$. Hence, $MK_{search} = \emptyset$.

For the second case, in Figure 5.7 mku_5 and mku_9 reference: the same attribute ■; similar environments; and time intervals that are *overlapping* according to Allen's relationship depicted in Figure 5.5. In this case, they are consolidated in a new MKU denoted by *new_mku*. It references the attribute ■ and the environment of mku_5, since both are similar. The time interval t'' replaces t and t', since they are overlapping each other. All the MKUs that have overlapping time intervals (e.g., mku_5 and mku_9) are consolidated according to Definition 5.2.2. All those MKUs are put aside in Line 12. The new consolidated MKU is added to MK_{new} in Line 11. Similarly to previous case, QM_{new} and MK_{old} are update according to the MKUs put aside. New MKUs are consolidated in this case. Considering that $QM^u_Q = \{mku_7\}$ and $ConsQHL(u) = \{mku_5\}$, the process returns $MK_{new} = \{new_mku\}$, $QM_{new} = \emptyset$, and $MK_{old} = \{mku_5\}$. Hence, $MK_{search} = \{new_mku\}$.

In those two cases, QM_{new} is always empty either because mku_7 has been discarded or mku_9 has been consolidated. Yet, the MKUs from a query metadata may not be consolidated into a new MKU. For instance, when processing $QM_{Q_4} = \{mku_5\}$ and the initial $QHL(u) = \{mku_1, mku_2, mku_3, mku_4\}$, only mku_2 references the same attribute as mku_5. Since their time intervals are not overlapping, they are not consolidated. The process returns $MK_{new} = \emptyset$, $QM_{new} = \{mku_5\}$, and $MK_{old} = \emptyset$. Hence, $MK_{search} = \{mku_5\}$.

A single query metadata contains MKUs which are discarded, consolidated, or neither, depending on the QHL. By incorporating those two modules in the Reasoner, we ensure that the

Cette thèse est accessible à l'adresse : https://theses.insa-lyon.fr/publication/2024ISAL0024/these.pdf
© [P. Lachat], [2024], INSA Lyon, tous droits réservés



(a) For the consolidation of mku_8, obtained via QM_{Q_6}, the mku_5 is retrieved from the $ConsQHL(u)$, since they both refer to the same attribute, and have overlapping intervals.

(b) For the detection, the MKUs mku_3 and mku_4 are retrieved from the $ConsQHL(u)$, since they refer to a different attribute than the newly consolidated MKU mku', and they have overlapping intervals.

Figure 5.8: Example of filtering ConsQHL before the consolidation or the detection.

ConsQHL never contains MKUs that lead to an IAISD, as formally defined by Invariant 5.2.1. Only the query metadata extracted from a new issued query enables a user to perform an IAISDs.

$$\begin{aligned}
&\forall\, u \in \mathcal{U}, \forall\, \langle P, F, _\rangle \in \mathcal{ICR},\ \nexists\, env \in \mathcal{E}: \\
&\qquad MK^u_P = \bigcup_{p \in P} p(ConsQHL(u), env),\ MK^u_P \neq \varnothing \wedge \\
&\qquad \nexists\, MK' \subseteq MK^u_P,\ MK' \neq \varnothing,\ \forall\, f \in F:\ f(MK^u_P) = MK'
\end{aligned} \tag{5.2.1}$$

The size of the ConsQHL continuously grows the more RICE-Sy receives query metadata from users. Performing the consolidation and the detection while considering the whole ConsQHL is quickly intractable. Instead, both modules only have to consider the subset of MKUs originating from the ConsQHL which can be consolidated with or used to build the search set with the MKUs originating from the query metadata. We illustrate this situation in the following section, by reusing our previous example.

5.2.4 Filtering only the relevant subset of users' knowledge

Let us consider the initial situation depicted in Figure 5.4, where user u obtains the query metadata QM_{Q_6}. Upon receiving QM_{Q_6}, the Reasoner relies on the consolidation module to consolidate the search set of the detection module. Yet, as illustrated in Figure 5.8, considering the whole ConsQHL for both the consolidation and the detection is not efficient, since only a subset of MKUs originating from ConsQHL are overlapping with MKUs originating from QM_{Q_6}.

As a first example, we focus on the consolidation of QM_{Q_6} with ConsQHL. We see in Figure 5.8a that mku_9 queried in QM_{Q_6} has the interval $(2, 2.75)$ and references the bottom attribute. As a reminder, to consolidate mku_9, we search another MKU in ConsQHL which references the same attribute and has an overlapping interval. In our example, the only MKU satisfying this criteria is mku_5. Instead of retrieving all MKUs from ConsQHL, only mku_5 can be retrieved for mku_9. This process can be repeated for all the other MKUs within QM_{Q_6}, in order to obtain the subset containing the relevant MKUs originating from ConsQHL.

Now, we focus on the detection performed for QM_{Q_6}. To do so, the Reasoner creates the initial search set that the set of patterns P_2 take as input, by combining the consolidated MKUs

Cette thèse est accessible à l'adresse : https://theses.insa-lyon.fr/publication/2024ISAL0024/these.pdf
© [P. Lachat], [2024], INSA Lyon, tous droits réservés

(i.e., $QM^u_Q = MK_{new} \cup QM_{new}$) with MKUs from ConsQHL. That is, for each consolidated MKU, we search for the MKUs within ConsQHL that have overlapping time intervals. For instance, we observe in Figure 5.8b that the newly consolidated MKU $new_mku = mku_5 \oplus mku_9$ has the interval $(1.25, 3)$. The MKUs with overlapping intervals are mku_3 and mku_4. This process is repeated for all the other consolidated MKUs, in order to form the consolidated search set.

By default, in both situation all the MKUs within the ConsQHL are considered as input of the consolidation module and the detection module. In our example, the set of MKUs is $ConsQHL(u) = \{mku_1, mku_2, mku_3, mku_4, mku_5\}$. However, the consolidation module needs to consider only the subset $\{mku_5, mku_3\} \subset ConsQHL(u)$ w.r.t. to the query metadata QM_{Q_6}. It returns the set $MK_{new} = mku_3 \oplus mku_8, mku_5 \oplus mku_9$, $QM_{new} = \emptyset$, and $MK_{old} = \{mku_3, mku_5\}$. The detection module needs to consider only the subset $\{mku_4\} \subset ConsQHL(u)$ w.r.t. to the consolidation module output. Equipping the Reasoner with two filtering modules enables retrieving only relevant MKUs from the ConsQHL. In the following section, we formalize those two modules.

5.2.5 The filtering modules

The purpose of the two filtering modules is to retrieve the minimum relevant subset of MKUs in ConsQHL to reduce the computation time of both the consolidation and the detection modules.

Algorithm 5.3: Filtering ConsQHL for the consolidation module.

Inputs: $\begin{cases} QM^u_Q & \text{The query metadata of user } u \in U. \\ ConsQHL(u) & \text{The } ConsQHL \text{ of user } u. \end{cases}$

Output: MK_{qbf} is a subset of $ConsQHL(u)$ to consolidated with QM^u_Q.

1 **Function** *query_based_filtering*$(QM^u_Q, ConsQHL(u))$
2 **return** $MK_{qbf} = \{ mku \mid \exists\, mku = \langle a, env, t, _\rangle \in ConsQHL(u), \exists\, mku' = \langle a', env', t', _\rangle \in QM^u_Q, a = a',\ env \simeq env',\ t \circ t' \}$

The *Query Based Filtering (QBF)* module ensures that the input of the consolidation module contains both the MKUs from the query metadata and only the subset of MKUs that, in ConsQHL, can be consolidated with the query metadata. The QBF task is formally defined in Definition 5.2.3. The QBF module is formalized by the function *query_based_filtering* in Algorithm 5.3. For a given user $u \in \mathcal{U}$, it builds the subset, denoted $MK_{qbf} \subseteq ConsQHL(u)$, containing MKUs that reference the same attribute, similar environments, and overlapping time intervals than MKUs in user u's query metadata, denoted by QM^u_Q. The consolidation module receives as a second input MK_{qbf}.

Definition 5.2.3 (Query based filtering)**.** The **query based filtering** is a function denoted as follows $query_based_filtering : \mathcal{MK} \times \mathcal{MK} \mapsto \mathcal{MK}$. It defines the set of MKUs from the ConsQHL that must be consolidated with the MKUs of a query metadata.

The *Search Set Filtering (SSF)* module builds the input of the detection module so that it contains MKUs from ConsQHL that could be combined with the new MKUs, from the output of the consolidation module, in order to form the search set of the detection task. The SSF task is formally defined in Definition 5.2.4. The SSF module is formally defined by the function *search_set_filtering* in Algorithm 5.4. It builds the subset, denoted $MK_{ssf} \subseteq ConsQHL(u) \setminus MK_{old}$, containing MKUs that reference different attributes, similar environments and overlapping time intervals than either: (i) the newly consolidated MKUs (denoted by MK_{new}) or (ii) the new MKUs

Cette thèse est accessible à l'adresse : https://theses.insa-lyon.fr/publication/2024ISAL0024/these.pdf
© [P. Lachat], [2024], INSA Lyon, tous droits réservés

Algorithm 5.4: Filtering ConsQHL for the detection module.

Inputs:
- $ConsQHL(u)$ — The $ConsQHL$ of user u.
- MK_{new} — Set of newly consolidated MKUs.
- QM_{new} — Set of new and non-consolidated MKUs.
- MK_{old} — Subset of $ConsQHL(u)$ used to consolidate MK_{new}.

Output: MK_{ssf} is a subset of $ConsQHL(u)$ to combine with $MK_{new} \cup QM_{new}$.

1 **Function** *search_set_filtering*$(MK_{new}, QM_{new}, MK_{old}, ConsQHL(u))$

2 **return** $MK_{ssf} = \{ mku \mid \exists\, mku = \langle a, env, t, _\rangle \in ConsQHL(u) \setminus MK_{old}, \exists\, mku' = \langle a', env', t', _\rangle \in MK_{new} \cup QM_{new}, a \neq a',\ env \simeq env',\ t \circ t' \}$

from QM_Q^u that where not consolidated (denoted by QM_{new}). Since the MKUs in MK_{old} have been consolidated, we do not consider them in ConsQHL. The detection module receives as a second input MK_{ssf}.

Definition 5.2.4 (Search set filtering)**.** The **search set filtering** is a function denoted as follow $search_set_filtering : \mathcal{MK} \times \mathcal{MK} \mapsto \mathcal{MK}$. It defines the set of MKUs from the ConsQHL that must be combined with the newly consolidated MKUs and the new MKUs from the query metadata.

In the following section, we describe the complete workflow of RICE-Sy and we provide the formal description of the Reasoner module.

5.3 Complete workflow of RICE-Sy

The task of the Reasoner is formally defined by Definition 5.3.1.

Definition 5.3.1 (Reasoner)**.** The *reasoner* is a function $reasoner : \mathcal{MK} \mapsto \mathcal{D} \times \mathcal{X}$. It reasons on the known inference channels to determine if a query metadata enables a user to perform an IAISD.

The Reasoner module is formalized by Algorithm 5.5. The complete workflow of this module is depicted in Figure 5.9. It relies on four distinct modules, namely: the QBF, the consolidation, the SSF, and the detection. Upon receiving the new query metadata QM_Q^u, extracted from a query Q issued by user $u \in \mathcal{U}$ on $\mathtt{DB}_{\mathtt{sen}}$ ❸, the QBF module uses function *query_based_filtering* to extract from $ConsQHL(u)$ ❹ the subset MK_{qbf} ①. Both QM_Q^u and MK_{qbf} constitute the input of the consolidation module. This module uses the function *consolidation_process* to compute three sets ②: the new consolidated MKUs denoted by MK_{new}, the non-consolidated new MKUs originating from QM_Q^u denoted by QM_{new}, and the MKUs from ConsQHL consolidated with MKUs in QM_Q^u denoted by MK_{old}. The new MKUs and $ConsQHL(u)$ serve as input to the SSF module which use function *search_set_filtering* to extract from $ConsQHL(u)$ the subset of MKUs MK_{ssf} ③ to combine with $MK_{new}^u \cup QM_{new}^u$. The detection module builds the search set to check with function *detection*, if user u attempts to exploit a known inference channels described in the ICR ❺. In case no inference is detected, $ConsQHL$ is updated ❼. This process is performed by the Reasoner. It first removes MK_{old}^u from $ConsQHL(u)$ and inserts the new MKUs $MK_{new}^u \cup QM_{new}^u$. By storing the consolidated MKUs and maintaining the Invariant 5.2.1, RICE-Sy enables the detection of IAISDs. The filtering improves the consolidation and detection time of the system by selecting only the relevant stored MKUs. Similarly to the previous chapter, we have compiled all

Cette thèse est accessible à l'adresse : https://theses.insa-lyon.fr/publication/2024ISAL0024/these.pdf
© [P. Lachat], [2024], INSA Lyon, tous droits réservés

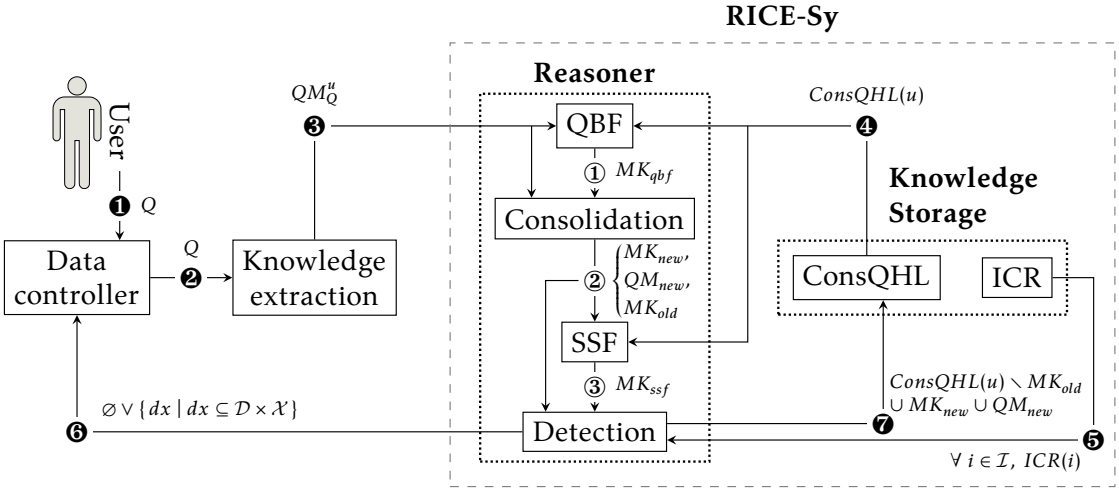


Figure 5.9: Complete workflow of RICE-Sy.

the defined symbols in Table 5.1. Next, we discuss how RICE-Sy can be extended to perform the detection considering inference channels having different constraints than our two case studies.

Algorithm 5.5: Reasoner module of RICE-Sy.

Inputs: QM^u_Q The query metadata of a user $u \in \mathcal{U}$.
Output: $\{dx \mid dx \in \mathcal{D} \times \mathcal{X}\}$ when QM^u_Q enables user u to make an IAISD, otherwise $\varnothing$.

1 $MK_{qbf} \leftarrow query_based_filtering(QM^u_Q, ConsQHL(u))$ // ❸,❹ → ①
2 $QM^u_{new}, MK^u_{new}, MK^u_{old} \leftarrow consolidation_process(QM^u_Q, MK_{qbf})$ // ❸,① → ②
3 $MK_{ssf} \leftarrow search_set_filtering(MK^u_{new}, QM^u_{new}, MK_{old}, ConsQHL(u))$ // ②,❹ → ③
4 $detected \leftarrow detection(QM^u_{new} \cup MK^u_{new}, MK_{ssf}, ICR)$ // ②,③,❺ → ❻
5 **if** $detected = \varnothing$ **then**
6 $\quad ConsQHL(u) \leftarrow (ConsQHL(u) \setminus MK^u_{old}) \cup MK^u_{new} \cup QM^u_{new}$ // ❼
7 **return** $detected$

5.4 Discussion

Next, we discuss the impact faced by RICE-Sy when incorporating more constraints and the cost of including the two filtering modules in our architecture.

5.4.1 Incorporating more constraints

As explained in Section 4.6, as a future work, more constraints have to be considered when modeling MKUs in order for RICE-M to be more generic. From the perspective of RICE-Sy, including the *Aggregation* and the *Sampling* within the MKUs implies that the *consolidation module* must consider which MKUs to consolidate not only according to the attribute they reference, but also based on the newly captured metadata (i.e., the aggregation functions or the sampling techniques). Let us first consider the Aggregation. If two MKUs reference the same attribute, similar environments, and have overlapping intervals, then they can be consolidated.

Cette thèse est accessible à l'adresse : https://theses.insa-lyon.fr/publication/2024ISAL0024/these.pdf
© [P. Lachat], [2024], INSA Lyon, tous droits réservés

Symbol	Description
$detection : \mathcal{MK} \times \mathcal{I} \mapsto \mathcal{D} \times \mathcal{X}$	Detects IAISDs.
$\oplus = consolidation : \mathcal{MK} \times \mathcal{MK} \mapsto \mathcal{MK}$	Consolidates users' knowledge.
$query_based_filtering : \mathcal{MK} \times \mathcal{MK} \mapsto \mathcal{MK}$	Filters ConsQHL w.r.t. a query metadata.
$search_set_filtering : \mathcal{MK} \times \mathcal{MK} \mapsto \mathcal{MK}$	Filters ConsQHL w.r.t. new MKUs.
$reasoner : \mathcal{MK} \mapsto \mathcal{D} \times \mathcal{X}$	Main module of RICE-Sy.

Table 5.1: Symbols defined to formalize RICE-Sy.

To do so, the consolidation module has to create a new MKU, as depicted by Figure 5.7, which will concatenate the aggregations functions referenced in the two initial MKUs. This new MKU will thus contain all the captured information. It will correctly reflect which function was applied to which attribute in the initial queries. On the other hand, the Sampling is more challenging to take into account in the consolidation of two MKUs. Only MKUs extracted from queries where data points are selected using the same sampling methods can be consolidated into a single MKU.

5.4.2 Filtering module

The more a user queries $\mathtt{DB}_{\mathtt{sen}}$, the larger their ConsQHL will become. This growing quantity of MKUs has a direct impact on the input of the consolidation module and the detection module. Hence, those two modules become computationally expensive the more MKUs are stored in ConsQHL. As presented intuitively in Section 5.2.4, filtering ConsQHL can drastically reduce the quantity of MKUs to consider when doing the consolidation and the detection. This greatly lessen the cost of performing the consolidation and the detection, at the expense of increasing the Reasoner computation time. The worst case occurs when the two filtering modules extract all MKUs stored in the ConsQHL. However, the probability that such a case occur decreases the more MKUs are stored, since for each attribute referenced in ConsQHL:

- For the QBF: a query metadata must contains a MKU referencing this attribute with a time interval overlapping all the MKUs in ConsQHL that reference this attribute.
- For the SSF: the consolidation module must output a new MKU referencing this attribute time a time interval overlapping all the MKUs in ConsQHL that reference a different attribute.

While this extreme situation is unlikely, we evaluate how those two filtering modules improve the detection time of RICE-Sy in Chapter 7.

5.5 Conclusion

In this chapter, we have presented RICE-M based inference detection System (RICE-Sy) which tackles the problem of detecting an Inference Attack Involving Sensor Data (IAISD) based on user's knowledge and inference channels modeled with Raw sensor data based Inference ChannEl Model (RICE-M). For each query received by the sensor database, query metadata is extracted and RICE-Sy uses its *Reasoner module* to determine if it enables the exploitation of an inference channel known by the system. The *Knowledge base* stores both the Inference Channel Repository (ICR) and the *Consolidated Query History Log (ConsQHL)*. The Reasoner employs the *consolidation module* to consolidate each Metadata Knowledge Unit (MKU) in a query metadata with MKUs in the ConsQHL. It ensures that the stored knowledge never leads to an inference w.r.t. the channels

Cette thèse est accessible à l'adresse : https://theses.insa-lyon.fr/publication/2024ISAL0024/these.pdf
© [P. Lachat], [2024], INSA Lyon, tous droits réservés

in ICR. It also reduces the quantity of processed and stored MKUs. This step is required by the *detection module* which determines if the newly MKUs and the previous one enable a user to exploit one of the inference channel. In addition, we propose the *query based filtering module* and the *search set module* to take from ConsQHL, only the relevant MKUs for the consolidation and the detection, respectively.

At this stage, RICE-Sy processes every issued query metadata. When an inference attack occurs, it means that a user has gathered all the required knowledge to satisfy the constraints of an inference channel. Evaluating the accuracy of the system is not relevant, since no attack can be missed with respect to the modeled channels. Yet, we must assess the computation time of the Reasoner module. We want to demonstrate that these optimizations work for the different ways users obtain sensor data. To do so, datasets containing query metadata sequences corresponding to different querying behavior of users are required. Such a dataset represents queries issued by users having different querying behaviors. Indeed, while an honest users may regularly query $\mathtt{DB}_{\mathtt{sen}}$ for their work, attackers may have different behaviors guided by the inference channels they aim to exploit. The objective is thus to demonstrate the applicability of our conceptual optimizations by considering the TSW and SSW constraints, for the main identified users' querying behaviors. To the best of our knowledge, such datasets do not exist due to two main issues: no InfDSs consider IAISDs, and no principal querying behaviors for dishonest users performing IAISDs have been identified. Therefore, in the following chapter, we present how we generate synthetic datasets to be able to evaluate our proposed system.

Cette thèse est accessible à l'adresse : https://theses.insa-lyon.fr/publication/2024ISAL0024/these.pdf
© [P. Lachat], [2024], INSA Lyon, tous droits réservés

Chapter 6

Generator: Archetypes & Query metadata sequences

To evaluate our second contribution, i.e., RICE-M based inference detection System (RICE-Sy), we need to have datasets containing queries metadata. To the best of our knowledge, we can distinguish two types of existing datasets: the one proposed in order to evaluate the query-time detection systems, focusing mainly on relational databases containing personal data, but no sensor data [31, 77, 52, 150, 139, 90], and the one proposed to benchmark systems in a sensor database setting [127, 78, 117]. While the first one considers the fact that some users follow inference channels, the inference channels considered are limited to profile databases. They do not contain queries issued by users to a sensor database. Consequently, we cannot reuse those datasets to create sequences of queries leading to an IAISD. In the latter case, the benchmarks can be used to create datasets containing queries issued to sensor databases. Yet, those queries are not issued with the intent of exploiting an inference channel and no queries metadata can be produced by the benchmarks.

Although it is possible to extend those works to obtain datasets containing queries metadata, it would necessitate modifications which are not directly useful for the evaluation of RICE-Sy. As depicted by Figure 6.1, we would have to modify the benchmark ❶ to include the possibility to generate query sequences that lead, or not, to the exploitation of our two case studies (i.e., mHealth and Orange4Home). Then, we would have to implement the knowledge extraction module ❷ to model the query metadata of each issued query (see Section 4.3), thus forming the query metadata datasets we need. All the environment deployed before those last datasets is

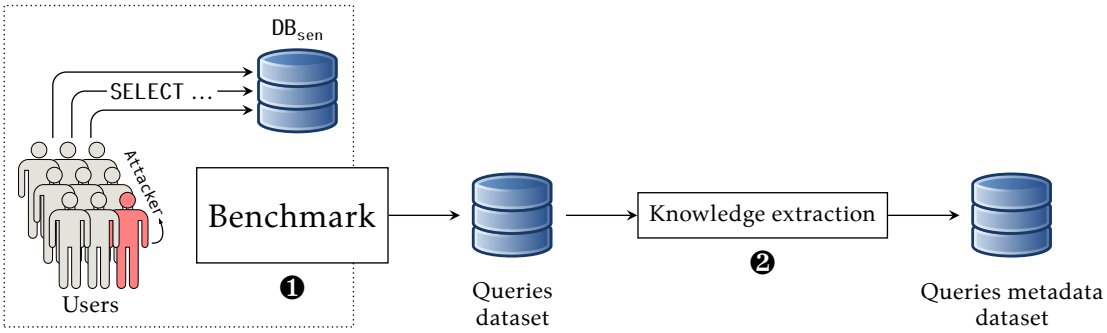


Figure 6.1: Theoretical queries metadata dataset obtained using existing benchmarks.

Cette thèse est accessible à l'adresse : https://theses.insa-lyon.fr/publication/2024ISAL0024/these.pdf
© [P. Lachat], [2024], INSA Lyon, tous droits réservés

not used for the evaluation. Consequently, we propose to directly generate the queries metadata datasets, thus bypassing the steps ❶ and ❷.

Such a dataset is appropriate for evaluating RICE-Sy if it contains:

(I) *Queries metadata are related to distinct users.*
We need to have more than a single user to evaluate how keeping track of each user's knowledge using the Query History Log (QHL) impacts RICE-Sy.

(II) *Both attackers and non-attackers are represented among users.*
This is required to show that our system differentiates between the user's knowledge that do and don't enable users to exploit an inference channel.

(III) *Users selecting sensor data via different approaches.*
To demonstrate that RICE-Sy is not sensitive to the way users query sensor data, we have to provide more than one way sensor data are selected.

(IV) *Approaches that reflect realistic querying behaviors.*
Randomly generating datasets following the previous points only would most likely lead to non-plausible queries metadata, i.e., modeling queries which would not have been issued in real-word situations. Instead, we must generate them based on real users' behaviors.

The purpose of RICE-Sy is not to identify the intent of users based on their querying behavior, nor to determine if they are allowed or not to access new personal information. It's purpose is limited to the detection of IAISDs transmitted to a data controller which can then prevent queries that would breach an individual's privacy. The querying behaviors we consider in this chapter are only leveraged to generated reasonable queries metadata.

The outline of this chapter is the following: we first identify in Section 6.1 the querying behaviors and define three archetypes which guide the generation of queries metadata. In order to control how they are produced, we introduce in Section 6.2, five theoretical concepts and we illustrate how they interact with each other. Then, we present in Section 6.3 how a dataset is generated. Furthermore, we describe, formalize, and illustrate in Section 6.4, how users' query metadata sequences are generated according to our archetypes. In Section 6.5, we present the values affected to the parameters of the generator, according to each archetype. Finally, we discuss assumptions related to the querying behaviors in Section 6.6, and we conclude this chapter in Section 6.7.

6.1 Assumptions about users' querying behaviors

To generate suitable datasets, we need to specify some of our requirements. According to Requirement (III) and Requirement (IV), we assume that users follow different approaches and querying behaviors, dictated by the task for which they query sensor data. In this section, to refine our requirements, we identify how sensor databases are queried and we present the *archetypes* that reflect realistic behaviors. Our methodology is to study the querying behaviors, associated to inference attacks, that are identified in the scientific literature. To do so, we leverage:

(i) The strategies described by Woodall et al. [161], see Section 3.3.1

(ii) The general features of sensor databases presented by Belfkih et al. [17] and Golab et al. [75], as well as the benchmarks [127, 78, 117] used to test sensor databases.

Cette thèse est accessible à l'adresse : https://theses.insa-lyon.fr/publication/2024ISAL0024/these.pdf
© [P. Lachat], [2024], INSA Lyon, tous droits réservés

6.1.1 Querying behaviors

To specify what approaches we consider as realistic, we analyze how sensor data are queried. Woodall et al. [161] show that, when using the *Split Query* strategy, the inference attack is performed by combining the answer of two distinct queries. However, in this thesis, we focus on the *External Information* strategy which does not restrict the number of issued queries. We assume that attackers can perform an inference attack by either querying all required information via a single query or multiple ones.

Besides the quantity of issued queries, different approaches are used to select sensor data. According to Belfkih et al. [17] the main types of queries considered in the literature are:

- Historical queries in order to obtain data related to events that occurred in the past.
- Aggregation queries using functions such a `SUM`, `AVG`, `MAX`, etc.
- Complex queries which are composed of sub-queries using several operators.
- Periodic (or continuous) queries which regularly select data points according to a defined time interval.
- Instant (or ad hoc, or one-time) queries which select data points at a specific time.

Tools used to benchmark sensor databases also provide information about the approaches that are considered as realistic. The TPC (Transaction Processing Performance Council) is an organization which proposes benchmark standards for systems managing data (e.g., online transaction processing database, data integration systems, and so on). To the best of our knowledge, the only benchmark related to sensor data is TPCx-IoT [127]. This benchmark aims to "[…] measure the performance of IoT gateway systems" by simulating sensor data received from edge devices and analyzed via real-time queries. According to the specification of TPCx-IoT, the benchmark generates analytic queries which randomly select generated data points over two time intervals of five seconds. The first query selects data generated up to five seconds before the time at which the query is issued. The second query selects data generated 1800 seconds before the interval of the first query. Gupta et al. [78] propose a benchmark called *SmartBench* which focuses on evaluating DataBase Management System (DBMS) w.r.t. the support of queries performed by near-real time applications and the long term analysis of sensor data. The authors propose twelve template of queries which represent the diversity of queries issued by IoT applications. Only four of those queries select raw sensor data. For instance, the template referred as *Observations* is used to generate queries which select data points generated during a given time interval either: by a single sensor, or by multiple ones. The second relevant template is named *C_Observation* and enables the production of queries selecting data points in an interval, if an only if a given condition is satisfied. Finally, the *Statistics* template produces queries which aggregate data points generated during an interval. Similarly, Mostafa et al. [117] propose a benchmark denominated *SciTS* which is designed to evaluate the performance of time-series databases. They also introduce query templates extracted from "[…] practical and real-life environments […]". The *Raw Data Fetching* and *Data Aggregation* templates correspond respectively to the Observation and Statistic template of Gupta et al.

Based on those observations, we consider that users select the raw sensor data using a temporal interval, without relying on the aggregation queries and the complex queries. Hence, even if we do not reuse those solutions to generate our datasets, we produce queries metadata which correspond to queries following the Observation or Raw Data Fetching templates. Moreover, the continuous nature of sensor data streams usually leads users to query the latest generated data to stay up to date. Based on the observations of Golab et al. [75], we assume that users tend to query

Cette thèse est accessible à l'adresse : https://theses.insa-lyon.fr/publication/2024ISAL0024/these.pdf
© [P. Lachat], [2024], INSA Lyon, tous droits réservés

sensor data in the order they are generated. Users may legitimately need to manually query old data points to perform specific analysis. Among periodic queries, users can issue instant queries, as described by Belfkih et al. [17]. Attackers can select the data of a targeted individual via multiple queries. Those queries can be periodic, instant, or any combination of the two types. Those multiple queries can result in the consolidation of queried information. Based on those assumptions related to querying behaviors, we define below the generic and realistic archetypes used to build our evaluation dataset.

6.1.2 Archetypes

Based on those querying behaviors, we determine three archetypes that define the realistic approaches associated to Requirement (IV). One of those archetype represents users querying the sensor database with a genuine intent, and the two others aim to perform IAISDs.

We refer to the genuine archetype as the *Genuine User (GU)*:

Definition 6.1.1 (Genuine user (GU))**.** A **genuine user** selects attributes required for its job using periodic and instant queries. Those queries never lead to an inference attack.

It corresponds to most of the employees in the external company (see Section 1.1). We assume that an employee following this archetype queries the sensor data required to calibrate sensors only. They do not have any background knowledge related to data mining algorithms and do not perform IAISDs. In addition to the genuine archetype, we consider a first attacking archetype called *One-time Attacker (OA)*:

Definition 6.1.2 (One-time attacker (OA))**.** A **one-time attacker** performs a single inference attack by querying all required information via a single instant query.

It corresponds to the archetype followed by an employee of the external company whose task is not to perform the calibration. Yet, they managed to obtain an authorized access to the sensor database.

Finally, the last archetype corresponds to users which assume that the best way to bypass the InfDS or an administrator is to use multiple queries hidden among genuine queries. We refer to this attacking archetype as *Deceptive Attacker (DA)*:

Definition 6.1.3 (Deceptive attacker (DA))**.** A **deceptive attacker** performs multiple inference attack using any combination of multiple consolidable instant queries selecting the required sensor data, and hides them between non-consolidable periodic and instant queries.

It corresponds for instance, to an employee which performs inference attacks while continuing working in order to not look suspicious within its company. Evaluating RICE-Sy w.r.t. the aforementioned archetypes implies that we have datasets containing queries metadata related to users following such archetypes. In the following section, we present the concepts which control the generation of queries metadata according to our requirements and archetypes.

6.2 Concept definitions

A query metadata is composed of MKUs. Generating queries metadata implies controlling the creation of MKUs. For this purpose, in this section, we introduce concepts that control the generation of a query metadata sequence, and illustrate how they interact at the end. In the following, we reuse the symbols defined in RICE-M. They are either defined in Table 4.1 (e.g., the environment, MKUs, etc.), or in Table 4.2 (e.g., the duration or quantity of inference channels, etc.).

Cette thèse est accessible à l'adresse : https://theses.insa-lyon.fr/publication/2024ISAL0024/these.pdf
© [P. Lachat], [2024], INSA Lyon, tous droits réservés

6.2.1 Regular groups of attributes

A user following the GU or DA archetype regularly selects specific attributes as part of their work. Generating the corresponding MKUs implies considering the *regular group of attributes* referenced.

Definition 6.2.1 (Regular group of attributes)**.** A **regular group of attributes** $rga \subset \mathcal{A}$ is a set of attributes often referenced in MKUs.

6.2.2 Blocks

We use the concept of *blocks* to guide the production of MKUs so that it forms a single query metadata or multiple queries metadata leading, or not, to an inference, with or without consolidation.

Definition 6.2.2 (Block)**.** A **block** $blk = \langle AP \subseteq P \in B_{A_i}, env \in \mathcal{E}, os \in \mathcal{T}, li \in \mathcal{T} \rangle \in \mathcal{B}$ is a tuple defining the information used to generate MKUs. It is built according to an inference channel identified by $i \in \mathcal{I}$. $A_i \subseteq \mathcal{A}$ denotes the required attributes referenced in the set of patterns P_i. B_{A_i} denotes the set of all partitions of A_i. AP is a subset of a partition P in B_{A_i}. env denotes the environment of the MKUs. The time interval os occurs during the time interval of all the MKUs. The time interval li corresponds to the limits of the block, such that $os \, \mathsf{d} \, li$ (see the *during* relationship of Allen's interval algebra [5] depicted by Figure 4.10).

By defining information such as the environment in which sensor data are generated, the attributes, and the shared time interval, we are able to the create blocks having a *malicious status* or a *genuine status*, according to an inference channel.

Definition 6.2.3 (Status of a block)**.** A block $blk \in \mathcal{B}$ is **malicious**, according to an inference channel identified by $i \in \mathcal{I}$, if the produced queries metadata enables to exploit the channel. Otherwise, blk is **genuine**.

A block blk produces queries metadata that select a regular group of attributes rga when the block reference a single set referencing them, i.e., $|AP_{blk}| = 1 \ \wedge \ AP_{blk} = rga$. For the sake of readability, when relevant, we indice symbols with their related block.

6.2.3 Periods

To control how many blocks are defined, how they interact or not with each other, and how often regular groups of attributes are queried, we use the concept of *periods* which correspond to arbitrary units of time.

Definition 6.2.4 (Period)**.** A **period** $period \in \mathcal{T}$ is a time interval during which we define blocks.

A block blk is defined in a period $period$ when its limit li_{blk} occurs during $period$, i.e., $li_{blk} \, \mathsf{d} \, period$. Separately, two genuine blocks generated queries metadata that do not lead to an inference. In case the limits li of those blocks overlap, the queries metadata can be combined to exploit an inference channel. To prevent that, the blocks BLK defined in a period $period$ do not overlap another block, i.e., $\forall \, blk \in \mathcal{B}, \nexists \, blk' \in \mathcal{B}, \, blk \neq blk' \colon li_{blk} \, \mathsf{d} \, period \wedge li_{blk'} \, \mathsf{d} \, period \wedge li_{blk} \, \mathsf{o} \, li_{blk'}$ (see the *overlap* relationship of Allen's interval algebra [5] depicted by Figure 5.5).

Cette thèse est accessible à l'adresse : https://theses.insa-lyon.fr/publication/2024ISAL0024/these.pdf
© [P. Lachat], [2024], INSA Lyon, tous droits réservés

6.2.4 Timelines

To control how periods are arranged, how often queries metadata leading to an inference are produced from blocks, and how often regular groups of attributes are referenced by blocks, we consider the concept of *generation timeline*.

Definition 6.2.5 (Generation Timeline)**.** The **generation timeline** $gt = (t^{-}, \infty) \in \mathcal{T}$ is a time interval during which sensor data points are generated.

The order in which queries metadata are generated over the generation timeline does not always represent the order in which they are issued to a sensor database. To control it, we use the concept of *emission timeline*.

Definition 6.2.6 (Emission Timeline)**.** The **emission timeline** $et = (t^{-}, \infty) \in \mathcal{T}$ is a time interval during which queries are emitted to a sensor database and the query metadata extracted.

The emission timeline is considered upon generating datasets. Yet, we introduce it here to explicitly differentiate the generation and emission time.

All periods are defined over the generation timeline, i.e., $\forall period \in \mathcal{T} : period \, \mathsf{d} \, gt$. Let us consider that a block references a regular group of attributes per period. Then, the occurrence at which this group is selected by queries metadata depends on the duration of periods. The lesser the duration, the more periods can be defined during the same part of the generation timeline. Hence, for the sake of simplicity, we consider that all periods have the same duration and are not overlapping.

A query is issued either as soon as some new data points are generated, in which case the query emission time is equal to the generation time, or later in time, in which case the emission time is strictly greater than the generation time. Hence, a query metadata containing MKUs with a time interval $t \in \mathcal{T}, t \, \mathsf{d} \, gt$ is emitted in a time interval $t' \in \mathcal{T}, t' \, \mathsf{d} \, et$ such that $t' \geq t$.

6.2.5 Sequences of queries metadata

The queries metadata produced during the generation timeline result in a *sequence of queries metadata*.

Definition 6.2.7 (Sequence of queries metadata)**.** A **sequence of queries metadata** $SQ = \langle QM_{Q_1}, \dots, QM_{Q_n} \rangle \subseteq \mathcal{SQ}$, $n \geq 1$ are queries metadata produced from blocks.

In the following section, we illustrate how those concepts interact with each other and how they enable controlling the generation of different queries metadata.

6.2.6 Interactions of concepts

To illustrate how each concept interacts with the other, let us use the example depicted by Figure 6.2. The generation timeline starts at 0 s and is subdivided in four periods denoted by $period_1 = (0, 3)$, $period_2 = (3, 6)$, $period_3 = (6, 9)$, and $period_4 = (10, 13)$. Each period contains at least one block. For instance, $period_1$ contains the block blk_1; $period_2$ the blocks blk_2 and blk_3; $period_3$ the block blk_4; and $period_4$ contains the blocks blk_5, blk_6, and blk_7.

In this example, we assume the existence of an inference channel that requires selecting data points of four attributes (denoted as $a_1, \dots, a_4$) generated during a common duration of at least 400 ms. This channel is arbitrarily chosen to show the diversity of generated queries metadata.

To explain how we illustrate blocks, let us focus on blk_2. It references two sets of attributes, i.e., $\{a_1, a_2\}$ and $\{a_3, a_4\}$. Its limit li corresponds to the blue dotted line, i.e., $li = (3.25, 4.25)$.

Cette thèse est accessible à l'adresse : https://theses.insa-lyon.fr/publication/2024ISAL0024/these.pdf
© [P. Lachat], [2024], INSA Lyon, tous droits réservés

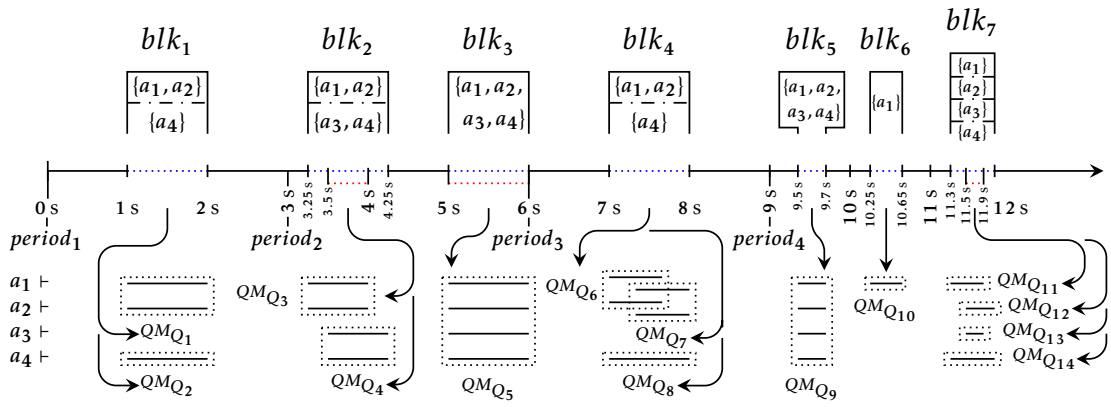


Figure 6.2: Illustration of the concepts.

The interval during which all MKUs overlap is the offsetted red dotted line, i.e., $os = (3.5, 4)$. Below the generation timeline, the queries metadata produced from each block are illustrated via arrows. Each solid line corresponds to a MKU referencing one of the four attribute, illustrated on the left side as $a_i \vdash$. Hence, the queries metadata QM_{Q_3} and QM_{Q_4} are produced from blk_2.

We observe that three blocks have a genuine status illustrated by the missing representation of the *os* time interval. Indeed, the block blk_1 references two sets of attributes $\{a_1, a_2\}$ and $\{a_4\}$. It produces two queries metadata QM_{Q_1} and QM_{Q_2}, containing MKUs (i.e., the solid segments) we assume not to be sufficient to perform an inference in this example. The block blk_4 references the two same sets of attributes. However, it simulates consolidation between the MKUs referencing attributes in the first set, resulting in the queries metadata QM_{Q_6} and QM_{Q_7}. QM_{Q_8} is the queries metadata of the last set of attributes. The block blk_5 is defined with all the required attributes, but for a time interval $li = (9.5, 9.7)$ with a duration smaller than the required one, i.e., $\Delta(li) < 400$ ms. It leads to a single query metadata QM_{Q_9}. The block blk_6 has a time interval which meet the required duration, but it references a single attribute, thus producing a query metadata $QM_{Q_{10}}$ with a single MKU.

The remaining blocks have a malicious status. An attack can occur via multiple queries, e.g., the block blk_2 previously described. The block blk_7 reference each required attribute in a different set. It produces one query metadata per attribute (i.e., $QM_{Q_{11}}, QM_{Q_{12}}, QM_{Q_{13}}, QM_{Q_{14}}$). The time interval of all MKUs overlap during the segment $os = (11.4, 11.8)$. An attack can also occur via a single query. The block blk_3 produces a single query metadata QM_{Q_5} where all MKUs share the same time interval.

This example results in the query metadata sequence $SQ = \langle\, QM_{Q_1}, \ldots, QM_{Q_{14}} \,\rangle$. To generate this sequence, we consider that the regular group of attribute $rga = \{a_1, a_3\}$ as been defined. We observe that those attributes are queried at least once per period. As stated in Section 6.2.4, considering a fix amount of blocks per period, decreasing the duration of periods enables defining more periods during the same part of the generation timeline. Those values influence the generated sequences and thus are parametrized at the dataset level. Hence, we leverage all our concepts to produce datasets containing sequences associated to users. The symbols defined for those concept are summarized in Table 6.1 to improve the readability when referencing them in the following sections. Next section, we present the parameters controlling the workflow of generating datasets.

Cette thèse est accessible à l'adresse : https://theses.insa-lyon.fr/publication/2024ISAL0024/these.pdf
© [P. Lachat], [2024], INSA Lyon, tous droits réservés

Symbol	Description
$rga \subseteq \mathcal{A}$	A regular group of attributs.
$blk \in \mathcal{B}$	A block producing queries metadata.
$period \in \mathcal{T}$	A period during which blocks are defined.
$gt \in \mathcal{T}$	The generation timeline divided in periods.
$et \in \mathcal{T}$	The emission timeline.
$SQ \subseteq \mathcal{SQ}$	A generated sequence of queries metadata.

Table 6.1: Symbols defined to generate queries metadata.

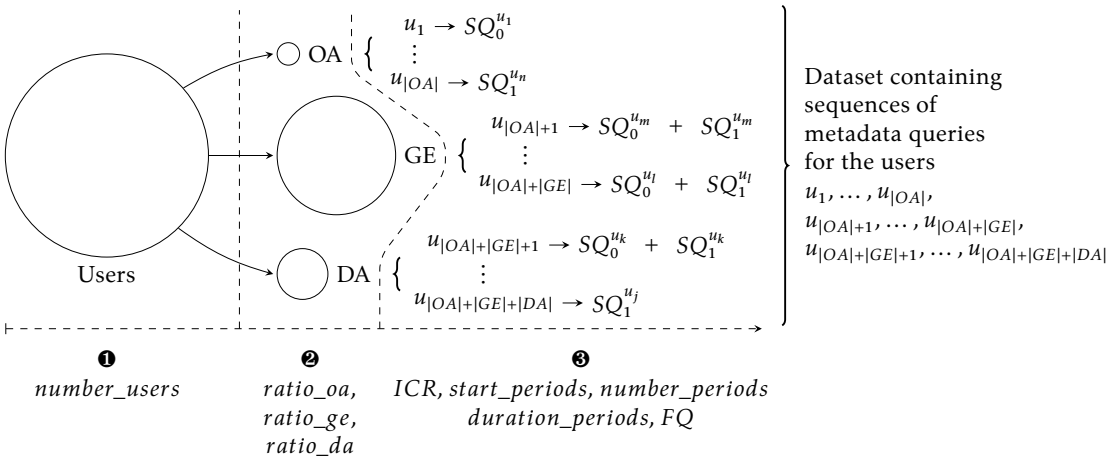


Figure 6.3: Workflow of the dataset generation. The indice 0 (resp. 1) denotes a sequence generated for the inference channel mHealth (resp. Orange4Home).

6.3 Workflow of the dataset generation

The purpose of a generated dataset is to associate sequences of queries metadata to users. As depicted in Figure 6.3, to generate a dataset, the first step is to define the number of users for which queries metadata must be generated ❶. Then, an archetype is chosen for each user. To do so, three ratios are defined in order to determine the number of users which follow the one-time attacker, the genuine employee, and the deceptive attacker archetype, respectively ❷. Finally, for each user and their associated archetype, a query metadata sequence is generated considering ❸: the inference channels (e.g., mHealth denoted by $i = 0$ and Orange4Home $i = 1$) described in the *ICR* of RICE-Sy; the timestamp of the first period on the generation timeline, denoted by *start_period*; the duration of each period in the user's sequence, denoted by *duration_periods*; and the frequencies associated to attributes, denoted by *FQ*. For all users in a dataset, we randomly select the timestamp of the first period on the generation timeline. We establish that all periods in a sequence have the same duration in order to easily modify the regularity of querying behaviors. For coherent sake, in case ICR contains inference channels having SSW constraints (e.g., the Orange4Home case study), the frequencies associated to the attributes of those inference channels are defined beforehand. All sequences generated for those inference channels reference the same frequencies for the same attributes (e.g., all *SQ* having the indice 0 in Figure 6.3). Finally, for the GU and DA archetypes, the generated queries metadata are positioned over the emission timeline to determine in which order the sequence is processed by RICE-Sy.

Cette thèse est accessible à l'adresse : https://theses.insa-lyon.fr/publication/2024ISAL0024/these.pdf
© [P. Lachat], [2024], INSA Lyon, tous droits réservés

Let us consider an hypothetical example in which a dataset is generated for 100 users ($number_users = 100$), with the following distribution of archetypes: 5% of one-time attacker, 85% of genuine employees, and 10% of deceptive attacker ($ratio_ota = 0.05$, $ratio_gu = 0.85$, $ratio_da = 0.1$). We consider that ICR contains only the description of our two case studies: mHealth and Orange4Home. In the remainder of this thesis, we generate datasets by considering a single inference channel for the sake of intelligibility. Yet, to concisely illustrate how a dataset is generated according to both archetypes and inference channels, we a single example which contains each archetype and each inference channel. The following kind of sequences of queries metadata are generated:

- For the 5 OA, the sequence contains a single query metadata which exploits one of the considered inference attacks. For example, user u_1's query metadata sequence is denoted by $SQ^{u_1} = SQ_0^{u_1}$. Only mHealth, denoted by the indice 0, is considered here. A single period is defined for this archetype. The duration of this period must be greater or equal than 2 s to define a malicious block according to mHealth (i.e., where $\Delta(os) \geq 2\,\text{s}$). The frequencies FQ are ignored, since no SSW constraint is considered in this case study. The single query metadata is affected a random emission time, greater than its generation time.

- For the 85 GU, the sequence contains multiple queries metadata which do not exploit any of the inference attacks in the ICR. Hence, for example, the user u_m's query metadata sequence is denoted by $SQ^{u_m} = SQ_0^{u_m} + SQ_1^{u_m}$. In such a sequence the duration of the period is smaller than for the other archetypes, to simulate regularly selected groups of attributes. The data points related to each inference channel may be selected during the same periods (e.g., some data points related to mHealth or Orange4Home are selected during the same periods, then the timestamp of the first period can be set to the same value for the two sequences) or in series (e.g., first some data points related to mHealth and then some data points related to Orange4Home, then the timestamp of the first period can be set to greatly different values). Since one part of the sequence considers Orange4Home (denoted by the indice 1), the frequencies FQ are considered for the referenced attributes A_1 in the queries metadata of $SQ_1^{u_m}$. Most of the queries are emitted in the order and at the time they are generated to simulate the continuous querying. The other queries are emitted in a random order, and with a time greater than their respective generation time.

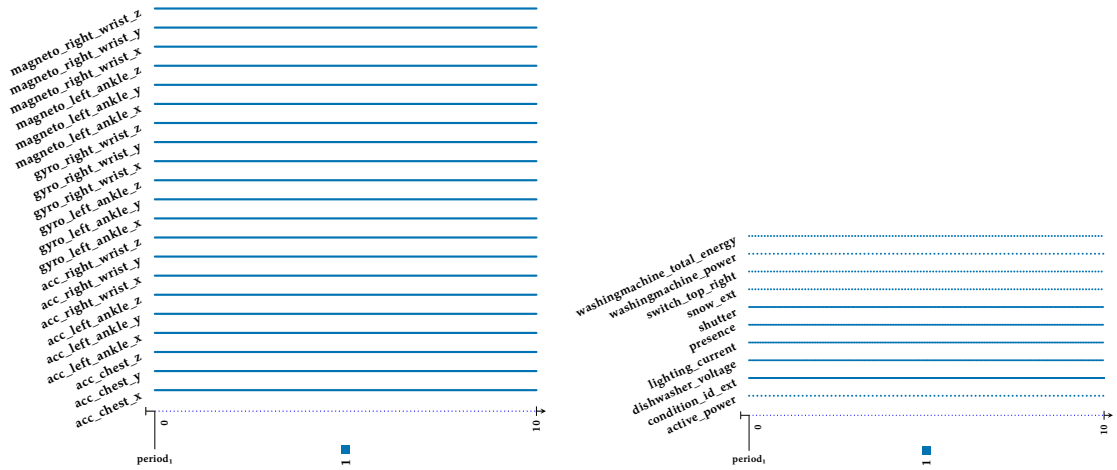
- For the 10 DA, the sequence also contains multiple queries metadata which exploit both of the considered inference channels. Hence, for example, the user u_k's query metadata sequence is denoted by $SQ^{u_k} = SQ_0^{u_k} + SQ_1^{u_k}$. For each sequence, the timestamp of the first period can be chosen as described for the GUs. Likewise, the frequencies FQ are here considered for the attributes A_1 only. The queries are positioned on the emission timeline similarly to the GU archetype.

The generation timeline is considered when generating the sequence of queries metadata. The emission timeline is considered only at the dataset level, to define the order in which queries metadata are processed by RICE-Sy. In the following section, we present and formalize how query metadata sequences are generated.

6.4 Archetype-based generation of query metadata sequences

Sequences of queries metadata are generated considering the two case studies, i.e., mHealth presented in Section 4.1.1, and Orange4Home presented in Section 4.1.2, and the three defined archetypes: the One-time Attacker (OA), the Genuine User (GU), and the Deceptive Attacker

Cette thèse est accessible à l'adresse : https://theses.insa-lyon.fr/publication/2024ISAL0024/these.pdf
© [P. Lachat], [2024], INSA Lyon, tous droits réservés



(a) mHealth case study. (b) Orange4Home case study.

Figure 6.4: Illustrations of sequences generated for a one-time attacker.

(DA) (see Section 6.1.2). Each step of the generation process is formalized using the concepts defined in Section 6.2.

6.4.1 The one-time attacker (OA)

Let us start by considering the simplest archetype corresponding to an attacker performing a single IAISD using a single query (see Definition 6.1.2). Since this archetype does not rely on the concepts of multiple queries, consolidation, regular groups of attributes, or genuine blocks, it's the most suitable starting point to explain and formalize how the generation of query metadata sequences works for our two case studies. For an OA, a single period is defined to generate a sequence. A single block is instantiated within this period, initialized with a malicious status, and is referencing a full partition of the required attributes. Therefore, it produces a single query metadata leading to an inference attack. In the following, we further illustrate with Figure 6.4 how this generation is performed for each of our two case studies.

mHealth The malicious block is initialized as formalized in Algorithm 6.1, by the function *oa_malicious_block_mhealth*. Since data points must be queried for each attribute in a single query metadata, it references a partition with a single set containing all the required attributes, corresponding to A_0 the set of attributes of the inference channel mHealth. The function then randomly selects (denoted by $\in_R$) one of the environment *env* among all the ones associated to this inference channel, denoted by E_0. It randomly selects a time interval during the single period of the sequence and set both *li* and *os* of the block to be equal to this interval. It ensures that this interval is equal or greater than the required duration of mHealth. Based on this malicious block denoted by *blk*, a single query metadata is generated as formalized by the function *oa_sequence_mhealth*. For each attribute in the set *AP* assigned to *blk*, an MKU is created and the frequency at which data points are generated for this attribute is fixed. Each MKU references the environment of *blk* and its overlapping time interval *os*. This set of MKUs forms the query metadata denoted by QM_{Q_1}. *oa_sequence_mhealth* returns a sequence containing QM_{Q_1} only.

Cette thèse est accessible à l'adresse : https://theses.insa-lyon.fr/publication/2024ISAL0024/these.pdf
© [P. Lachat], [2024], INSA Lyon, tous droits réservés

Algorithm 6.1: Generation of a query metadata sequence for the OA archetype & mHealth

Inputs:
- $period_start$ — Start of the single period.
- $period_duration$ — Duration of the single period.
- FQ — Set of frequencies related to each attribute.

Output: A sequence containing a single query metadata exploiting mHealth.

Function *oa_malicious_block_mhealth(period_start, period_duration)*
- $AP \leftarrow \{A_0\}$
- $env \leftarrow env \in_R E_0$ // Randomly select an environment
- $current_period \leftarrow (period_start, period_start + period_duration)$
- // Randomly select the block limit
- $li \leftarrow li \in_R \mathcal{T} : li \, \mathsf{d} \, current_period \ \wedge \ \Delta(li) \geq 2\,\mathrm{s}$
- $os \leftarrow li$
- **return** $\langle AP, env, os, li \rangle$

Function *oa_sequence_mhealth(period_start, period_duration, FQ)*
- $blk \leftarrow oa_malicious_block_mhealth(period_start, period_duration)$
- $QM_{Q_1} \leftarrow \varnothing$
- **foreach** $a \in AP_{blk}$ **do**
  - $QM_{Q_1} \leftarrow QM_{Q_1} \cup \{\langle a, env_{blk}, os_{blk}, fq_a \in FQ \rangle\}$
- **return** $\langle QM_{Q_1} \rangle\}$

We visualize the generated sequence as depicted by Figure 6.4a. For the sake of readability and to keep coherent visualization for future archetypes, the color and number of each query metadata is described as a legend below each block. For instance, the query metadata QM_{Q_1} is represented by a blue square with a rotated label 1. The colored segments corresponds to the MKUs in QM_{Q_1}. They each references one of the attributes on the y-axis for the same time interval on the x-axis. The x-axis corresponds to the generation timeline. The interval of the block from which MKUs are produced is represented has a horizontal dashed blue line. The single period is represented by a vertical solid line at the beginning of the block. In our second case study, the generation must consider the SSW data prerequisite, and thus the generation frequencies associated to attributes.

Orange4Home The malicious block is initialized in a similar fashion as for mHealth, but considering A_1 and E_1. In addition to TSW, we take the SSW constraint of Orange4Home into account. This implies that not all attributes A_1 have to be selected in the block (see Section 4.1.2). AP and li are chosen following Equation 6.4.1, where fq_a denotes the frequency associated to the attribute a. $period_start$ and $period_duration$ are denoted by pe_st and pe_du, respectively, for readability purpose. This equation ensures that considering that the attributes and the block limit are selected so that the quantity of the generated data points during the limit is equal or greater than the required quantity. The function dp_a computes the number of data points associated to MKUs (i.e., that will be here produced from the block). The sequence of query metadata is created following function *oa_sequence_mhealth*.

Cette thèse est accessible à l'adresse : https://theses.insa-lyon.fr/publication/2024ISAL0024/these.pdf
© [P. Lachat], [2024], INSA Lyon, tous droits réservés

$$\exists\, AP \subseteq P \in B_{A_1},\ \exists\, os \in \mathcal{T} : os \,\mathrm{d}\, (pe_st, pe_st + pe_du) \wedge \Delta(os) \geq 15\,\mathrm{s} \ \wedge \underset{\forall\ a \in AP}{\sum} dp_a(li, fq_a) \geq 20 \tag{6.4.1}$$

Figure 6.4b depicts such a sequence, where we can observe that, similarly to the visualization of mHealth, a single query metadata, denoted by QM_{Q_1}, is generated and is affected the blue color. Here, each colored segment also corresponds to an MKU within QM_1. They all have the same interval, which corresponds to the interval of the block from which they originate. Yet, we see on the y-axis that they reference only a subset of all available attributes in Orange4Home (see Section 4.1.2 in Chapter 4). Moreover, the MKUs represent when the selected data points are generated w.r.t. the frequency associated to each attribute.

Discussion As described in Definition 6.1.2, we assume that the OA archetype performs a single attack via a single query metadata. However, an attacker may perform multiple IAISDs, using each time a single query. Such a situation can be generated in our simulator by either increasing the number of periods associated to the OA archetype. It can also be achieved by decreasing the minimum number of queries metadata produced from a block to 1 for the DA archetype (see function *da_malicious_block_mhealth* in Algorithm 6.4). While the second approach produce sequences with different approaches of attacks (i.e., a single query metadata, multiple ones, etc.), the former produce a sequence focused completely on attacks. Moreover, as explained in Section 6.1.1, we have ignored complex queries, i.e., composed of sub-queries such as `SELECT ... UNION SELECT ...`, for the sake of simplicity. Yet, a user following the OA archetype may issue such a query. For instance, the block blk_2 in Figure 6.2 could produce a single query metadata $QM_{Q_{3\&4}}$ which contains the MKUs of both QM_{Q_3} and QM_{Q_4}. To do so, we would need to update the theoretical grammar depicted by Figure 4.5a in Chapter 4. The challenge would be to use the function *da_malicious_block_mhealth* to obtain a malicious block, to generate the associated queries metadata, and to merge them as a single one. In the next section, we focus on the second archetype, i.e., the genuine employee.

6.4.2 The genuine user (GU)

Let us continue by considering the GU archetype issuing multiple queries without performing a single IAISD. This archetype is the most suitable to introduce how the concepts of multiple periods, genuine blocks, and regular groups of attributes are combined to generate sequences of regular queries metadata. In our setting, the GU can have several regular groups of attributes that they regularly query, among a few queries selecting other attributes. Multiple periods are defined and each contains a few genuine blocks where most reference one of the regular group of attributes. To ensure that regular groups of attributes are selected, the number of genuine blocks defined in a period must at least equal to the number of defined groups. Each genuine block produces here a single query metadata. Since a block leads to a query metadata, that multiple blocks are defined for each period, then the resulting sequence contains multiple queries metadata. In the following, we further illustrate how this generation works for each case study.

mHealth

Considering mHealth, the genuine block is initialized as formalized in Algorithm 6.2, by the function *gu_genuine_block_mhealth*. Each block is generated within a period, and in each period each defined regular group of attributes must be selected by a query metadata in a

Cette thèse est accessible à l'adresse : https://theses.insa-lyon.fr/publication/2024ISAL0024/these.pdf
© [P. Lachat], [2024], INSA Lyon, tous droits réservés

Algorithm 6.2: Part 1: Generation of a sequence for the GU archetype & mHealth

Inputs: *current_period* (*cu_pe*), *period_duration* (*pe_du*), *attributes*

Output: A genuine block.

1 **Function** *gu_genuine_block_mhealth(cu_pe, pe_du, attributes)*
2 $AP \leftarrow \{attributes\}$
// Randomly choose a subset of attributes in case no regular group of attributes is provided
3 **if** $attributes = A_0$ **then** $AP \leftarrow \{p\}, P \in_R B_{attributes}, p \in_R P$
4 $li \leftarrow \begin{cases} \text{chose } li \text{ d } (cu_pe, cu_pe + pe_du) \text{ s.t. } \Delta(li) < 2\,\text{s} \wedge \\ \nexists\, li' \in \mathcal{T} : li' \text{ d } (cu_pe, cu_pe + pe_du) \ \wedge \ li \circ li' \end{cases}$
5 $env \leftarrow env \in_R E_0$ // Randomly select an environment
6 $os \leftarrow \emptyset$
7 **return** $\langle AP, env, os, li \rangle$

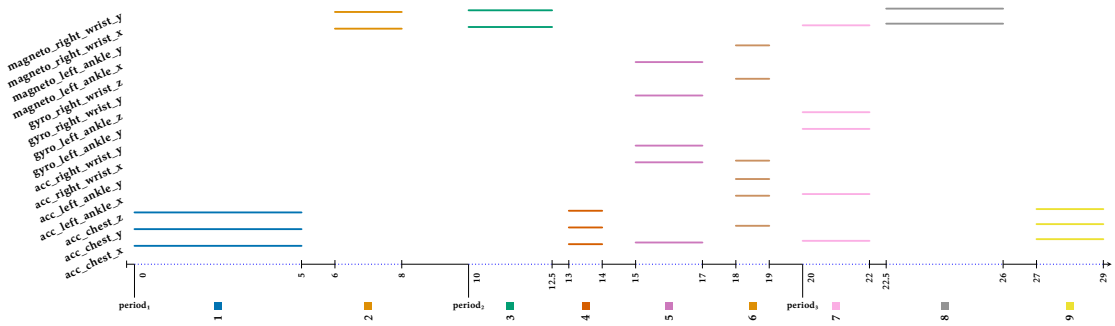


Figure 6.5: Sequence generated considering the mHealth case study.

distinct block, to be sure there is no possible inference situation. Otherwise, the genuine block purpose is to generate a query metadata which randomly selects (denoted by $\subset_R$) a sub-partition of the mHealth attributes. Then, the remainder members (i.e., *env*, *li*, and *os*) of the block are initialized following the same logic than in Algorithm 6.1. However, here the interval of the block, denoted by *li*, is chosen within the interval of the currently considered period, denoted by (*current_period*, *current_period* + *period_duration*). *li* is chosen so that it does not overlap another block in the same period. The query metadata sequence for the GU archetype is generated as formalized by the function *gu_sequence_mhealth* in Algorithm 6.3. For each considered period, it randomly determines the number of genuine blocks to initialize using a uniform distribution, denoted by U. This number is chosen to ensure that there is at least one block for each defined regular group of attributes, since they should be selected in each period. A maximum number of blocks is provided as a parameter, s.t. $max_block \geq |regular_groups_attributes|$, if the number of randomly selected block is greater than the number of regular groups of attributes, thus combining regularity and diversity of selected attributes. For each block, the corresponding query metadata is produced following the same approach as formalized in Algorithm 6.1. Finally, the sequence is built by concatenating all created queries metadata.

We can observe such a sequence depicted in Figure 6.5. Here, for this example, we have defined: three periods; a maximum number of four blocks; the two following regular groups of attributes {acc_chest_x, acc_chest_y, acc_chest_z} and {magneto_right_wrist_x, magneto_right_

Cette thèse est accessible à l'adresse : https://theses.insa-lyon.fr/publication/2024ISAL0024/these.pdf
© [P. Lachat], [2024], INSA Lyon, tous droits réservés

Algorithm 6.3: Part 2: Generation of a sequence for the GU archetype & mHealth

Inputs: *period_start* (*pe_st*), *number_periods* (*nu_pe*), *period_duration* (*pe_du*), *max_block* (*ma_bl*), *regular_groups_attributes* (*rga*)

Output: A sequence of queries metadata which are not leading to an inference attack.

1 **Function** *gu_sequence_mhealth(pe_st, nu_pe, pe_du, ma_bl, rga)*
2 $SQ \leftarrow \emptyset$
3 $cu_pe \leftarrow pe_st$
4 **foreach** $1, \dots, nu_pe$ **do**
5 $idx_group \leftarrow 1$
// Randomly choose the number of blocks using a uniform distribution
6 $number_block \leftarrow U(|rga|, ma_bl)$
7 $cu_pe \leftarrow cu_pe + pe_du + 1$
8 **foreach** $1, \dots, number_block$ **do**
9 **if** $idx_group \leq |rga|$ **then**
// Define a block per regular group of attributes
10 $idx_group \leftarrow idx_group + 1$
// Build a block which reference the regular group of attributes associated to the indice *idx_group*
11 $blk \leftarrow gu_genuine_block_mhealth(cu_pe, pe_du, rga_{idx_group})$
12 **else**
// Define more blocks when all groups are covered
13 $blk \leftarrow gu_genuine_block_mhealth(cu_pe, pe_du, A_0)$
14 $QM \leftarrow \emptyset$
15 **foreach** $a \in AP_{blk}$ **do**
16 $QM \leftarrow QM \cup \{\langle a, env_{blk}, li_{blk}, fq_a \in FQ \rangle\}$
17 $SQ \leftarrow SQ \cup QM$
18 **return** SQ

wrist_y}. We observe that a total of nine queries metadata are generated. In the first period, only two genuine blocks are defined: one for each group. In the second period, there are four blocks, hence the first and second one are selecting the regular groups of attributes, whereas the two last blocks select random subsets of attributes. Finally, the last period contains three blocks where the first selects random attributes and the two last select the regular groups of attributes. Therefore, we observe that all the blocks do not overlap. None of the produced queries metadata is exploiting the mHealth inference channel as the regular groups of attributes are sub-partitions of the total regular group of attributes composing the inference channel. In the following, we explain how such a query metadata sequence is built for Orange4Home.

Orange4Home

Considering our second case study, the genuine blocks are initialized following the logic of the function *gu_genuine_block_mhealth* in Algorithm 6.2. However, similarly to the OA query metadata sequence, we consider the frequencies FQ associated to each attribute. Indeed, to enforce the genuine status of blocks, the function chooses the time interval li so that, considering the attributes in AP and their frequencies, the block duration, denoted by $\Delta(li)$, is smaller than the expected duration or the number of data points is smaller than the expected quantity. Hence, instead of the clause at line 4 in Algorithm 6.2, the genuine blocks for Orange4Home

Cette thèse est accessible à l'adresse : https://theses.insa-lyon.fr/publication/2024ISAL0024/these.pdf
© [P. Lachat], [2024], INSA Lyon, tous droits réservés

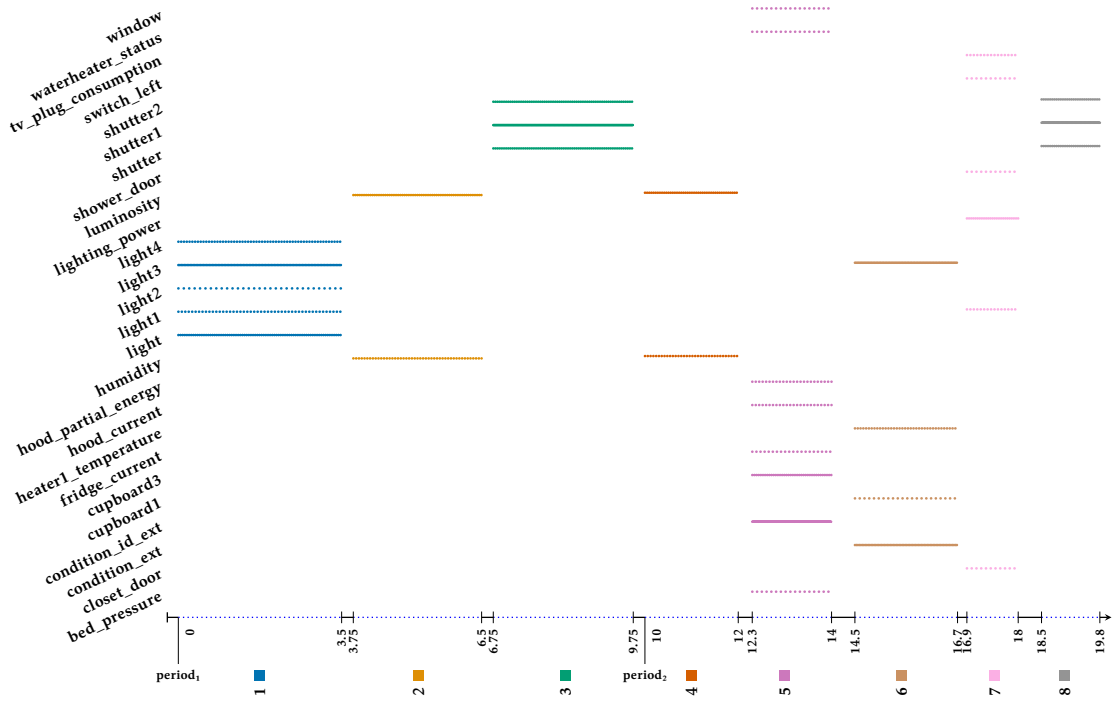


Figure 6.6: Sequence generated considering the Orange4Home case study.

must be initialized with an interval satisfying the clause of Equation 6.4.2. Finally, the query metadata sequence for this case study is built by following the logic defined by the function *gu_sequence_mhealth* in Algorithm 6.3. Moreover, the queries metadata produced from the genuine blocks are defined using the approach presented in Section 13, i.e., where the SSW constraint of the MKUs reference the timestamp of the first data point, in the block interval, and the frequency associated to an attribute.

$$\exists AP \subseteq P \in B_{A_1},\ \exists\, li \in \mathcal{T} : li \,\mathsf{d}\, (cu_pe, cu_pe + pe_du) \wedge \left(\Delta(li) < 15\,\mathrm{s} \ \vee \sum_{\forall a \in AP} dp_a(li, fq_a) < 20\right) \tag{6.4.2}$$

An example of a query metadata sequence generated for Orange4Home is depicted in Figure 6.6. Here, we have defined: two periods; a maximum number of six blocks; one regular group of attributes: {humidity, luminosity}. We observe that a total of eight queries metadata are generated. In the first period, three queries metadata are generated, where the second one is selecting the regular group of attributes. In the second period, five queries metadata are generated, and the group is selected by the first query metadata. Hence, outside of the those two queries metadata, all the other select random attributes without exploiting the Orange4Home inference channel.

Discussion As described in Definition 6.1.1, we assume that users following the GU archetype never performs any inference attack. Yet, a user with a genuine intent may query sensor data in such a way that, from the perspective of RICE-Sy, they exploit an inference channel, even if this

Cette thèse est accessible à l'adresse : https://theses.insa-lyon.fr/publication/2024ISAL0024/these.pdf
© [P. Lachat], [2024], INSA Lyon, tous droits réservés

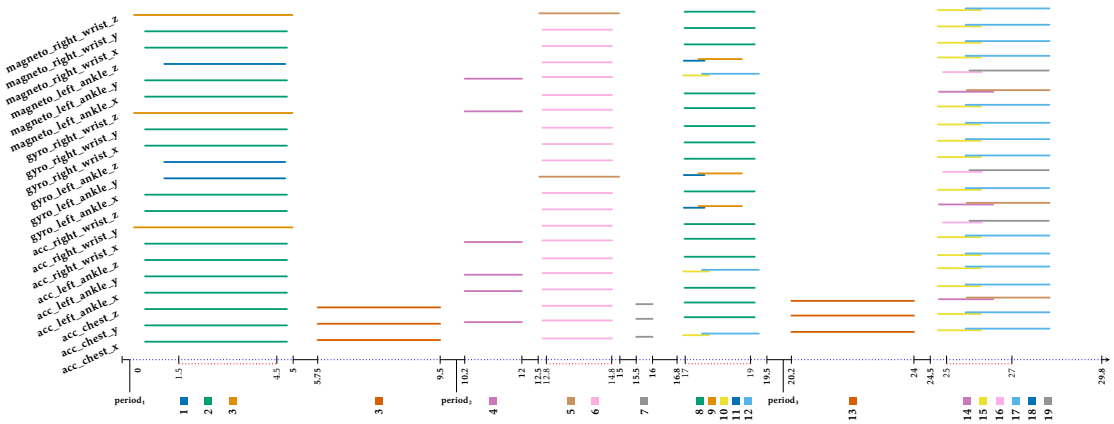


Figure 6.7: Sequence generated considering the mHealth case study.

user isn't aware of the existence of the inference channel. This specific situation is problematic since it leads to detect the query issued by a truly genuine user as being an inference attack. However, this problem is not specific to the detection system we propose, indeed it has been identified in other works [150, 144]. Since we can not determine the true intent of users from their queries only, this stays an open problem. We chose to ignore this case by enforcing the genuine status of blocks generated for the GU archetype. Moreover, we assume that this archetype cannot perform consolidation. While this assumption is introduced here to gradually present how sequences are generated, from the simplest to the most complicated case, the GU archetype may contain consolidation. Indeed, such a user can use instant queries that can consolidate themselves. The query metadata produce by a genuine block can be split using the function *da_consolidable_queries* in Algorithm 6.5. In the next section, we focus on the last archetype.

6.4.3 The deceptive attacker (DA)

Let us consider the most complex archetype, see Definition 6.1.3, corresponding to an attacker performing multiple IAISDs, using at least two queries metadata for each attack. This attacker issues queries to work like a genuine user and issues the necessary queries to exploit an inference channel. Consequently this archetype requires both genuine blocks, that represents a regular activity, and malicious blocks leading to an inference. Similarly to the GU archetype, the periods and the regular groups of attributes are leveraged to enforce a regular behavior for the genuine blocks. Within a sequence of queries metadata of the DA archetype, at least one malicious block is initialized in order to simulate an attack. Those blocks reference multiple sets of attributes (i.e., $|AP| > 1$). In the following, we illustrate how the generation of a query metadata sequence is performed considering our two case studies.

mHealth

Considering mHealth first, the malicious block is here initialized as shown in the function *da_malicious_block_mhealth* in Algorithm 6.4. For each malicious block, between two queries metadata up to a provided maximum is randomly chosen using a uniform distribution. Then, the attributes of mHealth are partitioned into one subset per query metadata to produce. The block interval li, as well as the environment env, are initialized following the logic defined by the function *ota_malicious_block_mhealth*. Since multiple queries metadata are produced, the

Cette thèse est accessible à l'adresse : https://theses.insa-lyon.fr/publication/2024ISAL0024/these.pdf
© [P. Lachat], [2024], INSA Lyon, tous droits réservés

Algorithm 6.4: Part 1: Generation of a sequence for the DA archetype & mHealth

Inputs: *period_start* (*pe_st*), *period_duration* (*pe_du*),
max_query (*ma_qu*) Maximum number of queries metadata produced by a malicious block.

Output: A malicious block.

Function *da_malicious_block_mhealth(period_start, period_duration, ma_qu)*
  // Randomly choose the number of queries metadata.
  $number_query \leftarrow U(2, ma_qu)$
  $AP \leftarrow$ partition A_0 into $number_query$ subsets
  ... // Initializes *env* and *li* similarly to Algorithm 6.1
  $os \leftarrow$ randomly choose $os \sqsubset li$ s.t. $\Delta(os) \geq 2\,s$
  return $\langle AP, env, os, li \rangle$

Algorithm 6.5: Part 2: Generation of a sequence for the DA archetype & mHealth

Inputs:
- QMs — Queries metadata to split.
- max_split (*ma_sp*) — Maximum number of split per query metadata.
- P_{cons} — Probability that a block produce consolidable queries metadata.

Output: Queries metadata splitted to simulate consolidation.

Function *da_consolidable_queries(QMs, max_split, P_{con})*
  $consolidable_QMs \leftarrow \varnothing$
  // Randomly choose the number of simulated consolidation
  $number_consolidation \leftarrow U(1, max_split)$
  foreach $QM \in QMs$ **do**
    $splited_QMs \leftarrow QM$
    if $rand() \geq P_{cons}$ **then**
      $splited_QMs \leftarrow$ split QM in $number_consolidation$ new subsets of MKUs s.t. they can all be consolidated into QM
    $consolidable_QMs \leftarrow consolidable_QMs \cup splited_QMs$
  return $consolidable_QMs$

malicious block ensures that the produced MKUs are overlapping during the common time interval *os*, having a duration defined by the TSW constraint.

Since we aim to evaluate how the consolidation of users' knowledge is performed by RICE-Sy, we need to control the quantity of consolidation simulated in a query metadata sequence. The queries metadata produced from a malicious block can be *split* in order to simulate consolidation. The more we want to simulate consolidation, the more we will split queries metadata. To do so, as formalize by the function *da_consolidation_queries* in Algorithm 6.5, the number of consolidation to simulate is randomly chosen following a uniform distribution. At least a single consolidation is simulated, up to a given maximum number. Then, each provided query metadata is split in the chosen number of new queries metadata. For instance, let consider the query metadata QM. If the number of consolidation is set to three, then QM is split into three queries metadata: QM_1, QM_2, and QM_3, such that the consolidation of their MKUs results in the MKUs

Cette thèse est accessible à l'adresse : https://theses.insa-lyon.fr/publication/2024ISAL0024/these.pdf
© [P. Lachat], [2024], INSA Lyon, tous droits réservés

of QM. To provide diversity among the queries metadata produced from a malicious block, each queries metadata have an equal probability to be split. The function may split a subset of the provided queries metadata.

The query metadata sequence for the DA archetype is generated following the formalization depicted by the function *da_sequence_mhealth* in Algorithm 6.6. Two distinct set of queries metadata are returned: the ones originating from genuine blocks, and the ones produced by malicious blocks. This is required to correctly order the queries metadata at the dataset level, see Section 6.3. For each period the number of blocks is chosen to guarantee that there is at least one more block than the given regular groups of attributes. It guarantees that there is one genuine blocks per provided regular groups of attribute. Once all groups have been referenced, the remaining blocks to generate are either genuine or malicious. Hence, more genuine blocks are generated according to a provided probability P_{mal} if the sequence already contains at least one malicious block. Otherwise, malicious blocks are generated and the resulting queries metadata are split according to the provided probability P_{cons}. Finally, the query metadata sequence of the DA archetype is built by combining queries metadata originating from all blocks, i.e., $SQ = SQ_{genuine} + SQ_{malicious}$.

We can observe such a sequence depicted in Figure 6.7. Here, for this example, we have defined:

- Three periods
- A maximum number of four blocks
- A single regular group of attributes: $\{acc_chest_x, acc_chest_y, acc_chest_z\}$
- A maximum number of three queries metadata produced in a block
- A probability of 0.75 that a block is malicious
- A probability of 0.5 that block, and each produced queries metadata are split
- A maximum number of three split performed for each query metadata

We observe that a total of 19 generated queries metadata are depicted in this plot. Note that the used color palette contains only ten distincts colors, hence you can observe here that some colors are shared by two queries metadata (e.g., QM_1 and QM_{11}), yet their are independent from each other. We notice that in each period there is a genuine block generated in order to select the single defined regular group of attributes. Hence, in the first period, we see that two blocks are generated. The first one is malicious and produces the maximum number of queries metadata (i.e., three in this example) which are not split. The second period contains four blocks. The second and last blocks are malicious and produce two and three queries metadata, respectively. The last block of this period simulates the consolidation of user's knowledge by splitting two of the queries metadata into four queries metadata. This last block produces a total of five queries metadata. Finally, the last period contains two blocks where the second produces a total of five queries metadata. The query metadata QM_{19} is produced and not split, whereas the queries metadata QM_{15} & QM_{16} and QM_{17} & QM_{18} simulate a split that will required consolidation. We observe that none of the blocks are overlapping other blocks. Consequently, none of the produced queries metadata are exploiting the mHealth inference channel. In the following, we explain how the query metadata sequence is built for Orange4Home.

Cette thèse est accessible à l'adresse : https://theses.insa-lyon.fr/publication/2024ISAL0024/these.pdf
© [P. Lachat], [2024], INSA Lyon, tous droits réservés

Algorithm 6.6: Part 3: Generation of a sequence for the DA archetype & mHealth

Inputs: $period_start\,(pe_st)$, $number_periods\,(nu_pe)$, $period_duration\,(pe_du)$, $max_block\,(ma_bl)$, $regular_groups_attributes\,(rga)$, $max_query\,(ma_qu)$, $max_split\,(ma_sp)$, P_{cons}, P_{mal} Probability that a block is malicious.

Output: A query metadata sequence related to the genuine blocks and another one related to the malicious blocks.

```
1  Function da_sequence_mhealth(pe_st, nu_pe, pe_du, ma_bl, rga, ma_qu, P_mal, P_cons, ma_sp)
2      cu_pe ← pe_st
3      has_attack ← ⊥
4      SQ_genuine, SQ_malicious ← ∅, ∅
5      foreach 1, ..., nu_pe do
6          idx_group ← 1
7          number_block ← U(|rga| + 1, ma_bl)
8          cu_pe ← cu_pe + pe_du + 1
9          foreach 1, ..., number_block do
               // Create a genuine block if there is still regular group of
                  attributes to cover or
               // if a malicious block has already been created for the current
                  period
10             if idx_group ≤ |rga| ∨ (rand() < P_mal ∧ has_attack) then
11                 attributes ← A_0
12                 if idx_group ≤ |rga| then
13                     idx_group ← idx_group + 1
14                     attributes ← rga_idx_group
                   // Algorithm 6.2
15                 blk ← gu_genuine_block_mhealth(cu_pe, pe_du, attributes)
16                 ... // Creates a query metadata QM similarly to Algorithm 6.1
17                 SQ_genuine ← SQ_genuine ∪ QM
18             else
19                 new_QMs ← ∅
20                 has_attack ← ⊤
21                 blk ← da_malicious_block_mhealth(cu_pe, pe_du, ma_qu)
22                 foreach attributes ∈ AP_blk do
23                     ... // Like line 16, but considering attributes
24                     new_QMs ← new_QMs ∪ QM
25                 if rand() ≥ P_cons then
26                     new_QMs ← da_consolidable_queries(new_QMs, ma_sp, P_cons)
27                 SQ_malicious ← SQ_malicious ∪ new_QMs
28     return SQ_genuine, SQ_malicious
```

Cette thèse est accessible à l'adresse : https://theses.insa-lyon.fr/publication/2024ISAL0024/these.pdf
© [P. Lachat], [2024], INSA Lyon, tous droits réservés

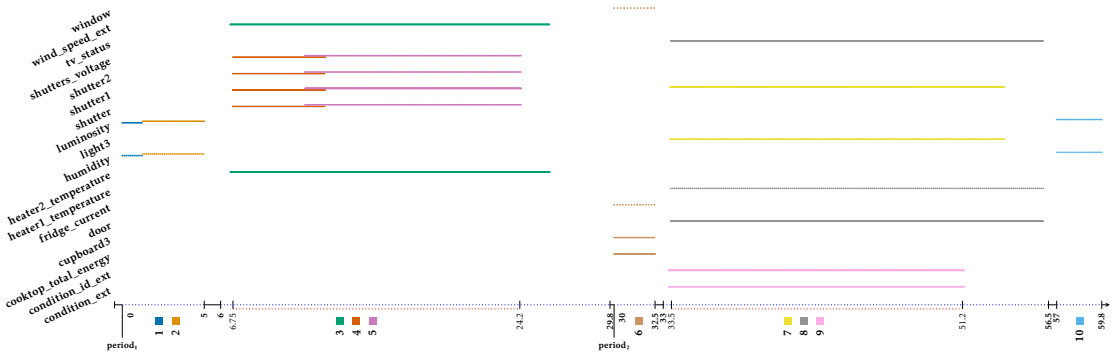


Figure 6.8: Sequence generated considering the Orange4Home case study.

Orange4Home

Considering our second case study, the sequence of queries metadata is generated with the function *da_sequence_mhealth* of Algorithm 6.5, presented in the previous section. The initialization of genuine blocks and the malicious blocks for Orange4Home follows the same logic than the GU archetype (see Section 18) and the OA archetype respectively (see Section 13). Similarly to mHealth, the malicious blocks set the interval *os* to a non-null value. Therefore, assuming that the set of attributes *AP* to reference and the block interval *li* are initialized w.r.t. the Equation 6.4.1. Then, *os* is randomly chosen according to the Equation 6.4.3.

$$\exists\, os \in \mathcal{T} : os \,\mathrm{d}\, li \;\wedge \Delta(os) \geq 15\,\mathrm{s} \;\wedge \sum_{\forall a \in AP} dp_a(os, fq_a) \geq 20 \tag{6.4.3}$$

Such a query metadata sequence is depicted in Figure 6.8. Here, for this example, we have defined: two periods; a single regular group of attributes: {humidity, luminosity}; the remaining settings are similar to the mHealth visualization for the deceptive attacker archetype. We observe that a total of ten queries metadata are generated. We observe that for the first and second period, the first and last block, respectively, select the defined regular group of attributes. In the first period the second block is malicious and produces two queries metadata which are further split in two, thus resulting in four queries metadata. In the second period, the first block is genuine and, since the last block already selects the defined regular group of attributes, it selects random attributes. The second block is malicious and produces three queries metadata which are not split. Consequently, we observe that here also the sequence of queries metadata contains both queries metadata which exploits and inference channel and queries metadata originating from genuine blocks. In the following section, we present the parameters values associated to each archetype.

6.5 Parameters value of query metadata sequences

We provide to the generator different parameters values, according to the archetype for which we want to generate query metadata sequences. We distinguish two categories of parameters: (i) The ones that are set similarly for all the archetypes in a dataset: The duration of periods in a sequence (*period_duration*) and the time at which the first period starts (*period_start*) on the generation timeline. (ii) The ones that are set taking into account the specificity of each archetype: The number of periods defined on the generation timeline (*number_periods*); the maximum number

Cette thèse est accessible à l'adresse : https://theses.insa-lyon.fr/publication/2024ISAL0024/these.pdf
© [P. Lachat], [2024], INSA Lyon, tous droits réservés

of blocks defined in a period (max_block); the maximum number of queries metadata produced from a block (max_query); the regular groups of attributes ($regular_groups_attributes$); the maximum number of time queries metadata are split (max_split); the probability that a block is malicious (P_{mal}); and the probability that a block simulates consolidation (P_{cons}).

In the following, we describe the values of the latter category of parameters according to each archetype:

- *One-time Attacker (OA)*: This archetype issues a single query metadata. The generator has to create a single block over a single period to produce a single query metadata ($number_periods = 1$, $max_block = 1$, $max_query = 1$). It performs an inference attack by itself ($P_{mal} = 1$), without performing any consolidation ($P_{cons} = 0$, $max_split = 0$). No regular group of attributes is defined since only a single period is considered ($regular_groups_attributes = \emptyset$).
- *Genuine User (GU)*: This archetype issues multiple queries following the same query model. The generator produces queries metadata over multiple periods ($number_periods > 1$). In each period, at least one block is generated ($max_block \geq 1$). The nature of this archetype leads to regular groups of attributes ($|regular_groups_attributes| > 1$) being regularly selected. The number of periods, of blocks, and the groups are set w.r.t. the need of each evaluation. We describe those values in the next chapter. We assume that, since the GU archetype knows the individual's data they need, they issue periodically a single query metadata asking for the same patterns of data (i.e., the same set of attributes values during a time interval). Consequently, each block contains only a single query metadata corresponding to this pattern of data ($max_query = 1$) without performing any consolidation ($P_{cons} = 0$, $max_split = 0$). The genuine nature of this archetype implies that no queries metadata lead to an inference attack ($P_{mal} = 0$).
- *Deceptive Attacker (DA)*: This last archetype issues regular queries to create noise hiding attacks. While the query metadata sequence of this archetype is composed of both genuine and malicious blocks, we consider that they mostly aim to perform attacks ($P_{mal} = 0.75$). The DA archetype performs attacks by issuing multiple queries metadata ($max_query \geq 2$) while performing potential consolidation of their user's knowledge ($P_{cons} = 0.25$, $max_split \geq 2$). Here, the number of periods, of blocks, maximum queries metadata, maximum split, and the groups are set on the basis of each evaluation.

Now that we have described how the sequences contained in a dataset are generated, let us explain how queries metadata are dispatched on the emission timeline. To do so, we leverage the example depicted in Figure 6.9. On the left side, for the sake of simplicity, we focus on the two first periods from Figure 6.5. As explained in Section 6.3, we assume that queries select data points that are already generated. Hence, the corresponding queries metadata can be placed on the emission timeline at a time equal or greater than the last selected data point in the query. In our example, all the blocks are genuine and as explained in the GU archetype (see Section 6.1.2). We assume that while most queries metadata are extracted from periodic queries, some can be related to instant queries. The first kind are emitted at the time they are generated. For example, we observe that the queries metadata QM_{Q_2}, QM_{Q_3}, QM_{Q_5}, and QM_{Q_6} are emitted in the order and at the time they are generated, thus representing periodic queries. The instant queries are here represented by QM_{Q_1} and QM_{Q_4} which model that the queries Q_1 and Q_4 select data points not at soon they are generated, but in a later time. For the DA archetype, the approach is similar for the queries metadata in the genuine sequence (i.e., $SQ_{genuine}$). For the malicious sequence (i.e., $SQ_{malicious}$), the queries metadata are randomly dispatched over the emission timeline similarly to the periodic queries of our example.

Cette thèse est accessible à l'adresse : https://theses.insa-lyon.fr/publication/2024ISAL0024/these.pdf
© [P. Lachat], [2024], INSA Lyon, tous droits réservés

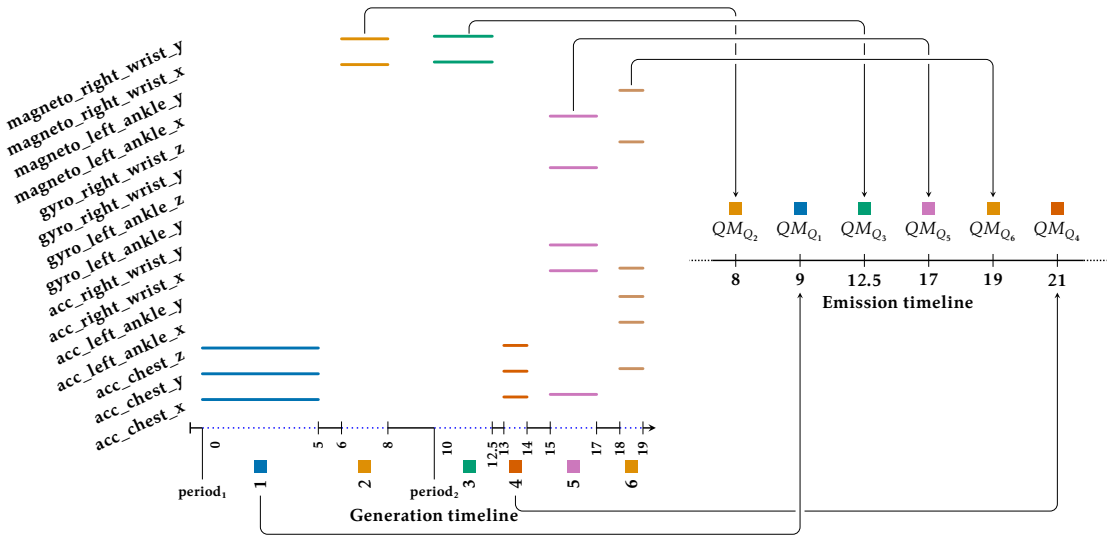


Figure 6.9: Example of queries metadata dispatched on the emission timeline.

Our proposed generator enables the creation of datasets suitable for the evaluation of RICE-Sy. This contribution produces sequences of queries metadata associated to multiple users, by considering both our two case studies mHealth and Orange4Home as inference channels, and three archetypes of querying behaviors which guide the creation of realistic queries metadata. In the following section, we discuss how our contribution can be improved to consider more divers query metadata sequences.

6.6 Discussion

We assume that all users follow a single archetype which does not change in time. While this assumption is strong, the change of archetype has no impact on the evaluation of RICE-Sy. The three archetypes have been defined in order to guide the generation of relevant query metadata sequences. Considering the DA archetype, one may modify the generator of queries metadata so that the parameters P_{mal} and P_{cons} change in time, for instance between each period. A sequence may be generated for a user which starts by following a GU archetype (i.e., $P_{mal} = 0$ and $P_{cons} = 0$), then switch slowly to the DA archetype (i.e., by incrementing P_{mal} and P_{cons} for a given or random value). To do so, the generator must receive a parameter describing how the values of those parameters change during the generation. While our assumption that users follow a single archetype during the whole generated sequence is quite strong, to the best of our knowledge, no study of how users change their behavior in time has been proposed in the scientific literature. Moreover, it does impact the evaluation of RICE-Sy, since the detection is not impacted by the order in which a user issues queries to $\mathtt{DB}_{\mathtt{sen}}$. Regardless of the changing querying behaviors, RICE-Sy has to detect IAISDs w.r.t. to its known inference channels.

6.7 Conclusion

Our third contribution, i.e., the query metadata generator, formalizes how to generate queries metadata used to evaluate RICE-Sy. We have observed that the existing datasets either do not contain queries selecting sensor data or do not contain queries issued to perform an inference

Cette thèse est accessible à l'adresse : https://theses.insa-lyon.fr/publication/2024ISAL0024/these.pdf
© [P. Lachat], [2024], INSA Lyon, tous droits réservés

attack. Since exploiting the later type of datasets implies creating genuine and malicious queries, as well as implementing the knowledge extraction module to obtain queries representation, we have decided to directly generate suitable queries metadata. To produce realistic sequences of queries, we have identified querying behaviors by analyzing inference attack strategies and the nature of sensor databases from which sensor data are queried. Based on those behaviors, we have defined the three querying archetypes that users follow: the *one-time attacker*, the *genuine user*, and the *deceptive attacker*. In addition to those archetypes, we have introduced concepts that allow us to control the generated queries metadata. Based on them, we demonstrate how to generate query metadata sequences for each archetype. Thus, we are able to generate datasets that met our requirements:

(I) *Queries metadata model queries issued by distinct users*.
The parameters associated to the generation of a dataset allow the specification of the number of users for which sequences must be generated, as well as the archetype associated to each user.

(II) *Both attackers and non-attackers are represented among users*.
Thanks to the concept of *block*, queries metadata that exploit the inference channel of the mHealth case study, the Orange4Home case study, or none. By preventing overlapping blocks, we guarantee that the produced queries metadata follow the defined status. For instance, two genuine blocks cannot produce queries metadata that, if combined, enable exploiting an inference channel.

(III) *Users selecting sensor data via different approaches*.
Furthermore, the block allows controlling that a single query metadata, multiple queries metadata with, or without consolidation, or any of those combinations are produced. This concept guarantees that this diversity of approaches preserve the status, i.e., that multiple consolidable queries metadata or a single query metadata lead to an attack if the block is malicious.

(IV) *Approaches that reflect realistic querying behaviors*.
Finally, with the concepts of *regular groups of attributes*; *periods*, and *generation & emission timelines*, we can control how many queries metadata are generated in a sequence, as well as ensuring that such a sequence follows the defined archetypes, thus defining realistic behaviors in addition to the genuine and/or malicious aspect of queries metadata.

In the following chapter, we leverage our query metadata generator to evaluate the detection capabilities of RICE-Sy.

Cette thèse est accessible à l'adresse : https://theses.insa-lyon.fr/publication/2024ISAL0024/these.pdf
© [P. Lachat], [2024], INSA Lyon, tous droits réservés

Chapter 7

Evaluation of the conceptual optimizations

Based on the generator presented in the previous chapter, we leverage in this chapter the generated queries metadata sequences to evaluate RICE-M based inference detection System (RICE-Sy). Users aim to regularly query sensor data when they are produced, in order to update their knowledge about individuals (e.g., the performed human activities). For instance, the employees of an external company frequently query the sensor data of the service customers to calibrate sensors based on the freshest generated data. To preserve the service's Quality of Service (QS), the data controller has to let authorized users (i.e., the employees) access data efficiently while preserving its customers' privacy. Moreover, by querying sensor data, users make their Consolidated Query History Log (ConsQHL) grow, which impacts the quantity of user's knowledge that must be processed to detect an Inference Attack Involving Sensor Data (IAISD) for each new query metadata. Consequently, in the following evaluations, we consider as our two metrics: the detection time of a query metadata and the size of the ConsQHL. In this chapter, we present four evaluations which enable to observe how the metrics are impacted by: (i) our conceptual optimizations and the storage of consolidated MKUs in the ConsQHL, to assess how they improve the detection time of RICE-Sy (ii) the query emission order, to determine if RICE-Sy is sensitive to the order in which queries metadata are processed (iii) different consolidation settings, to assess both the impact of the consolidation on the detection time & the benefit of saving consolidated MKUs in the ConsQHL, and (iv) different quantities of users issuing a fixed total amount of queries, to examine how the volume of users impact the detection time.

This chapter is structured as follows: we start by presenting the two metrics considered to evaluating RICE-Sy in Section 7.1. We provide in Section 7.2, a technical overview of the prototype of RICE-Sy, and the generator. In Section 7.4, we present the objective of each evaluation, as well as the settings of the measurements and the datasets. Then, we present the obtained results. Finally, we discuss the observation and conclusion obtained in all the evaluations in Section 7.5, and conclude this chapter in Section 7.6.

7.1 Metrics: Detection overhead and ConsQHL size

To protect individuals, RICE-Sy delays the time at which users receive their answer by the time required to detect if a query metadata leads or not to an IAISD. In the following, we refer to this delay as the *overhead* of RICE-Sy. As depicted in Figure 7.1, we divide the query answer time of the data controller in four phases: the time taken by $\mathtt{DB}_{\mathtt{sen}}$ to forward a new query to the knowledge extraction module ❶, the extraction time of the query metadata ❷, the detection time of RICE-Sy ❸, and the time taken by the data controller to decide how to control the query based

Cette thèse est accessible à l'adresse : https://theses.insa-lyon.fr/publication/2024ISAL0024/these.pdf
© [P. Lachat], [2024], INSA Lyon, tous droits réservés

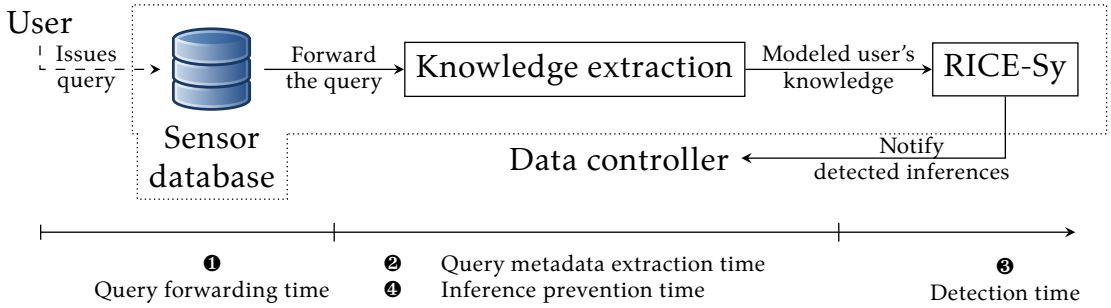


Figure 7.1: Temporal sections of the data controller workflow.

on the output of RICE-Sy ❹. The overhead of RICE-Sy corresponds to the phase ❸. It results from the sum of the computation time of each of its modules.

As depicted in Figure 5.9, when a query metadata is processed, both the consolidation and detection module receive as input a set of MKUs originating from this query metadata and from the ConsQHL. The overhead of the detection depends then on the quantity of MKUs in this set. When considering RICE-Sy without the filtering modules, the whole ConsQHL is considered for each query metadata. With the filtering modules, their respective computation time depends on the number of MKUs stored in the ConsQHL. Consequently, the *size* of the ConsQHL impacts the overhead in both settings, more than the number of MKUs in the processed query metadata.

Therefore, to observe at which scale detecting IAISDs via our system deteriorates the quality of service of the protected sensor database, we consider the following two metrics: the overhead of RICE-Sy (in seconds) and the size of the ConsQHL (in number of stored MKUs). In the following section, we present the implementation of both RICE-Sy and our generator, that we leverage to perform those measurements.

7.2 Implementation of RICE-Sy & the Generator

The prototype of RICE-Sy is implemented as a Python package. It relies on the probabilistic logic programming language ProbLog 2 [48] to model and reason on the inference channels modeled from our case studies: mHealth and Orange4Home. This language has logical mechanisms similar to the Prolog language to reason about truth values and allows the definition of probabilistic distributions over atoms. Besides its capability to model logical goals, the choice of ProbLog is motivated by the possibility of modeling the inferrable knowledge of our inference channels (e.g., the random variable X_i using *annotated disjunction*). Moreover, we plan in the future to integrate in the reasoning process, probabilistic dependencies related to profile data, in order to consider inference channels leveraging both profile and sensor databases. We further detail this research perspective in the Section 7.5 of this chapter.

Our implementation follows the workflow depicted in Figure 5.9:

- The ConsQHL is implemented as a Python module interacting with a MySQL database. Timestamps in MKUs and frequencies used in the inference channel representation has been modeled using the `datetime` and `timedelta` classes from Python's standard library. It ensures a coherent representation of temporal information between the database, the modules implemented in Python, and the one interacting with the ProbLog API.

Cette thèse est accessible à l'adresse : https://theses.insa-lyon.fr/publication/2024ISAL0024/these.pdf
© [P. Lachat], [2024], INSA Lyon, tous droits réservés

	mHealth	Orange4Home
GU & DA	$max_block = 3, period_duration = 100\,\text{s}$	
GU	$number_periods = 20$	
	$\{acc_chest_x, acc_chest_y, acc_chest_z\}$	$\{fridge_total_energy, fridge_voltage\}$
	$\{magneto_right_wrist_x, magneto_right_wrist_y\}$	$\{wm_current, wm_partial_energy, wm_power\}$
DA	$max_split = 3, number_periods = 50$	
	$\{magneto_right_wrist_x, magneto_right_wrist_y\}$	$\{fridge_total_energy, fridge_voltage\}$

Table 7.1: Values of parameters for the first evaluation. *wm* stands for washingmachine.

- The Query Based Filtering (QBF) and Search Set Filtering (SSF) modules are both implemented as methods which run filtering queries metadata and convert MKUs from the database representation to the one used in Python modules.
- The detection module is implemented as a Python module which creates a ProbLog program containing the search set of MKUs to analyze and the predicates defined for an inference channel, used to determine if this set enables its exploitation. It relies on the Python API of ProbLog to run the program and to obtain the detection output. It converts MKUs into the ProbLog representation we have defined.
- The consolidation module is implemented as multiple Python functions and thus utilize the Python representation of MKUs only.

The generator is implemented as a Python package which implements the concepts introduced in Section 6.2 and the algorithms presented in Section 6.4. Moreover, it defines a function per archetype to generate query metadata sequences according to their respective parameters values, see Section 6.5. Next, we present the evaluation methodology and how we generate datasets.

7.3 Evaluation methodology and settings

To obtain those observations, we perform four main evaluations structured as follow: we start by describing the goal of the experiment; we present the generated datasets and the parameters values set specifically for an experiment; we describe the obtained results and analyze them w.r.t. the two main objective described above. For each evaluation, 10 datasets are randomly generated using the same parameters values. Using the same values still produce different sequences of queries, e.g., the blocks select different attributes, start at different timestamp, or have different duration, and so on. This enables us to validate that our observation are not specific to an instance of an experiment dataset.

In the remainder of this chapter, we will often display the results obtained for a single dataset to display the size of the ConsQHL (which is specific to a dataset). We will explicitly state when the observation are common to all datasets or not. Moreover, the one-time attacker (OA) archetype is used in Chapter 6 to introduce the generation of malicious queries metadata in a simple setting. Since we aim to evaluate how the metrics evolve when multiple queries are issued, we do not consider for our experiments this archetype that issues a single query, see Definition 6.1.2. Instead, we leverage datasets containing sequences of queries metadata generated for the genuine user (GU) and the deceptive attacker (DA) archetypes.

Since the setting of those datasets is shared by multiple evaluations, we present here how they are generated. In addition to the values of the parameter presented in Section 6.5, we define the other values used for this evaluation, as depicted in Table 7.1. In this evaluation a single

Cette thèse est accessible à l'adresse : https://theses.insa-lyon.fr/publication/2024ISAL0024/these.pdf
© [P. Lachat], [2024], INSA Lyon, tous droits réservés

user is considered. The start of the period (*period_start*) is arbitrarily set to the time at which the dataset is generated. This time is shared by all the datasets generated in this evaluation. The duration of the periods needs to be set according to the maximum quantity of blocks to generate, as well as the duration of the inference channel constraint TSW, to ensure that both genuine and malicious blocks may be defined in a period. We chose a maximum number of blocks set to three to enable having at least two blocks dedicated to the groups of attributes (if two groups are defined) and at least one block remaining to produce genuine (resp. malicious) queries metadata for GE (resp. DA). Then, we searched the values that could be used for the two archetypes. Based on this maximum number, we searched the duration of period which enables to generate sequences for both case studies and both archetypes. For the DA archetype, the maximum number of split is chosen to simulate consolidation, without having too many *small* MKUs being created. For each case study, we associate a regular group of attributes for the two considered archetypes. The groups are defined based on the semantic of the attributes names, to represent plausible recurrent queries. We define a single group for the DA archetype since it does not mainly focus on issuing regular queries. For mHealth, two groups of three attributes are defined for the GU archetype and one group of two attributes for the DA archetype. For Orange4Home, two groups of two attributes are defined for the GU archetype and one group of two attributes for the DA archetype. We use 50 periods to generate query metadata sequences long enough to enable observing how the overhead evolves in presence of increasing quantities of MKUs in a user's ConsQHL. Yet, since the DA archetype produces more queries metadata than the GU for mHealth, only 20 periods are considered in this case. This enables generating balanced sequences of queries. Finally, we assume that most of the time, periodic queries are issued as soon as sensor data are generated, and from time to time instance queries are issued to select older sensor data. To simulate this situation, we randomly select 25% of queries metadata produced by genuine blocks an we emit them in a time greater than their creation time. All the remaining queries metadata are emitted in the order they are generated. The one produced by malicious blocks are also randomly placed on the emission timeline, after their creation.

In addition to those datasets, we have utilized three VMware virtual machines to perform the evaluations. Two first run on an Intel(R) Xeon(R) Silver 4214 CPU @ 2.20 GHz and 15 GiB of RAM. The third virtual runs on an Intel(R) Xeon(R) Gold 6126 CPU @ 2.60 GHz and 15 GiB of RAM. In the next section, we present our evaluations.

7.4 Evaluations of RICE-Sy

The overall objectives of the evaluations is to observe for both mHealth and Orange4home: (i) How the conceptual optimizations, presented in Section 5.2.5 of Chapter 5, and the storage of consolidated MKUs reduce the overhead of RICE-Sy. (ii) How specific querying settings impact the overhead of RICE-Sy and ConsQHL.

In the following sections, to address those objectives, we present the four evaluations organized as follow:

7.4.1 We start by observing the impact that our proposed optimizations have on RICE-Sy, considering a single user following the GU and the DA archetypes.

7.4.2 We study how the order in which queries are issued impacts the overhead.

7.4.3 We determine which impact different settings of consolidating queries metadata have on our two metrics.

7.4.4 We observe how the quantity of users issuing queries impact our system when we consider a fixed amount of queries.

Cette thèse est accessible à l'adresse : https://theses.insa-lyon.fr/publication/2024ISAL0024/these.pdf
© [P. Lachat], [2024], INSA Lyon, tous droits réservés

7.4.1 Monitoring the impact of our proposed optimizations

The two type of optimization we proposed are the usage of filtering modules and the storage of consolidated MKUs in the ConsQHL.

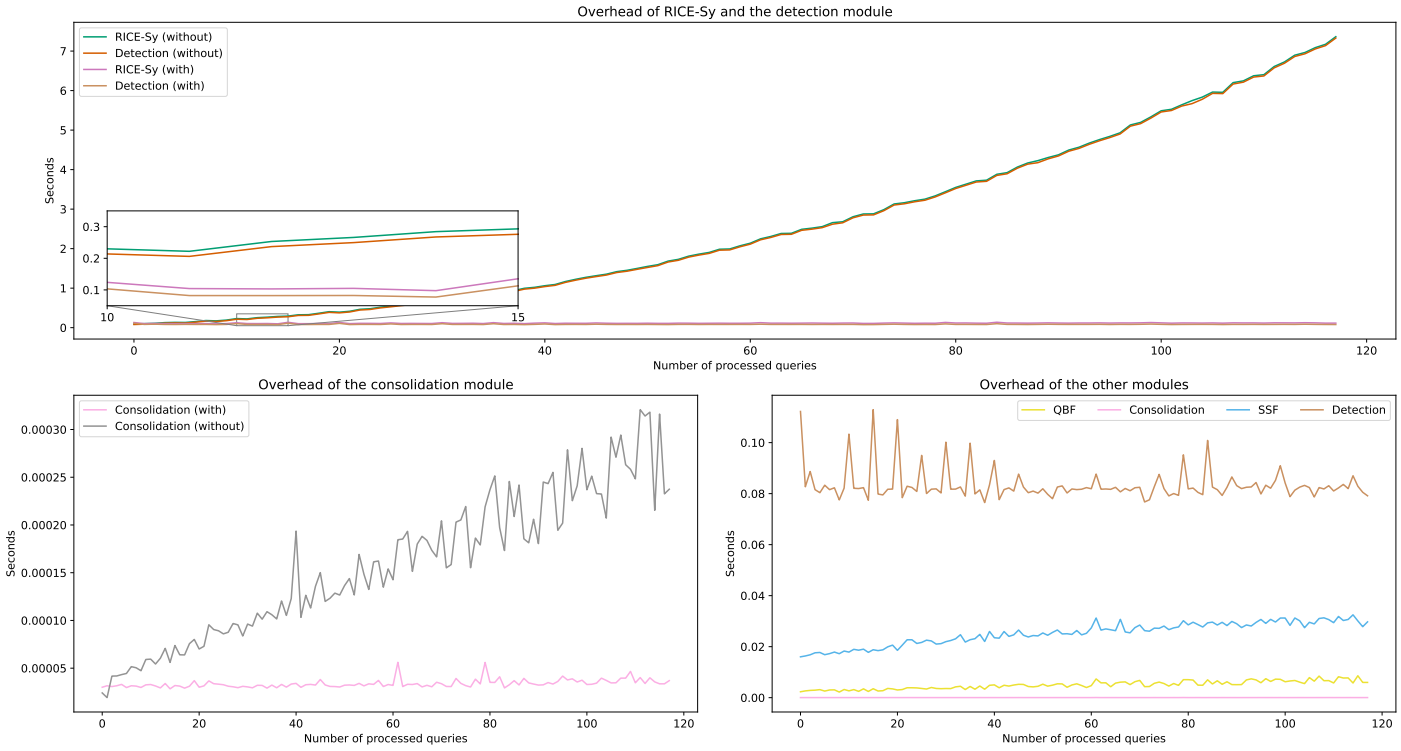
Filtering modules

When processing the query issued by a user, the consolidation and the detection modules have to consider both MKUs originating from the ConsQHL and the query metadata. The default approach is to consider the whole ConsQHL of the user. Yet, this results in a large quantity of MKUs to process, which increases the overhead of RICE-Sy. By using the two filtering modules as conceptual optimization: QBF is placed before the consolidation module to filter from the ConsQHL only the MKUs that are relevant to consolidate with the query metadata, and SSF is placed before the detection module to filter from the ConsQHL only the relevant MKUs to finalize the search set. The objective of this second part is to obtain the enhancement magnitude when using the filtering module. We only consider this metric, since the filtering modules do not impact the size of the ConsQHL.

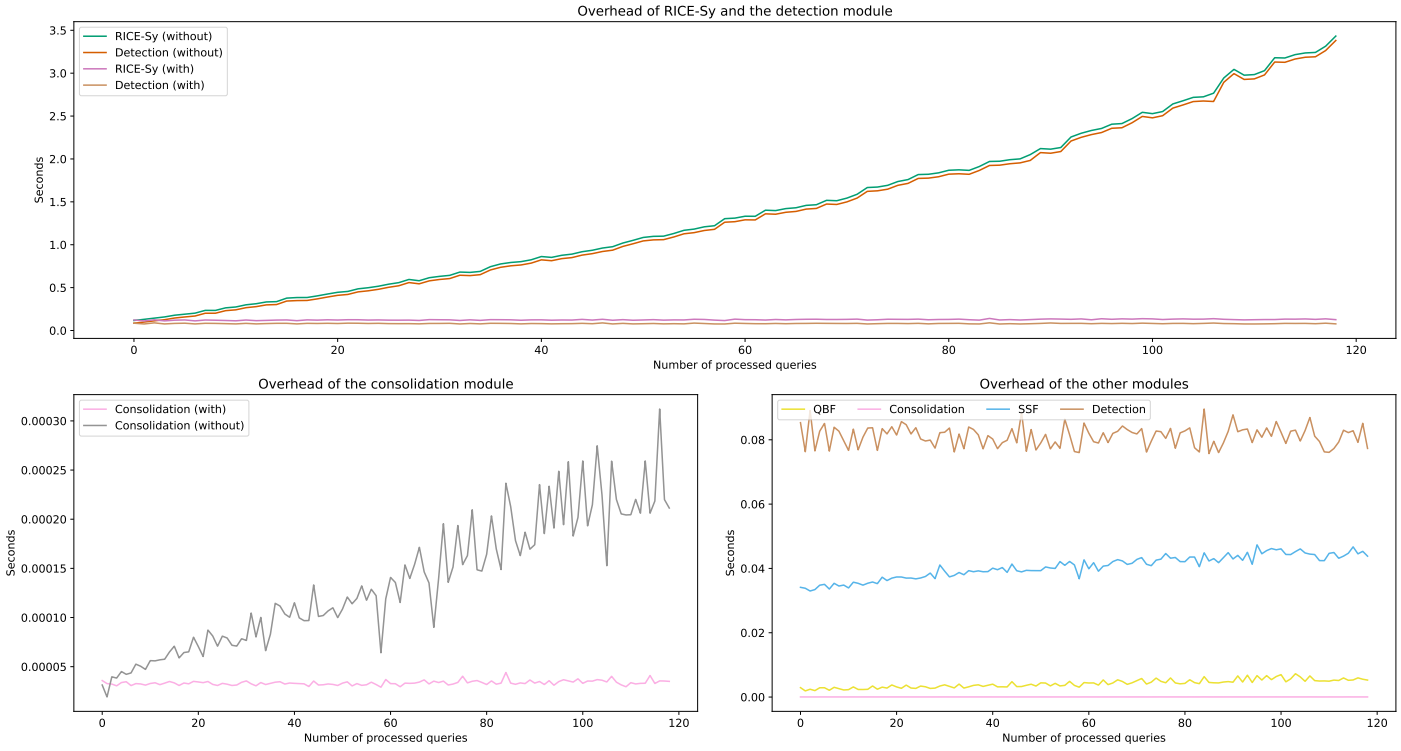
To be able to perform this comparative analysis, we measure the computation time of each module composing RICE-Sy, with and without enabling the filtering modules. To eliminate the variation of overhead resulting from a ConsQHL containing MKUs of multiple users, we consider a single user. Since we monitor how the overhead evolves w.r.t. the size of the ConsQHL, we do not consider the OA archetype which issues only a single query. The single user follows either GU or the DA archetypes.

GU: Results & Observations The results depicted in Figure 7.2 correspond to the median of the measurements performed over the 10 datasets generated for the GU archetype and our two case studies. They both share the same plot layout. The top plot in Figure 7.2a and Figure 7.2b show how the overhead (in seconds) of RICE-Sy, and of the detection module, evolves for each processed query metadata. It compares the overhead when the filtering modules are disabled (the top curves, denoted by *without* in the legend) or are enabled (the bottom curves, denoted by *with*). The zoom insert is used to show that a total of four curves are depicted. The bottom left plot compares the overhead of the consolidation module, with (top) and without (bottom) the filtering enabled. The y axis is scaled to enable the comparison of both overheads. Since in the top plot, the overhead with the filtering enabled cannot be observed precisely, the bottom right plot depicts the overhead of each module with the filtering enabled. For each query metadata, the sum of those overheads corresponds to the overhead displayed as *RICE-Sy (with)* on the top plot. We observe that for both case studies: the overhead of RICE-Sy follows the overhead of the detection module, and that the overhead of RICE-Sy increases in an exponential fashion. Due to the increasing computation time without the filtering modules, to ensure that each evaluation can be conducted within a reasonable time frame for the 10 generated datasets, we stop an evaluation once the difference of overhead is sufficiently visible. We explain the difference of overhead between mHealth and Orange4Home by the constraint of the required attributes. For mHealth, the detection module checks if MKUs satisfying the TSW and referencing the 21 attributes exists in the search set. For Orange4Home, the module checks for the MKUs satisfying the TSW and the SSW, no matter the referenced attributes. This difference is especially highlighted when the search set increases, e.g., when the filtering modules are disabled. Moreover, the more a user issues queries, the larger their ConsQHL is. Then, with the SSF disabled, the size of the search set is directly defined by the size of a user ConsQHL. We observe that for the consolidation module, when the QBF is disabled the overhead grows linearly.

Cette thèse est accessible à l'adresse : https://theses.insa-lyon.fr/publication/2024ISAL0024/these.pdf
© [P. Lachat], [2024], INSA Lyon, tous droits réservés



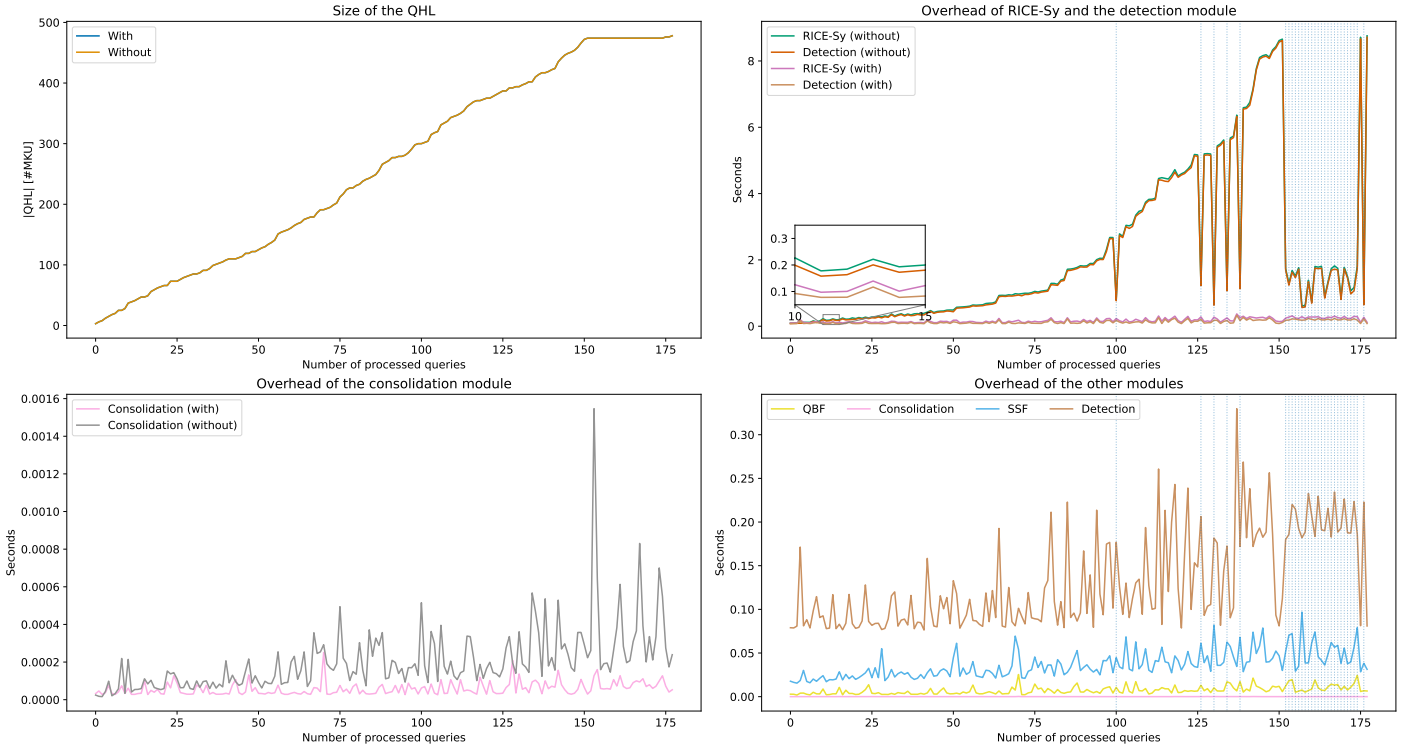
(a) mHealth



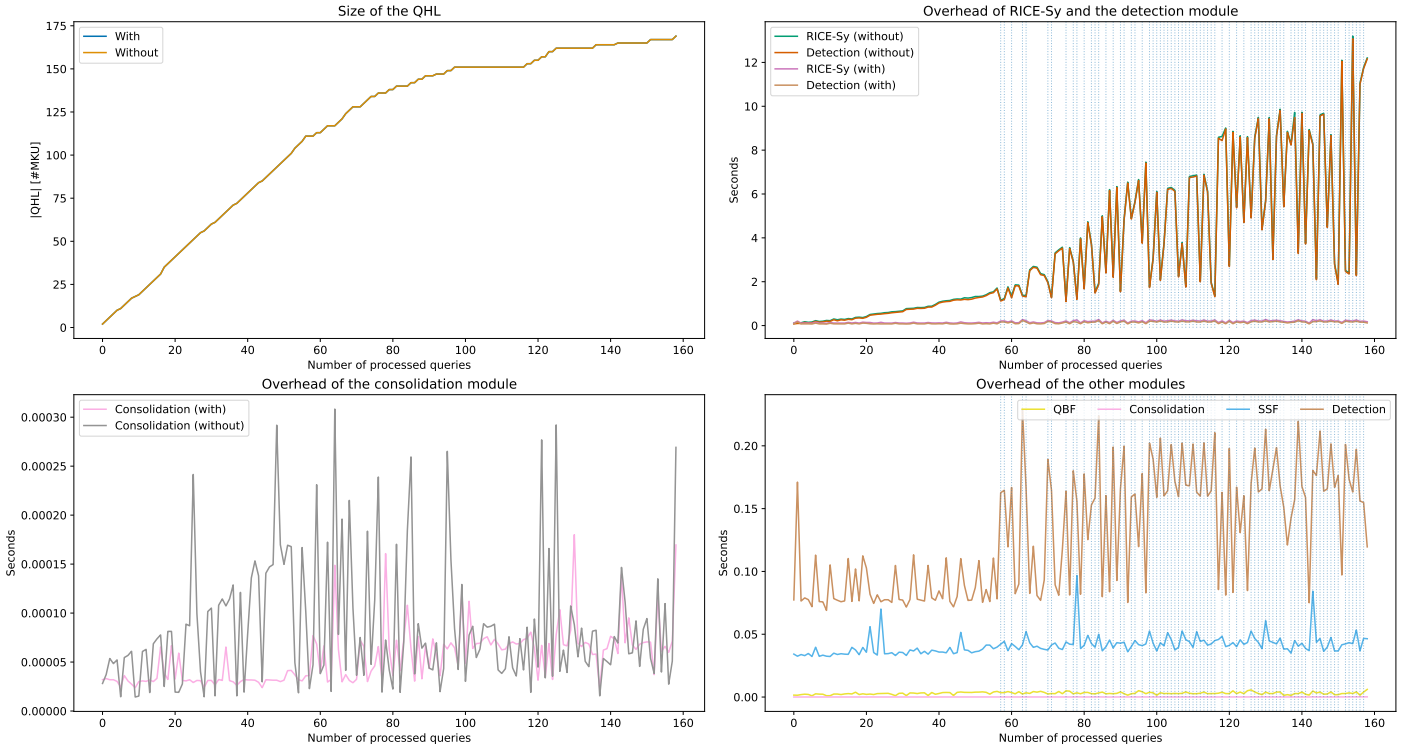
(b) Orange4Home

Figure 7.2: Evolution of the overhead considering a single user following the GU archetype. Median of measurements performed over 10 datasets using the same parameters.

Cette thèse est accessible à l'adresse : https://theses.insa-lyon.fr/publication/2024ISAL0024/these.pdf
© [P. Lachat], [2024], INSA Lyon, tous droits réservés



(a) mHealth



(b) Orange4Home

Figure 7.3: Evolution of the overhead considering a single user following the DA archetype. Measurements performed over the first dataset, among the 10 generated for mHealth, and the 10 generated for Orange4Home. Attacks are shown as vertical dotted bar.

Cette thèse est accessible à l'adresse : https://theses.insa-lyon.fr/publication/2024ISAL0024/these.pdf
© [P. Lachat], [2024], INSA Lyon, tous droits réservés

Now focusing on the overhead of each module with the filtering, we observe once again that the detection module has the largest overhead, followed by the SSF. We explain the difference of overhead between the two filtering modules by the difference of considered selection conditions. For each MKU in the query metadata, QBF retrieves the intersecting MKUs referencing the same attribute. Instead, for each MKU returned by the consolidation module, SSF retrieves the intersecting MKUs referencing a different attribute. Consequently, independently from the number of MKUs in the processed query metadata, SSF extracts from the ConsQHL a larger quantity of information than QBF. For mHealth (respectively Orange4Home), the median detection of RICE-Sy is 2.03 s (resp. 1.31 s) without the filtering modules and 0.11 s (resp. 0.12 s) with the modules enables. The proposed conceptual optimization reduce the overhead of RICE-Sy by 94% (resp. 90%). To further analyze the results, in the following we consider the measurements obtained for the DA archetype.

DA: Results & Observations For our second archetype, we consider in Figure 7.3 the measurements of a single dataset, among the 10 generated for the DA archetype and for each inference channel, vertical dotted lines are plotted whenever a query metadata leading to an inference is processed. This enables us to correlate the detection of attacks and the perturbation in the overhead of RICE-Sy or the ConsQHL size. Moreover, focusing on a single dataset enables displaying in the top left plot, how this second metric evolves for each processed query metadata. Note that for this evaluation, the size of the ConsQHL evolve similarly, hence the two curves are drawn on top of each other and only the last one is visible. Besides those changes, both Figure 7.3a and Figure 7.3b show the same information, order, and colors as for the GU archetype.

We observe that, for the 10 datasets, the ConsQHL size (denoted $|ConsQHL|$) evolves either linearly or stagnates. According to the vertical lines, this stagnation is explained by the fact that MKUs originating from a query metadata leading to an attack are not inserted into the ConsQHL, since we assume that the data controller does not answer the user with the query result. Consequently, the query metadata is not added to the user's knowledge. We observe in both Figure 7.3a and Figure 7.3b that this stagnation occurs even when vertical lines are missing. This comes from the usage of queries metadata consolidating knowledge in the DA archetype. When issuing the last query metadata produced by a genuine block simulating consolidation, its MKUs are consolidated with MKUs in the ConsQHL. The consolidated MKUs are inserted in the ConsQHL, in place of the MKUs used for the consolidation. Hence, the size of the ConsQHL stays the same in those situations. We observe that, like for GU in Figure 7.2, the overhead of RICE-Sy is drastically higher without the filtering modules enabled. Yet, we observe in Figure 7.3b that when a query metadata leads to an attack (e.g., for the processed query metadata >150), the overhead with the disabled filtering drops significantly. This stems from the fact that in the implemented prototype, once the detection module finds a subset of MKUs satisfying the patterns and constraints of an inference channel, it stops the search and notify the data controller (e.g., the processed query metadata 100 in Figure 7.3b). When processing a genuine query, the module has to check each subset of the search set to verify if the query metadata does not provide new user's knowledge leading to an attack (e.g., the processed query metadata 99 in Figure 7.3b). We observe that the difference in overhead between the consolidation module is less pronounced compared to the GU archetype but remains discernible. This observation is shared among the 10 databases. This variation is caused by the presence of consolidable queries metadata produced by malicious blocks. In the case of the GU archetype, the QBF fails to extract any MKUs from the ConsQHL, resulting in no computation required for the consolidation module. However, when QBF is disabled, the module must process the entire ConsQHL. Conversely, for the DA archetype, QBF successfully extracts MKUs from the ConsQHL, necessitating computation by the consolidation module. Without the filtering module, the consolidation module still needs to

Cette thèse est accessible à l'adresse : https://theses.insa-lyon.fr/publication/2024ISAL0024/these.pdf
© [P. Lachat], [2024], INSA Lyon, tous droits réservés

	0.25	0.5	0.75	1
mHealth	73%	70%	65%	64%
Orange4Home	73%	65%	66%	67%

Table 7.2: Percentage of prevented accesses when saving consolidated MKUs in ConsQHL, for mHealth and Orange4Home and different probabilities of blocks consolidation.

traverse the entire ConsQHL. We also notice that the overhead of the detection module continues to be the most important one, followed by the overhead of the SSF. Finally, we observe that when queries metadata leading to attacks are processed, the overhead of the detection module reaches approximately 200 ms, for less than 100 ms when processing queries metadata which does not lead to attacks (e.g., the GU archetype). We explain this difference by the nature of the archetype. Indeed, for the DA, a malicious block produces either multiple queries metadata which are not consolidable or multiple queries metadata which are consolidable. The search set of the detection module contains more MKUs when processing the last query metadata of such a block. For mHealth (respectively Orange4Home), the median detection of RICE-Sy is 1.09 s (resp. 2.10 s) without the filtering modules and 0.15 s (resp. 0.16 s) with the modules enabled. The same trend is observed for the other datasets generated for this evaluation. The proposed conceptual optimization reduces the overhead of RICE-Sy by 86% (resp. 92%).

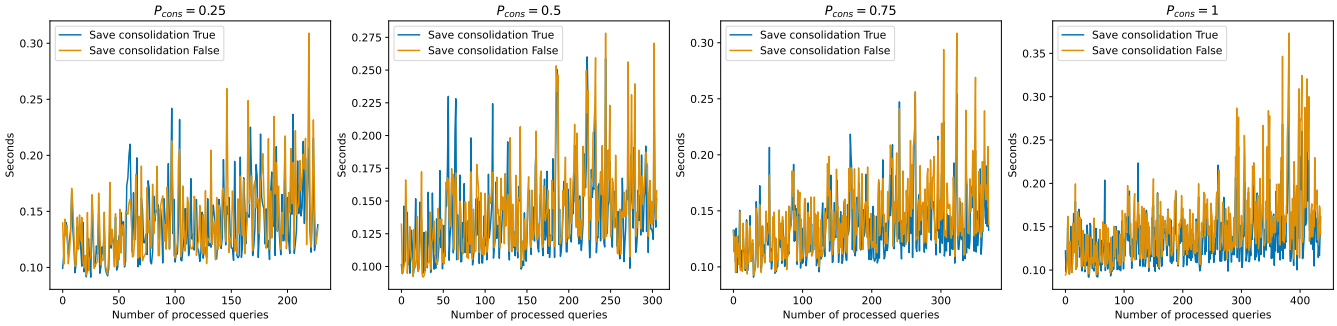
Storing consolidated MKUs

As stated in Section 5.2.3 of Chapter 5, placing the consolidation module before the detection module allows: (i) consolidating MKUs only once while processing a query metadata and (ii) when no inference is detected, caching consolidated MKUs in the ConsQHL to prevent superfluous computation when processing future queries metadata. The objective of the first part of this evaluation is to determine how this storage impacts the overhead of RICE-Sy.

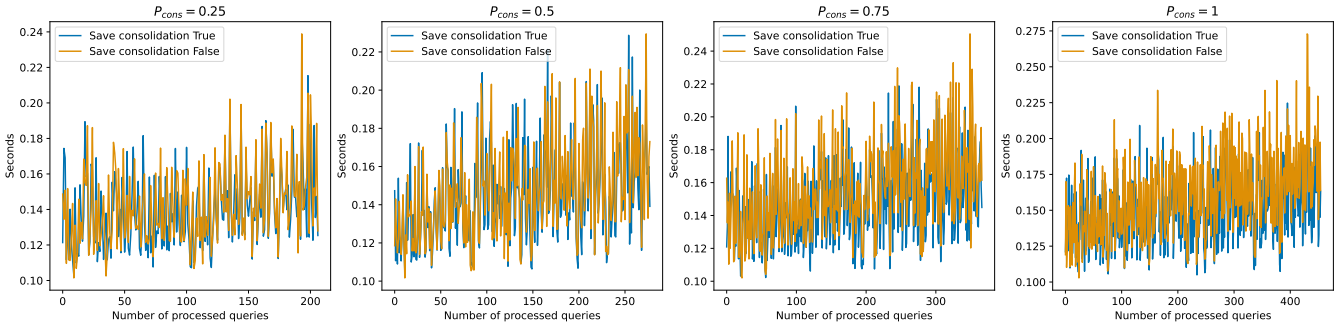
We first measure the evolution of the overhead for five versions of the same GU query metadata sequence. Each version contains the queries metadata produced by the same blocks, but with different probabilities of generating consolidable queries metadata (i.e., 0, 0.25, 0.5, 0.75, and 1). The initial queries metadata are thus similar, only the number of split queries metadata changes. For each block, based on the probability, we chose to do a maximum number of four splits (i.e., $max_split = 4$ for this evaluation only). This enables generating sequences where at least two MKUs are consolidated, up to four for a given block. Hence, each version contains more and more consolidable queries metadata than the previous version. We consider only the GU archetype to eliminate the variation which stems from having queries metadata leading to inference attacks. For each version of a dataset, we measure the overhead of RICE-Sy twice, once by storing consolidated MKUs in ConsQHL and another by storing only the non-consolidated MKUs, thus forcing RICE-Sy to re-consolidate MKUs multiple times if needed.

GU: Results & Observations The more a stored MKU is accessed when processing queries, the more unnecessary consolidation can be avoided, leading to improved efficiency and performance. To observe the impact of this optimization, the number of accesses to consolidated MKUs was measured for the 10 datasets previously generated, with respect to the density of consolidable queries. To do so, for each version of a dataset, we measured once the number of access when storing the new or consolidated MKUs (i.e., $QM^u_{new} \cup MK^u_{new}$ in Algorithm 5.5), and once by storing the MKUs of the processed query metadata (i.e., QM^u_Q in Algorithm 5.5). Figure 7.2 presents the results for the first dataset, considering mHealth and Orange4Home. Similar trends were observed for the other datasets. It depicts the sum of access to MKUs, with and without

Cette thèse est accessible à l'adresse : https://theses.insa-lyon.fr/publication/2024ISAL0024/these.pdf
© [P. Lachat], [2024], INSA Lyon, tous droits réservés



(a) mHealth



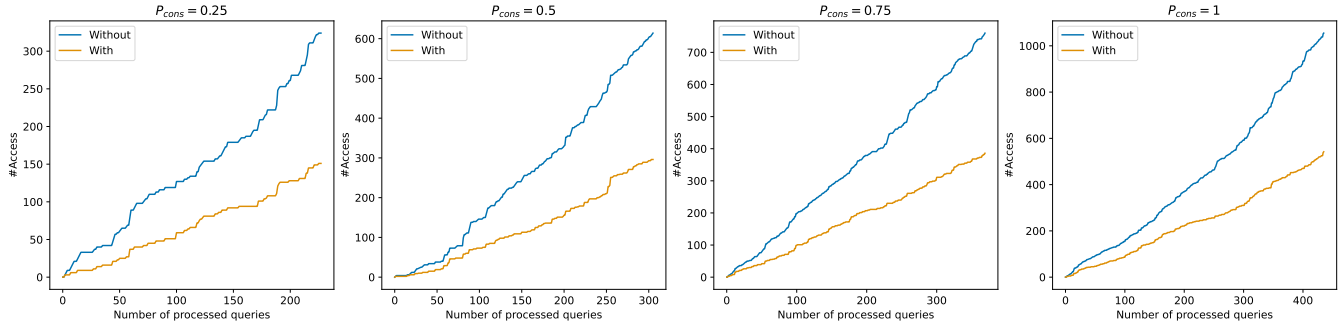
(b) Orange4Home

Figure 7.4: Evolution of the overhead considering a single user following the GU archetype. Median of measurements performed over 10 datasets using the same parameters.

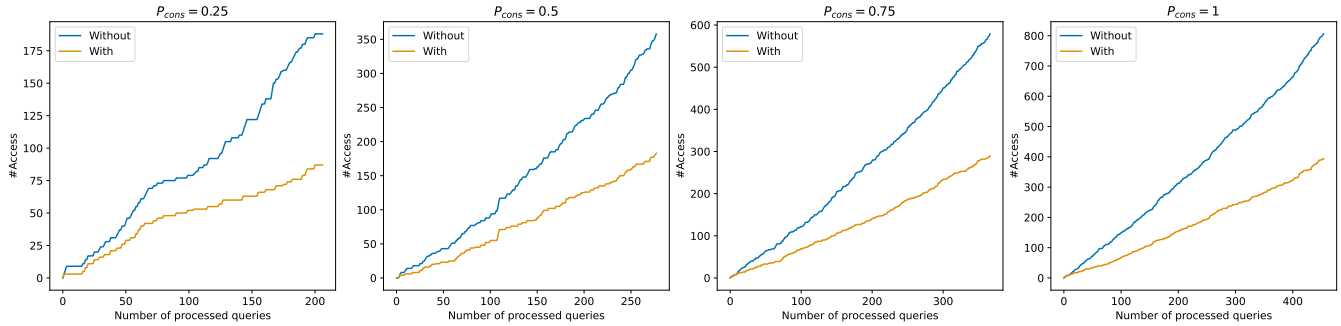
storing consolidated MKUs in ConsQHL. The x-axis represents the number of processed queries metadata. Each plot is labeled using the probability of consolidation (ranging from 0.25 to 1, since the initial sequence contains no consolidation for each dataset) used for the respective version of the dataset. The observation is that, for all datasets both case studies (mHealth and Orange4Home), saving the consolidated MKUs greatly reduce the number of access in ConsQHL. Table 7.2 shows that the percentage of avoided access is above 60%, independently from the probability of consolidation.

By reusing the same datasets, we can observe how the overhead behaves. Figure 7.4 depicts the results for the first dataset for mHealth and for Orange4Home. All the other 9 datasets produce similar result. For both inference channels, the layout of Figure 7.4a and Figure 7.4b is identical: each plot displays two curves representing the overhead of RICE-Sy with and without saving the consolidated MKUs in to the ConsQHL. Each plot (from left to right) is labeled using the probability of consolidation (from 0.25 to 1, since the initial sequence contains no consolidation for each dataset) used for the version of the dataset. We observe here that for both case study, when the filtering modules are enabled, the impact of caching consolidated MKUs in the ConsQHL is negligible for the probabilities $P_{cons} = 0.25$ and $P_{cons} = 5$. Indeed, even if the ConsQHL contains more MKUs when the consolidated one are not stored, the filtering modules greatly amortize this drawback by extracting only a subset of the ConsQHL. Even if the difference isn't large, we observe that for higher probability of consolidation, the overhead slowly increases when consolidated MKUs are not cached. As observed in the first part, this support the fact that even with the filtering modules, the growth of ConsQHL negatively impacts the overhead of RICE-Sy.

Cette thèse est accessible à l'adresse : https://theses.insa-lyon.fr/publication/2024ISAL0024/these.pdf
© [P. Lachat], [2024], INSA Lyon, tous droits réservés



(a) mHealth



(b) Orange4Home

Figure 7.5: Sum of access to MKUs in the ConsQHL, with and without storing the consolidated MKUs, for varying probabilities of consolidation considering the GU archetype. Measurements of the first dataset, among the 10 generated for mHealth and Orange4home.

Through this first evaluation, we validate that the conceptual optimizations we propose, i.e., the filtering modules, reduce the overhead of RICE-Sy by $\approx 90\%$, when considering queries issued by a single user for both of our case studies. Other case studies with different quantities of required attributes and constraints may lead to small changes in our observation, yet the principal factor that increase the overhead is the quantity of collected MKUs, independently from the considered inference channel.

7.4.2 How the query issuing order impacts RICE-Sy?

Assessing the influence of the query metadata order requires a sequence of multiple queries, in which certain queries metadata can be rearranged freely on the emission timeline (see Figure 6.9 in the previous chapter). The GU and DA archetype enable to consider the ordering of queries metadata generated from both genuine and malicious blocks. The objective of this evaluation is to determine if the order in which the same sequence of queries metadata is processed has a significant impact on both metrics we are interested with (i.e., the overhead of RICE-Sy and the size of ConsQHL).

We measure the size of the ConsQHL (as the number of stored MKUs), and the overhead of RICE-Sy (as the sum of the overhead of the QBF, consolidation, SSF, and detection modules). The measurements are performed on the first virtual machine, see Section 7.4. To eliminate the variation of overhead resulting from a ConsQHL containing MKUs of multiple users, we consider a single user. We used the ten datasets generated for the first experiment, with the GU and the DA

Cette thèse est accessible à l'adresse : https://theses.insa-lyon.fr/publication/2024ISAL0024/these.pdf
© [P. Lachat], [2024], INSA Lyon, tous droits réservés

		1	2	3
GU	mHealth	0.1205 s	0.1194 s	0.1176 s
	Orange4Home	0.1341 s	0.1327 s	0.1312 s
DA	mHealth	0.1510 s	0.1461 s	0.1508 s
	Orange4Home	0.1648 s	0.1587 s	0.1582 s

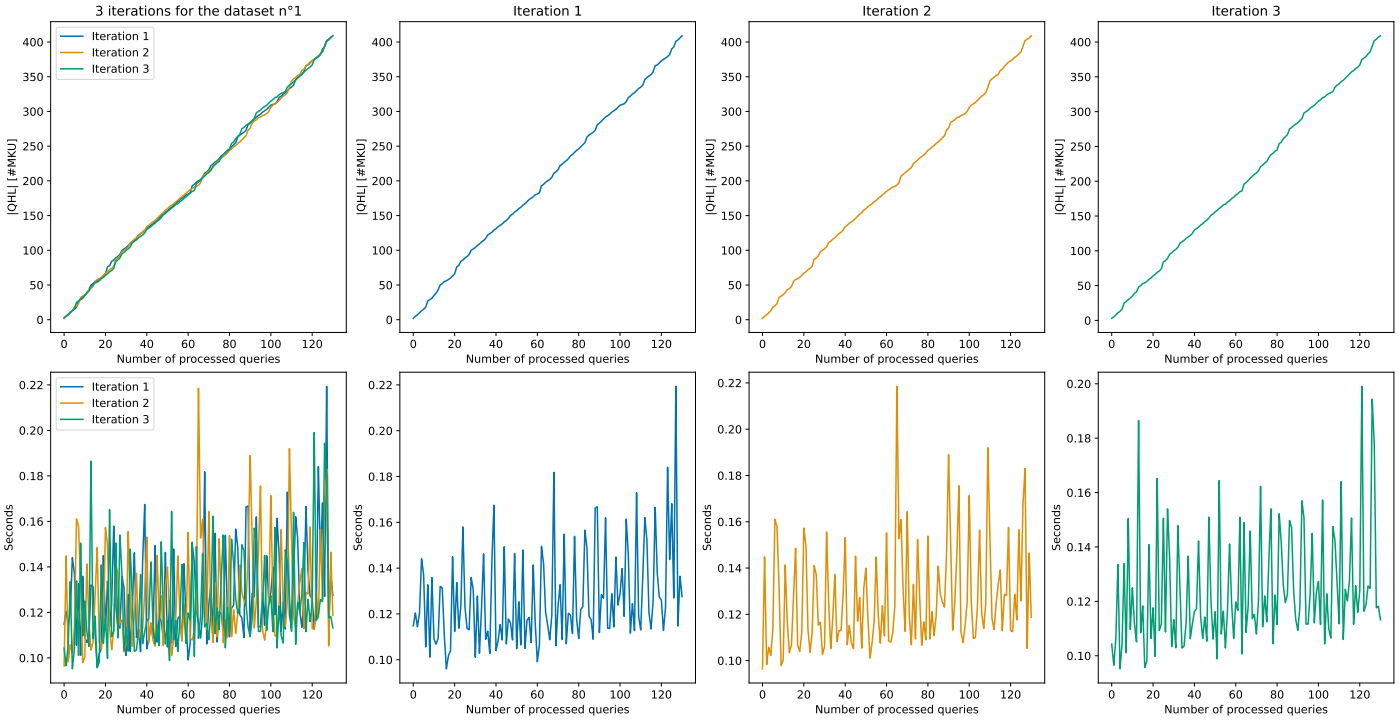
Table 7.3: Median overhead for mHealth and Orange4Home, according to the three iterations.

archetype. The experiment is repeated three times for each dataset. For each dataset, we perform the measurements three times by modifying the emission time of queries metadata related to the genuine instance queries metadata and the malicious queries. However, in this evaluation, randomly placing malicious queries metadata can lead to curves that cannot be compared. For instance, let us consider three queries metadata produced by the same malicious block and containing different quantities of MKUs. In each iteration, the two first queries metadata are inserted into the ConsQHL and only the third one, which may be different from one iteration to the other, is not since it is detected as leading to an IAISD. In this setting, the final size of the ConsQHL will be different between each iteration. Consequently, to obtain comparable results, the queries metadata originating from the same malicious block are placed randomly over the emission timeline, but in the same order in each iteration.

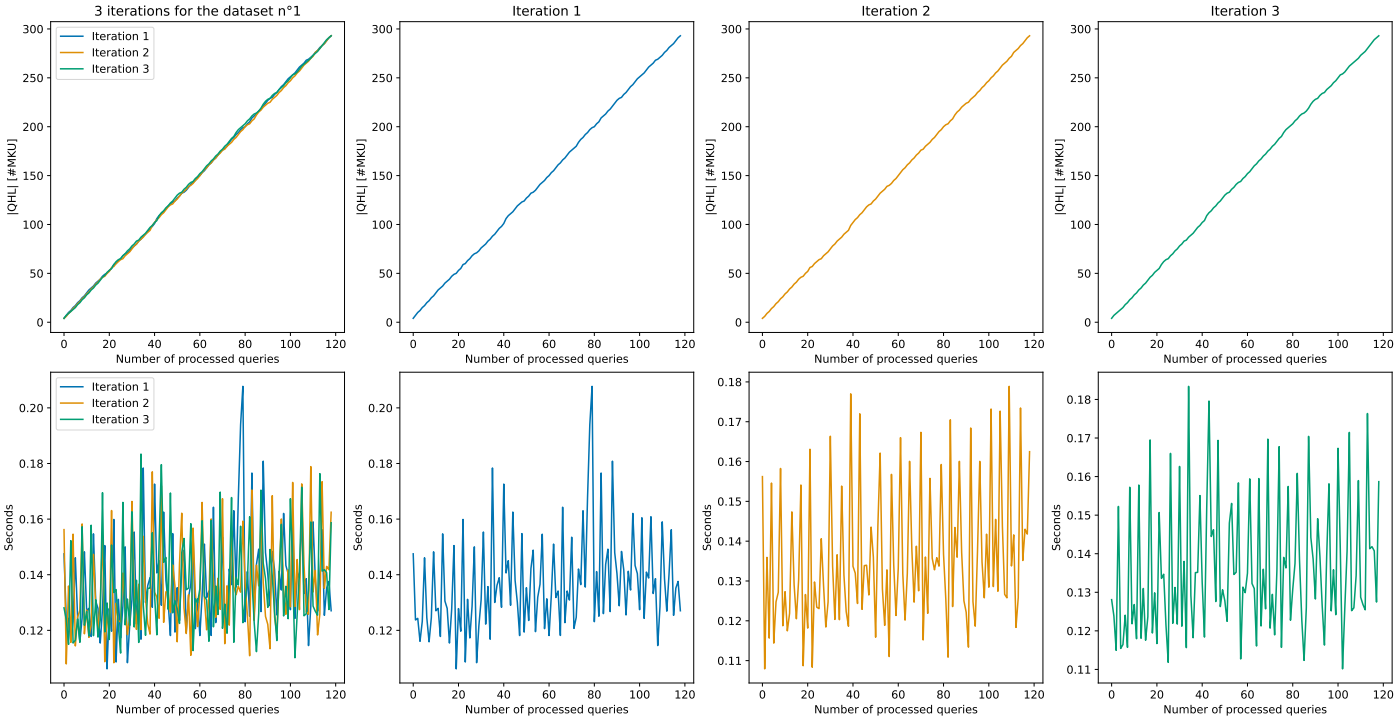
Results & Observations The results depicted in Figure 7.6 and Figure 7.7 correspond to measurements performed for the first dataset among the 10 generated for mHealth, and the 10 generated for Orange4Home. For both inference channels, the layout depicts for the same sequence and for all (first plot) as well as each of the three iterations (three other plots): the evolution of the size of the ConsQHL above; and the evolution of the overhead below. Focusing on a single dataset enables displaying the queries metadata leading to an inference attack for each iteration. We observe that while queries metadata are processed, the size of the ConsQHL has some variation between two iterations. This is comes from the fact that the same quantity of selected genuine queries metadata and malicious queries are issued in-between genuine queries metadata from one iteration to the other. Consequently, the number of MKUs inserted in the ConsQHL depends on the order in which they are processed. As compiled in Table 7.3, for both archetypes, the median overhead of RICE-Sy has only small variation between each of the three iterations depicted in both Figure 7.6 and Figure 7.7. Those limited variations are observed for the 10 datasets. They can be explained by the described variation of the size of the ConsQHL.

Through this second evaluation, we conclude that for the same sequence of queries issued by the DA archetype, the order in which queries metadata produced by malicious blocks are issued have no significant impact on the two metrics we consider. To determine the limits of RICE-Sy, besides the archetypes, we have to focus on extreme cases. In the following, we discuss a situation where a user issues diverse quantities of consolidable queries.

Cette thèse est accessible à l'adresse : https://theses.insa-lyon.fr/publication/2024ISAL0024/these.pdf
© [P. Lachat], [2024], INSA Lyon, tous droits réservés



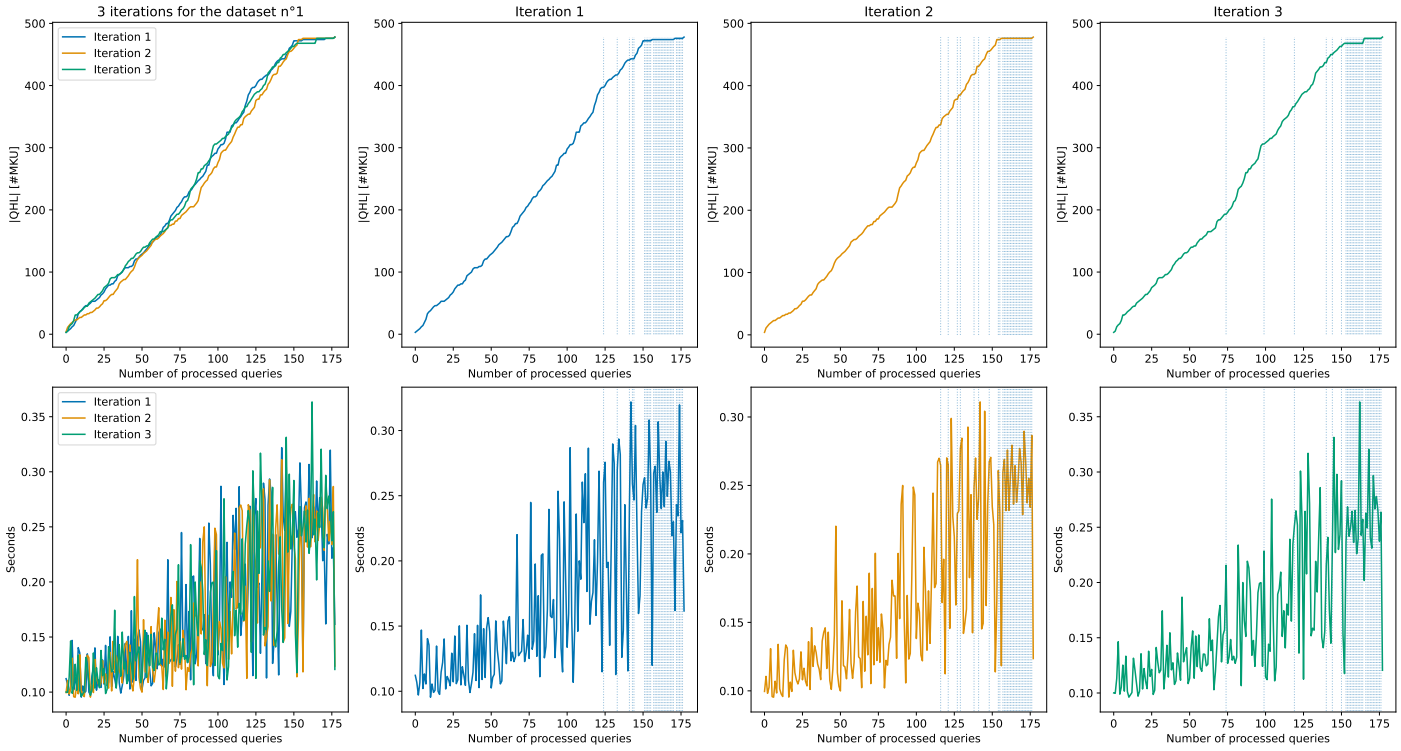
(a) mHealth



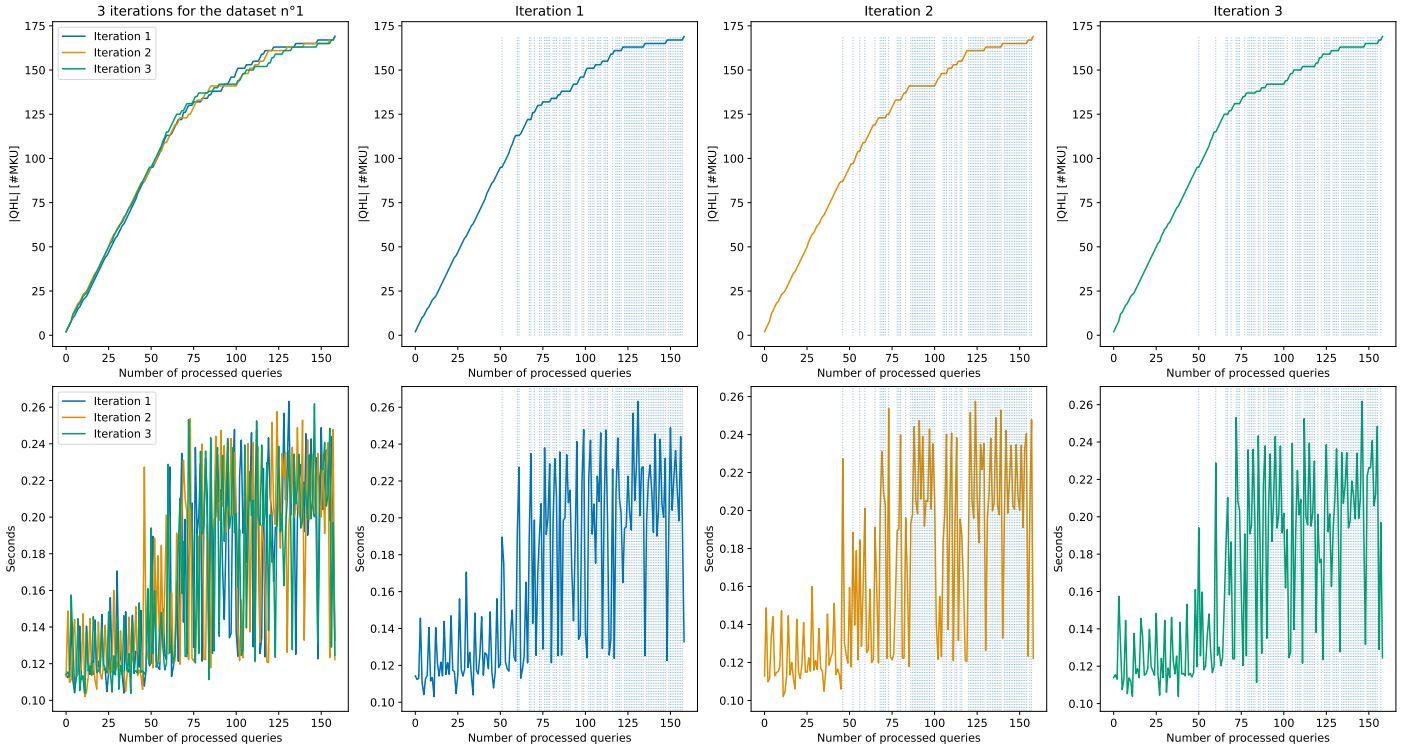
(b) Orange4Home

Figure 7.6: Impact of varying three time the order in which queries are issued by the GU archetype. Measurements of the first dataset, among the 10 generated for mHealth and Orange4home.

Cette thèse est accessible à l'adresse : https://theses.insa-lyon.fr/publication/2024ISAL0024/these.pdf
© [P. Lachat], [2024], INSA Lyon, tous droits réservés



(a) mHealth



(b) Orange4Home

Figure 7.7: Impact of varying three time the order in which queries are issued by the DA archetype. Measurements of the first dataset, among the 10 generated for mHealth and Orange4home. Attacks are shown as vertical dotted bar.

Cette thèse est accessible à l'adresse : https://theses.insa-lyon.fr/publication/2024ISAL0024/these.pdf
© [P. Lachat], [2024], INSA Lyon, tous droits réservés

	0	0.25	0.5	0.75	1
mHealth	0.0992 s	0.1054 s	0.1053 s	0.1090 s	0.1102 s
Orange4Home	0.1145 s	0.1171 s	0.1210 s	0.1175 s	0.1256 s

Table 7.4: Median overhead for mHealth and Orange4Home, according to different probabilities of blocks consolidation.

7.4.3 How the consolidation impacts RICE-Sy

We have observed in the first evaluation that the consolidation module has the smallest overhead in RICE-Sy. Yet, a user may issue large quantities of consolidable queries metadata which results in an increase of the consolidation overhead. The objective of this evaluation is to investigate in which of those cases the overhead of RICE-Sy increases significantly. For readability purpose, in the remainder of this section, we use the formulation "consolidable queries" to describe queries metadata having MKUs that are consolidable. We first measure the evolution of our two metrics for five versions of the same GU query metadata sequence. Since the settings are shared with the evaluation presented in Section 7.4.1, we reuse the same datasets.

Results & Observations Figure 7.8 depicts the results for the first dataset for mHealth, and for Orange4Home. All the other 9 datasets produce similar result. For both inference channels, the layout of Figure 7.8a and Figure 7.8b is identical to the first evaluation: the ConsQHL size is displayed on the top left plot, the overhead of RICE-Sy is on the top right plot, the overhead of the consolidation module is on the bottom left plot, and the last plot depicts the overhead of all modules (without the consolidation). Each curve (from left to right) is labeled using the probability of consolidation (from 0 to 1) used for the version of the dataset. Since a higher probability produces more queries, each version of the same initial dataset (i.e., the one with the probability equals to zero) has different number of queries.

For the initial version of the dataset, the size of ConsQHL steadily increases. In other version of the dataset, the size can decrease temporarily when a query metadata leads to the consolidation of several MKUs. Moreover, as depicted in the zoomed embedded plot of Figure 7.8a, MKUs can accumulate quickly in one version (e.g., the one with $P_{cons} = 1$) when a sequence of received queries metadata is not consolidable. Here, the explanation is similar to the one provided in the analysis of the first evaluation. We observe that the final size of the ConsQHL is equal for all versions. This is expected, since they all stem from the same sequence of queries. After being consolidated, they produce the same set of MKUs. As depicted by Table 7.4 for our two case studies, we see that no matter the version, the median overhead of all modules is not significantly impacted by the density of consolidable queries. We explain this as a side effect of issuing consolidated queries metadata in the order they are generated. By doing so, each query metadata can be consolidated as soon as possible by RICE-Sy. mHealth and Orange4Home have an overhead median difference of 5% and 3%, respectively. Those differences are observed among the 10 datasets. Consequently, the density of consolidable queries metadata has no significant impact on RICE-Sy when their MKUs are consolidated upon processing the query metadata.

To further study the impact of the consolidation, we consider another extreme case in which a GU archetype issues a query metadata sequence containing mainly non consolidable queries, with few queries metadata triggering the consolidation of large quantities of MKUs stored in the ConsQHL. It enables showing how RICE-Sy behaves when the consolidation does not append for each processed query metadata, but rather at once, when processing a triggering query metadata. While this extreme case can be shown for Orange4Home, it isn't as important as for mHealth, since no large triggering query metadata can be generated without selecting less than 20 data

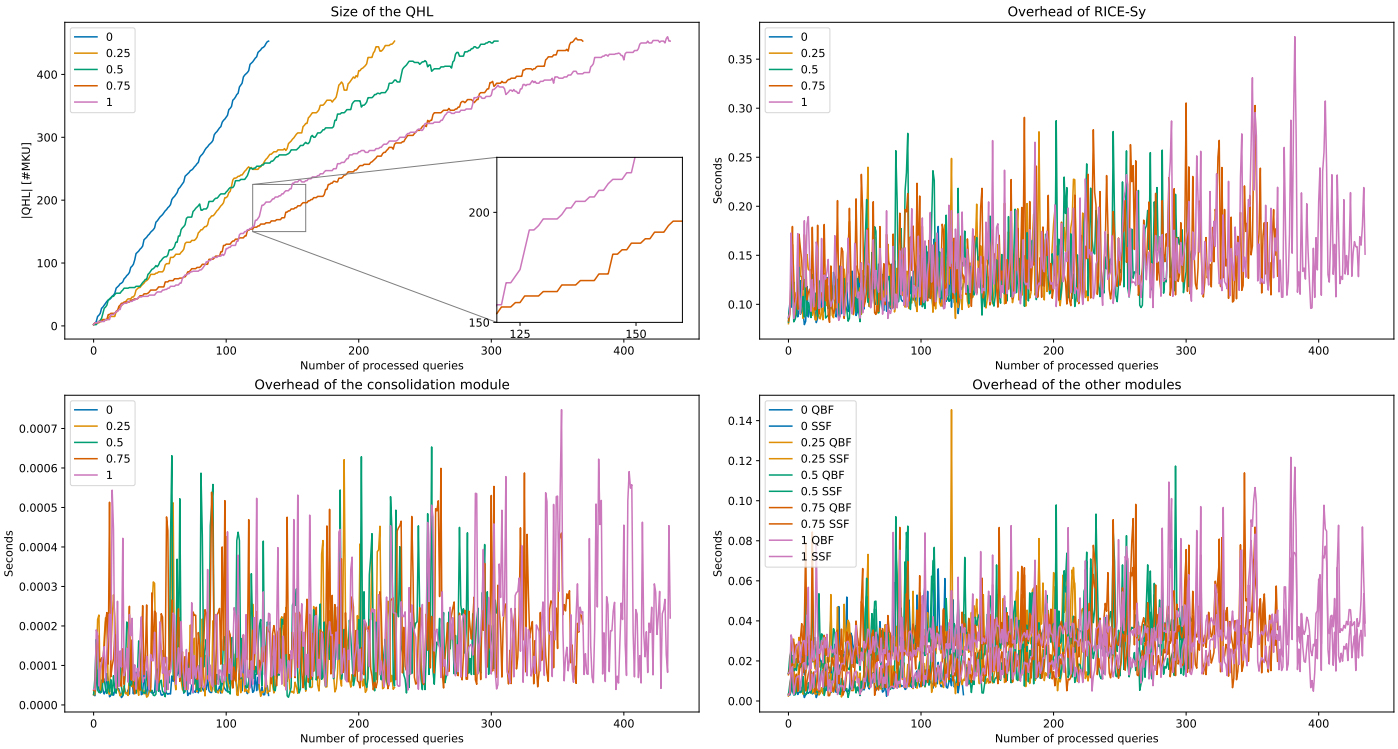
Cette thèse est accessible à l'adresse : https://theses.insa-lyon.fr/publication/2024ISAL0024/these.pdf
© [P. Lachat], [2024], INSA Lyon, tous droits réservés

points. We should observe an increase of the overhead from QBF and the consolidation module. The overhead of both SSF and the detection module should not be drastically impacted, since the search set is consolidated thus containing the same quantity of MKUs than the triggering query metadata. The measurements and datasets settings are similar to the one previously described. The difference is that from the initial dataset, we generate four versions of this dataset in which we have one, two, three, and four queries, respectively. The emission time of those additional queries metadata is equally distributed over the initial version. For instance, in the version containing three new queries, the first one is issued after the first third of the initial queries. It contains an MKU per attribute referenced in the first third, having a temporal duration spanning all MKUs in the group of queries. We called those queries, *trigger query metadata* in the following.

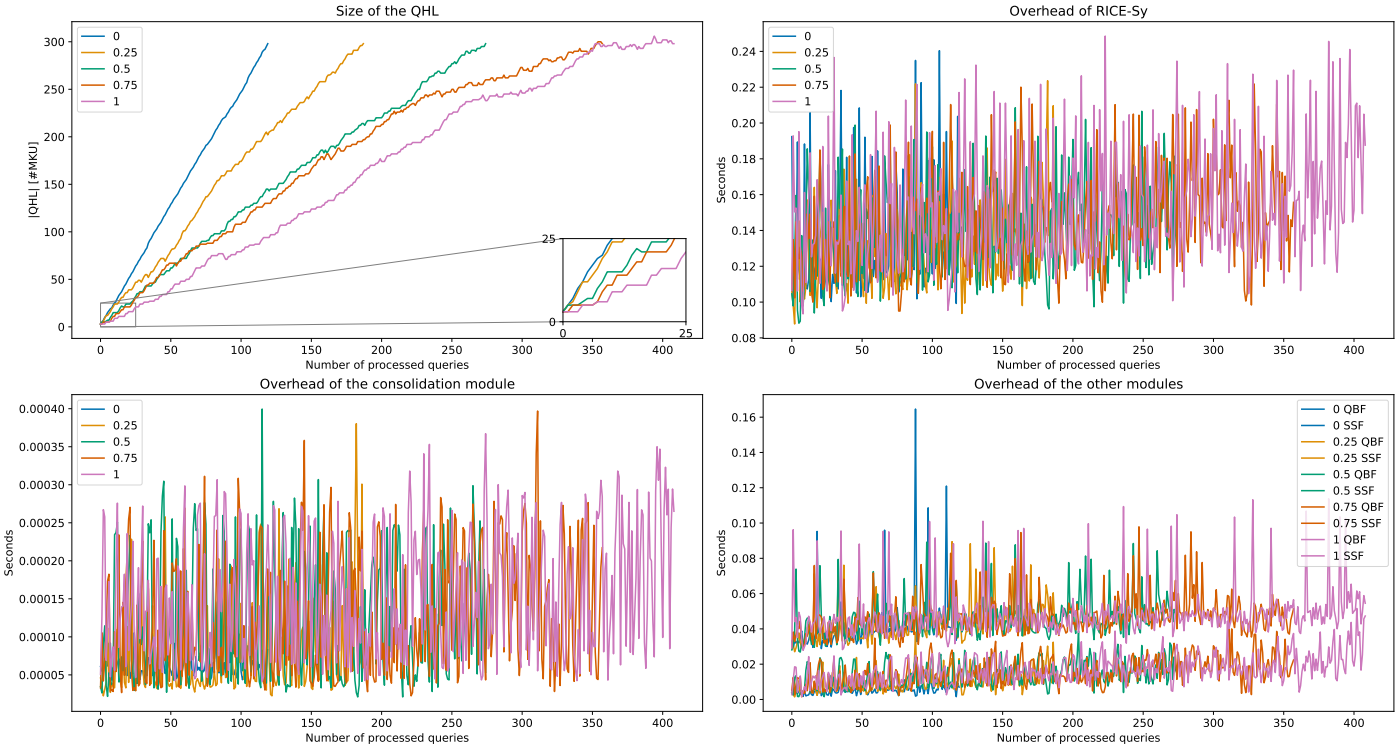
Results & Observations The layout of Figure 7.9 is identical to Figure 7.8. Each curve is labeled with the quantity of added queries metadata triggering the consolidation. Considering the size of the ConsQHL first, for each version of the dataset we observe the peaks corresponding to the query metadata triggering the consolidation. Those peaks are also visible for the overhead of RICE-Sy and each of its modules. The triggering queries metadata cause a maximum increase of overhead reaching 6.32 s for mHealth in Figure 7.9a, compared to 1.43 s for Orange4Home in Figure 7.9b. This is related to the limitation of our second case study, presented in the previous paragraph. Even if each trigger query metadata contains MKUs with temporal interval overlapping a large quantity of MKUs in the ConsQHL, the overhead of both filtering modules stays in the same order of magnitude as in the previous evaluation. For mHealth, the SSF module has a median overhead of 18 ms in Figure 7.9a, versus $\approx$ 24 ms in Figure 7.8a. We observe that for mHealth, the highest overhead of the consolidation module is nearly doubled in presence of triggering queries metadata (i.e., equal to 8 ms), than compared to the previous evaluation when consolidation is performed as soon as possible (i.e., equal to 4 ms). Even in those extreme cases, the overhead of this module stays negligible compared to other modules, since the overhead of RICE-Sy increases from 0.10 s to 6.32 s for the two highest peaks. We explain those two observations, discovered in the 10 datasets, by the fact that they exploit the optimization related to the indexing of the MySQL database and the iterative algorithm implemented in Python, compared to the usage of ProbLog in the detection module. The overhead of the detection, and thus RICE-Sy, "explodes" when processing the trigger query metadata. When processing the trigger query metadata, the SSF retrieves a large quantity of MKUs, compared to the non-trigger query metadata. The search set of the module increases significantly. Yet, in this specific context, the overhead of the detection module is even higher than the one observed in the first evaluation, when the filtering modules are disabled, e.g., Figure 7.2. We explain this difference by the presence, in the search set, of MKUs originating from the trigger query metadata. Those MKUs temporally intersect all the other MKUs in the search set, thus drastically increasing the possible combination of subsets to check to determine if the query metadata leads or not to an inference attack.

Via this third evaluation, we conclude that, whatever the density of issued consolidable queries, when the consolidation module consolidates MKUs as soon as possible, i.e., when the query metadata is processed, its impact on the overhead of RICE-Sy is not significant. While extreme case can trigger costly consolidation, they seem unlikely to happen compared to the type of querying behavior observed in the previous chapter. In the first evaluations, we have considered a single user to simplify the analysis. However, in a real situation multiple users interact with sensor databases. In the following, we analyze the impact of multiple users issuing the same queries metadata as a single user.

Cette thèse est accessible à l'adresse : https://theses.insa-lyon.fr/publication/2024ISAL0024/these.pdf
© [P. Lachat], [2024], INSA Lyon, tous droits réservés



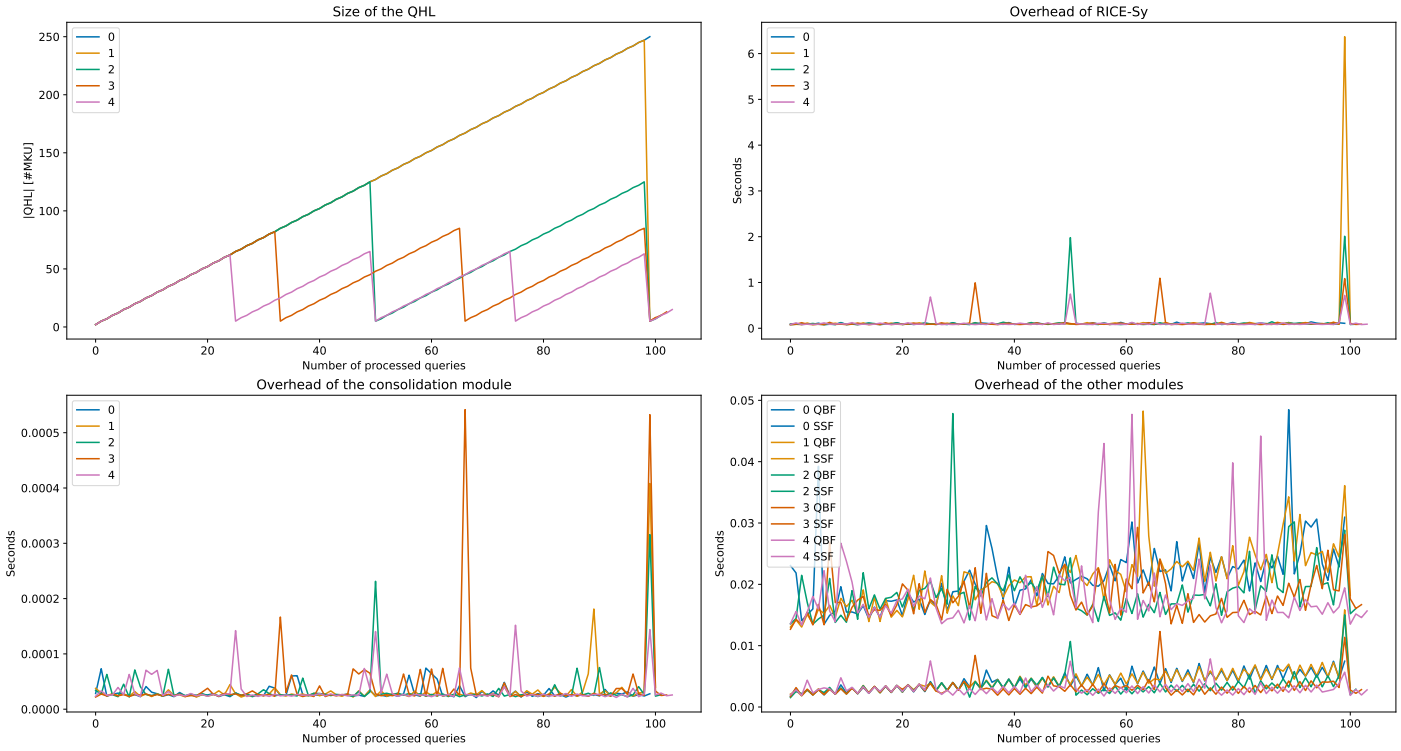
(a) mHealth



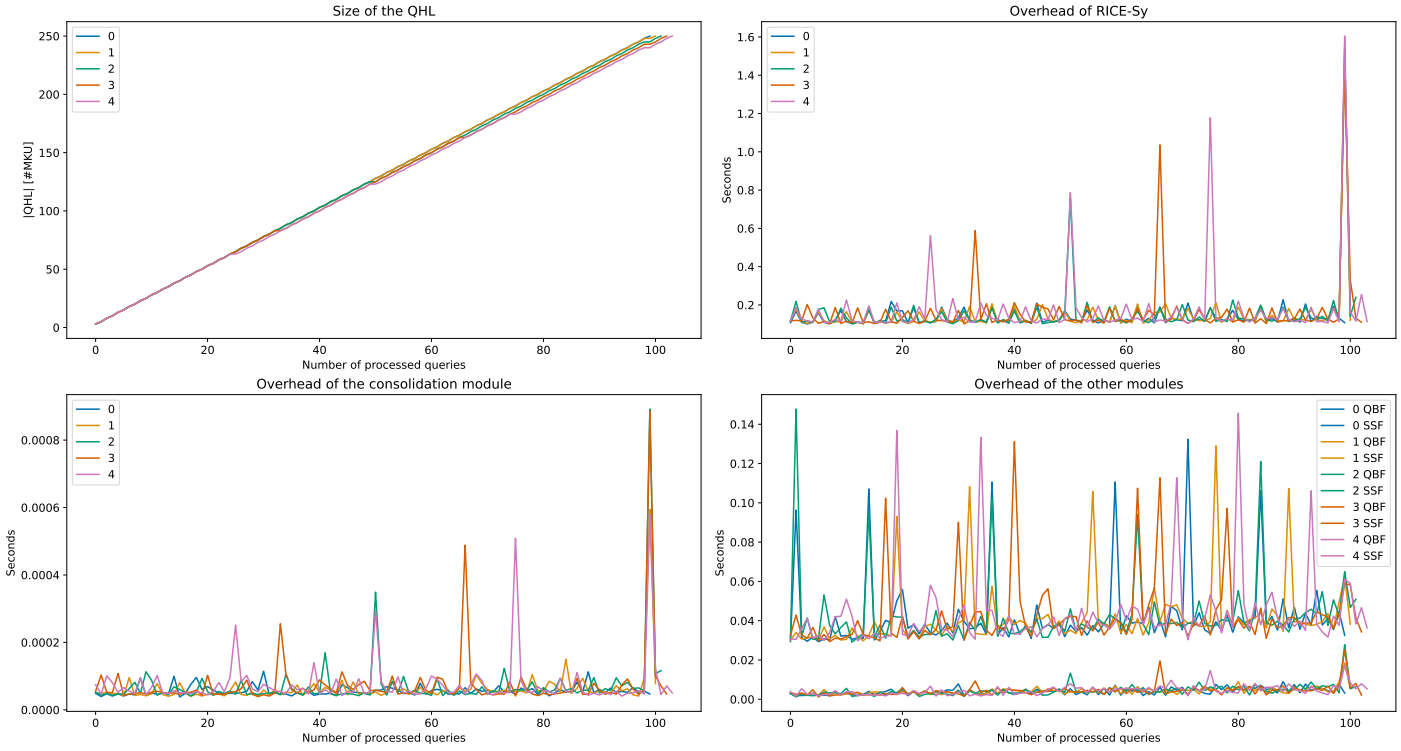
(b) Orange4Home

Figure 7.8: Impact of varying probabilities of consolidation for the GU archetype. Measurements of the first dataset, among the 10 generated for mHealth and Orange4home.

Cette thèse est accessible à l'adresse : https://theses.insa-lyon.fr/publication/2024ISAL0024/these.pdf
© [P. Lachat], [2024], INSA Lyon, tous droits réservés



(a) mHealth



(b) Orange4Home

Figure 7.9: Impact of varying the number of query metadata triggering the consolidation for the GU archetype. Measurements of the first dataset, among the 10 generated for mHealth.

Cette thèse est accessible à l'adresse : https://theses.insa-lyon.fr/publication/2024ISAL0024/these.pdf
© [P. Lachat], [2024], INSA Lyon, tous droits réservés

7.4.4 How multiple users impact RICE-Sy for a fixed number of queries

When a user issues a query to the sensor database, the query metadata is processed by RICE-Sy. According to the results of the detection, our system updates the ConsQHL dedicated to this user to keep track of their knowledge. Hence, issuing the same queries by a single user, or by a set of users results in a ConsQHL containing the same MKUs, partitioned differently. The objective of this evaluation is to assess how issuing a fixed quantity of queries by different number of users impacts the performance of RICE-Sy, and to observe the impact of the filtering process in the two settings.

To be able to perform this comparative analysis, we measure the computation time of the detection and the consolidation modules with the filtering modules enabled. Since we monitor how the overhead evolves w.r.t. the size of the ConsQHL, we do not consider the OA archetype. We generate an initial dataset with a single sequence, and we create new versions of the dataset by randomly assigning the initial queries metadata to five, ten, and twenty users of the same archetype.

GU: Results & Observations The results shown in Figure 7.10 correspond to the measurements realized on the first datasets generated for mHealth, and for Orange4Home, considering the GU archetype. Both Figure 7.10a and Figure 7.10b share the same layout. The top plot displays how the size of the ConsQHL evolves for each processed query metadata. Each curve is labeled by the quantity of users in the version of the dataset (i.e., from left to right: a single user, 5, 10, and 20 users). The bottom plots compare, for each of those version, the overhead of RICE-Sy, with (bottom curve) and without (top curve) the filtering modules enabled. The y axis scale is adapted between the first bottom plot and the other bottom plots. This enables us to determine the order of magnitude between the versions with a single user and the version with multiple ones, as well as comparing the overhead for the three latter bottom plots. We observe that for both case studies, the size of the ConsQHL evolves in a similar fashion, up to a final size which is equal between each version. Since all the considered queries metadata originate here from genuine blocks, dividing them among different quantities of users still lead to the same MKUs being inserted in the ConsQHL. The only difference between the versions is that those MKUs are affected to different users in the ConsQHL. We explain the small differences w.r.t. how the size changes between versions, by the quantity of MKUs associated to each query metadata. Focusing now on the overhead, we observe that it is similar for all the considered quantity of users. We explain this situation by our filtering modules which extract from ConsQHL only the relevant MKUs, independently from the quantity of users.

DA: Results & Observations The results shown in Figure 7.11 correspond to measurements realized on the first dataset generated for mHealth, and the first generated for Orange4Home, considering the DA archetype. Both Figure 7.11a and Figure 7.11b share the same layout. The four top plots display how the size of the ConsQHL evolves for each processed query metadata. To visualize queries metadata leading to attacks as vertical bar, each plot focuses on a specific version of the dataset. From left to right, the main curve corresponds to the version with a single user, with 5 users, 10 users, and 20 users. Similarly to the previous results, the bottom plots compare the overhead of RICE-Sy, with (bottom curve) and without (top curve) the filtering modules enabled.

Focusing first on mHealth, we see in Figure 7.11a that the size of the ConsQHL evolves quite differently for the DA archetype than for the GU archetype. The final size of the ConsQHL is here not equal between each version of the dataset. This comes from the presence of consolidation and queries metadata leading to attacks in the initial sequence of query metadata. When distributing

Cette thèse est accessible à l'adresse : https://theses.insa-lyon.fr/publication/2024ISAL0024/these.pdf
© [P. Lachat], [2024], INSA Lyon, tous droits réservés

them among multiple users, two consolidable queries may be associated to two distinct users. The corresponding MKUs are not consolidated in the ConsQHL. Likewise, if two queries metadata are produced by the same malicious block, then when a single user issues them, the second query metadata is detected as leading to an attack. The ConsQHL is not updated with the MKUs of this query metadata. Yet, if those two queries metadata are divided between two distinct users, then none are detected as leading to an attack. The ConsQHL is thus updated with more MKUs than for the single user. This division of the queries metadata explains why each version of the dataset does not contain the same number of queries metadata leading to an attack, or contains attacks emitted at different times. Considering now Orange4Home, Figure 7.11b depicts the stagnation already observed in presence of queries metadata leading to attacks. We observe that between the version with one and five users, the distribution of the queries metadata is drastically different, thus leading to curves increasing at different times. Finally, we remark that the overhead of RICE-Sy follows our previous observation for multiple users, even in presence of attacks.

Via this last evaluation, we conclude that, when considering a fix amount of queries metadata related to different number of users, no impact on our metrics is observed thanks to the proposed filtering modules. In the following, we discussing the impact of our observations.

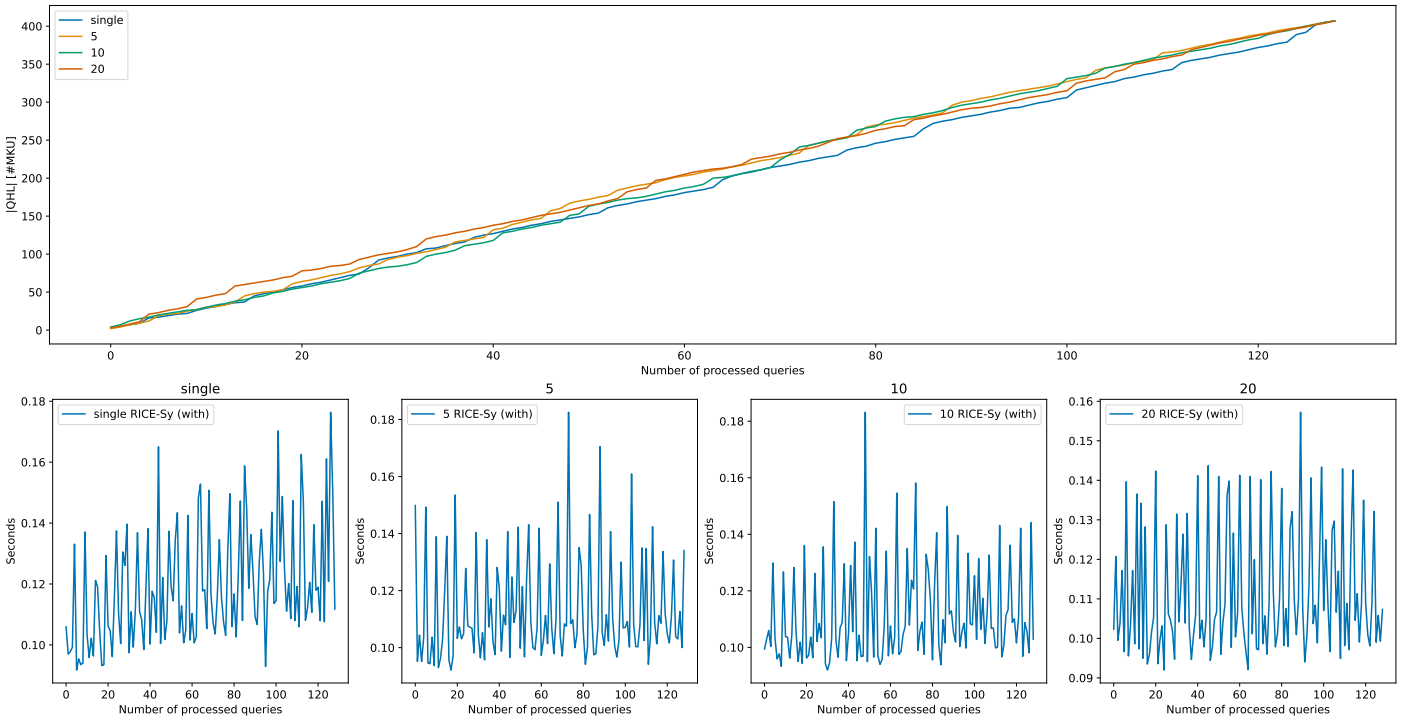
7.5 Discussion

The results obtained enable us to observe that the conceptual optimizations drastically reduce the overhead of RICE-Sy, up to 94%. We observe that considering the proposed archetypes (i.e., DA and GU), the order in which queries are issued has no significant impact on RICE-Sy. Likewise, huge quantities of consolidable queries metadata have no significant impact when queries are issued in an order allowing RICE-Sy to consolidate MKUs for each query metadata. We observe that when a user issues a query from which is extracted a large number of MKUs with time interval overlapping the interval of MKUs stored in the ConsQHL, the overhead of the consolidation module stays negligible. However, when processing those queries, the search set of the detection growths sharply compared to the other queries.

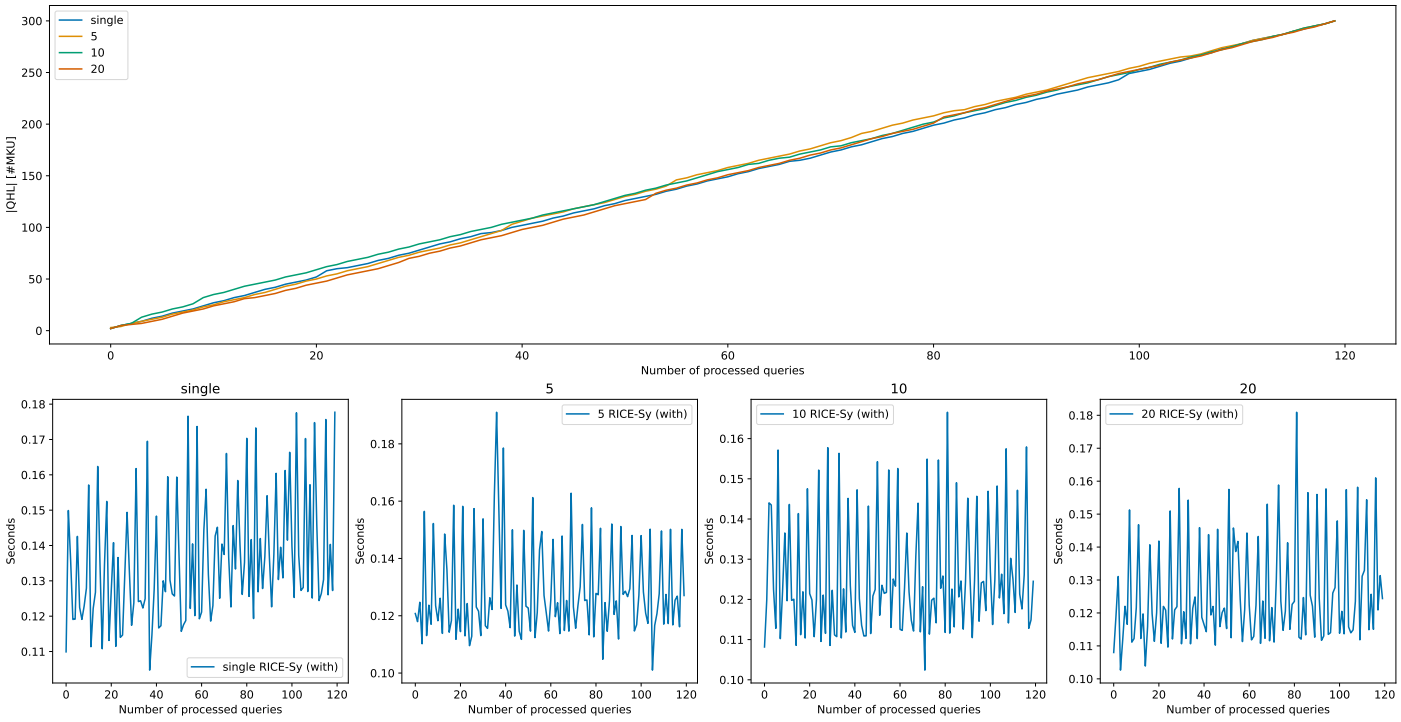
Considering the DA archetype, it seems unrealistic that such a user issues a new query to mainly select data points that they had already obtained. Besides being redundant, it would not be a discrete strategy, since the administrator of RICE-Sy would easily see a drop in the size of the ConsQHL, or a peak in the system overhead. The purpose of those queries may not be to perform an IAISD, but rather to perturb RICE-Sy, forcing it to use more computing resources than usual, or to prevent the system to process queries metadata of other users. In such a situation, a user's objective may be to *disrupt* the quality of service of the data controller via RICE-Sy. To prevent the peaks observed in Figure 7.9, the SSF module builds the search set, with only MKUs originating from the ConsQHL which were not used during the consolidation. In presence of a disrupting query metadata, the SSF module would not add any new MKUs to the search set. The detection module would only consider the output of the consolidation module, thus preventing the sharp increase of the overhead.

The last observation is that, while external factors (e.g., the quantity of users that RICE-Sy must keep track of, the presence of inference channels which require a large quantity of attributes to query, etc.) can lead to a large size of ConsQHL, over time the size of the ConsQHL affected to each user will grow for each query metadata. This continuous growth is the principal factor impacting our metrics. It is crucial to have the filtering modules as a mean to amortize the resulting increase of overhead. Indeed, while the trend is light, we observe in both Figure 7.2 and Figure 7.3 that the overhead of the QBF and SSF module increases query after query.

Cette thèse est accessible à l'adresse : https://theses.insa-lyon.fr/publication/2024ISAL0024/these.pdf
© [P. Lachat], [2024], INSA Lyon, tous droits réservés



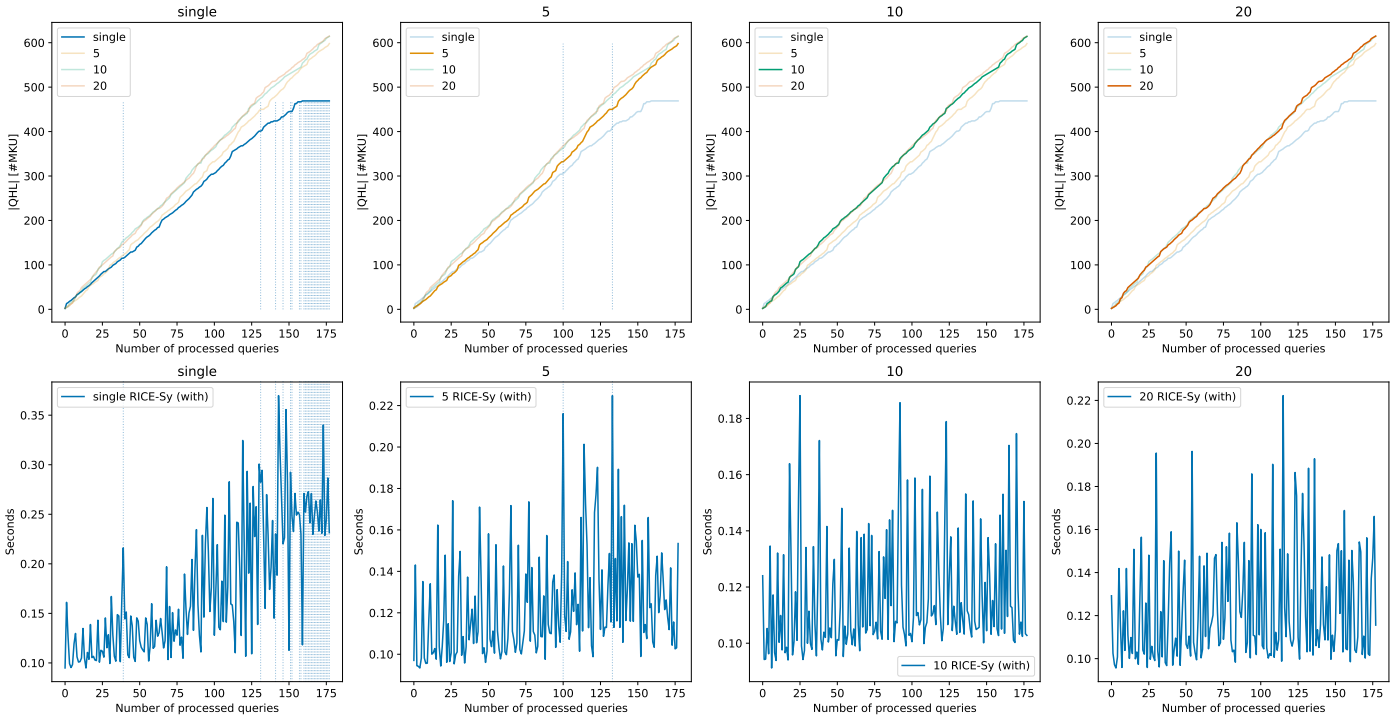
(a) mHealth



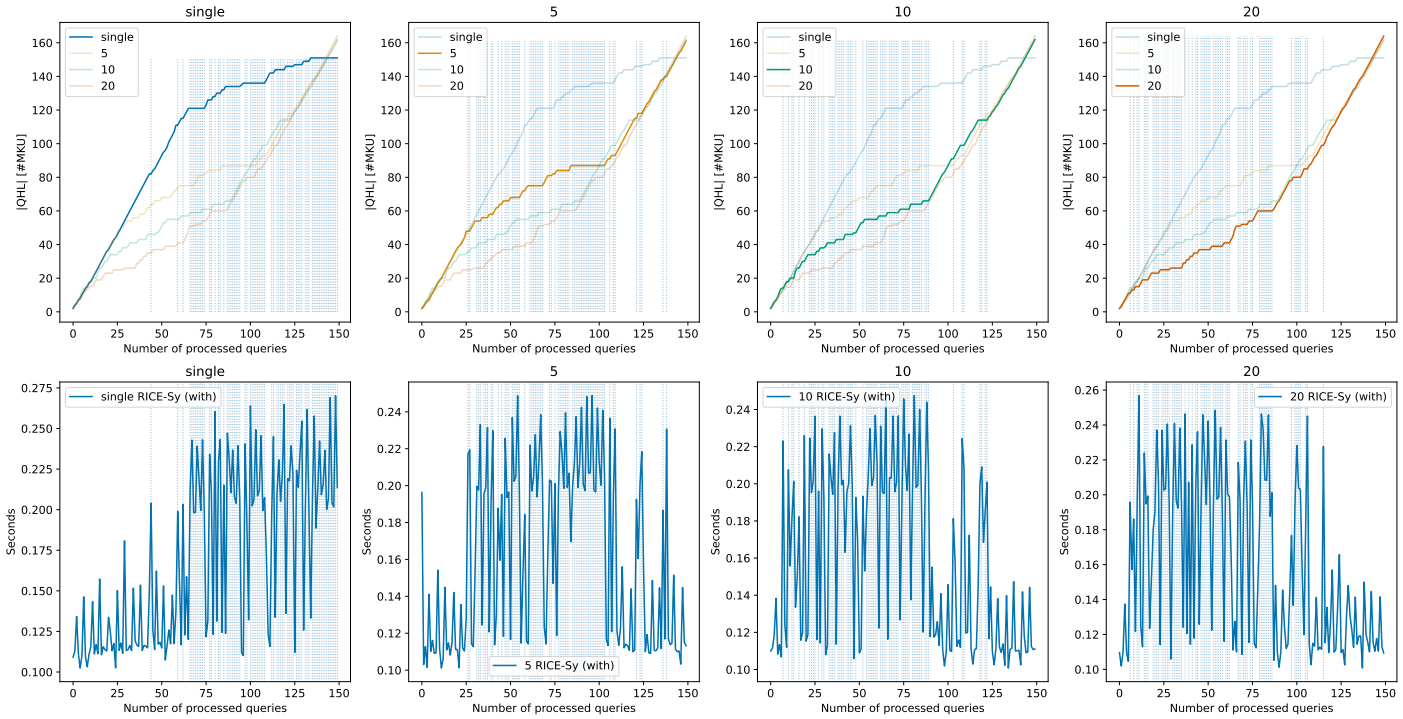
(b) Orange4Home

Figure 7.10: Impact of varying the number of users following the GU archetype for a fix quantity of queries. Measurements of the first dataset, among the 10 generated for mHealth and Orange4home.

Cette thèse est accessible à l'adresse : https://theses.insa-lyon.fr/publication/2024ISAL0024/these.pdf
© [P. Lachat], [2024], INSA Lyon, tous droits réservés



(a) mHealth



(b) Orange4Home

Figure 7.11: Impact of varying the number of users following the DA archetype for a fix quantity of queries. Measurements of the first dataset, among the 10 generated for mHealth and Orange4home. Attacks are shown as vertical dotted bar. The bottom plots display the overhead with and without the filtering modules.

Cette thèse est accessible à l'adresse : https://theses.insa-lyon.fr/publication/2024ISAL0024/these.pdf
© [P. Lachat], [2024], INSA Lyon, tous droits réservés

After a long time, this overhead may become higher than the detection module. This is especially important, since in this thesis we considered that RICE-Sy processes queries metadata one after the other. In the real world, users interact with the data controller in parallel. The time taken to perform the detection on a query may impact the query answer time of other users, thus reducing the data controller's QS. Next, we conclude this chapter by providing a summary of the observations obtained via our evaluations.

7.6 Conclusion

In this chapter, we have presented the evaluation of RICE-M based inference detection System (RICE-Sy) based on the generator proposed in Chapter 6. We have considered two metrics the *overhead* of RICE-Sy (i.e., the time elapsed between the reception of a query and the notification sent to the data controller) and the *size* of the ConsQHL. We have performed the following five evaluations which aim to:

- Determine the impact of our conceptual optimizations. We observe that the conceptual optimization we propose in Section 5.2.5 enables to reduce the overhead of RICE-Sy by $\approx$ 90% for both the GU and DA archetype, and our two case studies having different data prerequisites. Moreover, we observed that caching consolidated MKUs in ConsQHL reduce the quantity of accessed MKUs up to 73%. While considering inference channels with other specificities than our case studies can create variation in those results, it should be negligible in a setting where users continuously issues queries.
- Study the impact of the order in which queries are issued. We observe that the order in which queries are issued has no significant impact (i.e., $\leq$ 5%) on the global overhead, for both case studies.
- Observe in which situation the consolidation impacts significantly RICE-Sy. We observe that, when consolidable queries are issued in the order they are generated, the change in overhead in insignificant, for both case studies. In an extreme case where a large chunk (i.e., 250 as depicted in Figure 7.9a for the ConsQHL size plot) of MKUs are consolidated at once, we observe peaks of overhead. Yet, this is not caused by the overhead of the consolidation module. It is the SSF module which creates a search set containing redondant information.
- Discover how issuing the same metrics divided between multiple users impacts our metrics. We observe that for an equal quantity of queries metadata divided among different users, the quantity of user has not impact on the overhead of RICE-Sy, for both the GU and the DA archetype.

The observations validate the efficiency of the proposed conceptual optimizations and demonstrates the feasibility of detecting IAISDs at query-time using RICE-Sy. While the conceptual optimizations reduce the overhead of our system, they are not enough to guarantee that after a long period of detection, the overhead of RICE-Sy does not become larger and larger. In the following chapter, we conclude this thesis by providing a summary of our contributions and the short and long term future works.

Cette thèse est accessible à l'adresse : https://theses.insa-lyon.fr/publication/2024ISAL0024/these.pdf
© [P. Lachat], [2024], INSA Lyon, tous droits réservés

Chapter 8

Conclusion and perspectives

We conclude this thesis by providing an overview of our work and its future research perspectives.

8.1 Conclusion

In Chapter 1, we introduced the threat of the Inference Attack Involving Sensor Data (IAISD) and motivated the need of detection systems to tackle it. In Chapter 2, to depict the challenge of detecting IAISDs, we illustrated the diversity of data mining algorithms exploiting sensor data and heterogeneity of inferred personal information. In Chapter 3, we reviewed the literature of inference attack detection systems, based on our presented taxonomy. This discussion highlighted the lack of query-time detection system for IAISDs. In Chapter 4, we presented Raw sensor data based Inference ChannEl Model (RICE-M), a two parts model which enable capturing users' knowledge obtained by querying sensor databases, as well as the condition of disclosure this knowledge must met to enable the inferring a personal information. In Chapter 5, we introduced RICE-M based inference detection System (RICE-Sy), our detection system which leverage the captured information to detect IAISDs attempt at query-time. In addition, we proposed two conceptual optimizations to improve the detection time of RICE-Sy. In Chapter 6, we proposed a generator capable of producing controlled and realistic datasets to evaluate RICE-Sy. Finally, in Chapter 7, we performed multiple evaluations to show the impact of our optimizations and on different realistic setting of datasets. We summarize our contributions in the following sections.

RICE-M

Modeling both the users' knowledge obtained upon querying a sensor database, and the inference channels that may be exploited with this knowledge, is mandatory to detect inference attacks at query-time. While multiple models have been proposed in the literature, they face the following limitations w.r.t. the IAISDs: they represent the exact queried data and they do not capture conditions of disclosure associated to data mining algorithms. To tackle those limitations, we proposed RICE-M. It enables modeling inference channels based on the condition of disclosure implied by constraints of data mining algorithms, and the inferrable knowledge that a user gains when performing an IAISD. Through two case studies, namely *mHealth* and *Orange4Home*, we have demonstrated how such conditions, related to sliding windows and varying sensor deployment environments, can be effectively modeled using this framework. Moreover, RICE-M allows representing user's knowledge as queries metadata, i.e., information extracted from a query such as the selected attributes, the identity of the individual sharing the selected sensor data, the duration and quantity of selected data points, etc. Our model is thus capable of

Cette thèse est accessible à l'adresse : https://theses.insa-lyon.fr/publication/2024ISAL0024/these.pdf
© [P. Lachat], [2024], INSA Lyon, tous droits réservés

effectively capturing and representing the intricacies of different data mining algorithms usage scenarios.

RICE-Sy

To overcome the lack of detection systems for IAISDs, we proposed RICE-M based inference detection System (RICE-Sy) as a system capable of detecting IAISDs during query processing, by utilizing the metadata representation proposed by RICE-M. It is composed of two main modules:

- The *Knowledge Storage* which keeps track of the knowledge obtained by users via the Query History Log (QHL) and the inference channels attackers may attempt to exploit
- The *Reasoner* which detects IAISDs for each new query metadata by: using the *Query Based Filtering (QBF)* module to extract from the QHL the user's knowledge that can be combined with the new query knowledge, all units of knowledge are merged by the *consolidation* module to obtain a search set on which the detection can be performed, to complete this search set the Reasoner utilizes the *Search Set Filtering (SSF)* module to retrieve from the QHL the remaining knowledge, finally the *detection* module is used to determine if this set of knowledge enables a user to exploit one of the inference channels registered in the Knowledge Storage.

We have conducted the evaluation of RICE-Sy by utilizing two metrics, namely the *overhead* (i.e., the time elapsed between query reception and notification sent to the data controller) and the *size* of the QHL. We have performed the following four evaluations, resulting in multiple observations:

- Impact of QBF and SSF: We have observed that the proposed conceptual optimizations significantly reduce the overhead of RICE-Sy by approximately 90%, for both the GU and the DA archetypes, across the our two case studies having different data prerequisites.
- Impact of query order for the GU and DA archetype: This experiment illustrates that the order in which malicious and genuine queries are issued has no significant impact, as the observed difference of overhead is within 5% for both mHealth and Orange4Home.
- Observe significant consolidation impacts on RICE-Sy: It is observed that when consolidable queries are issued in the order they are generated, the variation in overhead is negligible. In an extreme case where a large number of MKUs are consolidated at once, occasional peaks of overhead occur. However, this is not caused by the overhead of the consolidation module itself, but rather by SSF containing redundant information.
- Explore the impact of fixed queries divided between multiple users: We have noticed that, for an equal quantity of queries divided among different users, the more users are considered, the smaller the overhead becomes. The conceptual optimizations remain effective in this situation, reducing the overhead by 24% and 18% for the GU and DA archetypes, respectively, when queries are divided between 20 users.

The results validate the efficiency of the proposed conceptual optimizations and showcase the feasibility of detecting IAISDs at query-time using RICE-Sy.

Cette thèse est accessible à l'adresse : https://theses.insa-lyon.fr/publication/2024ISAL0024/these.pdf
© [P. Lachat], [2024], INSA Lyon, tous droits réservés

Query metadata sequences generator

Due to the lack of existing works addressing the detection of IAISDs, no datasets enable evaluating RICE-Sy. Although it is possible to extend existing benchmarks of sensor databases to obtain datasets containing queries metadata, it would necessitate modifications which are not directly useful for the evaluation of RICE-Sy. Consequently, we proposed a generator to directly produce suitable queries metadata datasets. To guide creation of realistic sequences of queries metadata, we have analyzed inference attack strategies and the nature of sensor databases from which sensor data are queried, identifying different querying behaviors. Based on this analysis, we have defined three querying archetypes followed by users: the One-time Attacker (OA), the Genuine User (GU), and the Deceptive Attacker (DA). To control this generation according to the archetypes, we used the two main concepts of: *blocks* to create queries metadata which lead, or not, to an IAISD; and *periods* to simulate behaviors of regularly selected attributes. Our generator produces query metadata sequences that satisfy the necessary requirements for evaluating RICE-Sy.

8.2 Future research perspectives

In environments where both profile and sensor data are collected and exchanged, it is important to protect individuals' privacy by detecting personal information obtained from sensor data. In this thesis, we have investigated this problem and proposed, to the best of our knowledge, the first system that tackles the detection of IAISDs. Our contributions are limited to assumptions which must be challenged and results that needs to be further improved, in both the short and long term.

In the following, we describe research directions related to: (i) The generalization of modeled constraints (ii) A trust-less interaction with RICE-Sy (iii) The optimization of RICE-Sy (iv) The detection based on fuzzy goals (v) The detection considering both profile and sensor data.

Reducing the overhead beyond conceptual optimizations

According to Grzesik et al. [76], selection queries on time series databases are executed in 80 ms on average. Until now, we have considered that the queries metadata extracted from issued queries are processed one after the other by RICE-Sy. However, in a realistic setting distinct users issue their queries in parallel. A short term objective is to analyze in this parallel setting, the impact of RICE-Sy overhead on the average query time answer time. The challenge of this perspective is to find the adequate setup in which RICE-Sy can be evaluated according to this setting and the requirement of leveraging our generated datasets.

Furthermore, a long term perspective is to optimize our system further than the proposed conceptual optimizations. A first direction is to study how to parallelize the detection using multiple instances of RICE-Sy and how the users could be dispatched between instances according to their profiled querying behaviors. The challenge stems here from the definition of suitable features to profile users based on their issued queries only. Another challenge to address is related to the criterion that must determine how queries metadata would be dispatched between instances. A second direction aims to reduce the overhead by decreasing the quantity of queries RICE-Sy processes at query-time. To do so, by collaborating with the data controller, our system may sort queries metadata between the one for which the detection must be perform online, i.e., upon being received, and the less critical queries which may be put aside to perform the detection offline, e.g., when the data controller receives less queries. In case RICE-Sy detects an IAISD during the offline period, the data controller cannot prevent the query answer anymore. The main

Cette thèse est accessible à l'adresse : https://theses.insa-lyon.fr/publication/2024ISAL0024/these.pdf
© [P. Lachat], [2024], INSA Lyon, tous droits réservés

challenge is to determine which heuristics must be use to perform this trade-off between reducing the overhead at query-time and the quantity of detection offline. We have presented in [104] a set of preliminary architectural optimizations w.r.t. two directions: the first approach aims at partitioning the ConsQHL into smaller search set based on logical criteria, so that when RICE-Sy processes a query the two filtering modules only process a part of the ConsQHL thus reducing the computation time. The second approach focuses on reducing the number of detection performed at query-time by buffering some queries metadata for a latter processing.

Reasoning on profile databases and sensor databases

Detecting IAISDs is relevant when the non-sensitive personal information obtained can be combined with external information to infer a sensitive profile data. Multiple challenges are related to this direction:

- The models used to represent users' knowledge obtained from profile databases and sensor database are heterogeneous. To reason on inference attacks which combine both information sources, one must find how the user's knowledge obtained from the selection of sensor data impact this user's knowledge on the profile database, and vice versa. Assuming that this model represents profile data as random variables, a direction is to consider incorporating the inferrable knowledge detected by RICE-Sy as a *soft evidence* [118] on the corresponding random variable.
- The same information could be coded differently in two distinct databases (e.g., in one database the first and last name of an individual are stored as a single attribute while in the second database the first name and the last name are represented as two distinct attributes). This issue is known as the *record linkage problem* [36, 153] in the literature and is well studied.
- When databases are managed by distinct services, the inference detection task can be delegated to a centralized system [103]. This generates an additional issue, the external inference detection system needs to have access to (i) the database schema to be able to build a data model that captures all the data dependencies among attributes, which implies disclosing the database structures and attributes. It also requires (ii) that the user's knowledge is managed at the external inference detection system level. Data controllers may refuse disclosing data those data to a centralized detection system. Instead, the detection must be performed in a decentralized way using techniques to preserve each data controller privacy (e.g., using multi-party sharing mechanisms). We have presented in [132] a preliminary multi-base context in which detection is performed using a decentralized system for profile databases.

Trust-less interactions with RICE-Sy

The primary purpose of RICE-Sy is to detect non-sensitive personal information inferred by the users. Our system operates under the assumption that the data controller provides identities of individuals within the environment where queried sensors are deployed. These identities may be represented by any value as long as they uniquely identify individuals. This ensures that RICE-Sy can reason about metadata related to sensor data produced for the same individuals. Furthermore, when notifying the data controller, RICE-Sy includes the inferred data and the corresponding individuals' identities linked to that information. This assumption implicitly entails that the data controller trusts RICE-Sy with sharing such sensitive information about its customers.

Cette thèse est accessible à l'adresse : https://theses.insa-lyon.fr/publication/2024ISAL0024/these.pdf
© [P. Lachat], [2024], INSA Lyon, tous droits réservés

Sharing the identities of individuals with RICE-Sy discloses: (i) The fact that an individual's data is collected from different data streams. (ii) The physical observations collected for a specific individual. (iii) The linkage of an individual to other individuals, but only in environments with multiple inhabitants. The first set of information does not provide any details about the true identity of individuals or their physical locations, as it depends on how the data controller represents the data, either as one data stream per sensor or one data stream for multiple wearable or ambient sensors. However, in the case of ambient sensors, RICE-Sy might be able to deduce partially the type of environment an individual resides in (e.g., home, office) based on the selected attributes. This could lead to attribute disclosure if individuals have not consented to sharing this information with RICE-Sy. Furthermore, this information, when combined with external knowledge, might lead to re-identification of individuals. For example, if the physical observations correspond to usual ambient sensors deployed in a smart-home, and an individual is linked to a large number of other individuals in the same environment, external public census datasets could be leveraged to identify a family with the same number of people, potentially leading to re-identification. This risk arises when a single data stream provides data produced by ambient sensors deployed in an area with multiple individuals.

To mitigate this risk, the strong assumption of sharing identities can be relaxed while still allowing the system to detect inferences. The data controller may share with RICE-Sy only the unique identifier of the selected data stream. In this setting, RICE-Sy can still consider the data that originates from the same data stream and detect inferences accordingly. When an inference is detected, RICE-Sy notifies the data controller with the identifier of the data stream. The data controller can then determine for which individuals' the user obtained information. However, if the mHealth attributes are divided among two data streams, RICE-Sy cannot detect inferences performed by combining data selected from distinct streams.

Via our two case studies, we have considered environment with a single individual. In mHealth, only wearable sensors are considered. In Orange4Home, only a single individual inhabits the smart-home. The disclosure (iii) cannot happen in this setting. However, case studies such as smart-building with multiple dwellers are common in the literature and needs to be considered. An essential research direction is to determine how the data controller can share information with an instance of RICE-Sy, without leading to the disclosures (ii) and (iii). For example, attribute names may need to be obfuscated to prevent RICE-Sy from inferring the type of environment based on the semantics of the names. The challenge is that this transformation must be synchronized with the inference channels' descriptions to enable correct reasoning by RICE-Sy.

8.3 Publications

Journal

- Paul Lachat et al. "Detecting Inference Attacks Involving Raw Sensor Data: A Case Study". In: *Sensors* 22.21 (21 Jan. 2022), p. 8140. DOI: 10.3390/s22218140
- Paul Lachat et al. "Detecting Inference Attacks Involving Sensor Data in a Multi-Database Context: Issues & Challenges". In: *Internet Technology Letters* 5.6 (2022), e387. DOI: 10.1002/itl2.387

Conference

- Sad Rafik et al. "Towards a Distributed Inference Detection System in a Multi-Database Context". In: *2022 IEEE 46th Annual Computers, Software, and Applications Conference*

Cette thèse est accessible à l'adresse : https://theses.insa-lyon.fr/publication/2024ISAL0024/these.pdf
© [P. Lachat], [2024], INSA Lyon, tous droits réservés

(COMPSAC). 2022 IEEE 46th Annual Computers, Software, and Applications Conference (COMPSAC). June 2022, pp. 1550–1554. DOI: 10.1109/COMPSAC54236.2022.00246

- Paul Lachat, Veronika Rehn-Sonigo, and Nadia Bennani. "Towards an Inference Detection System Against Multi-database Attacks". In: *New Trends in Databases and Information Systems*. Ed. by Jérôme Darmont, Boris Novikov, and Robert Wrembel. Communications in Computer and Information Science. Cham: Springer International Publishing, 2020, pp. 199–209. DOI: 10.1007/978-3-030-54623-6_18

Cette thèse est accessible à l'adresse : https://theses.insa-lyon.fr/publication/2024ISAL0024/these.pdf
© [P. Lachat], [2024], INSA Lyon, tous droits réservés

Bibliography

[1] *A Face Is Exposed for AOL Searcher No. 4417749 - The New York Times*. Nov. 15, 2022. URL: https://web.archive.org/web/20221115102520/https://www.nytimes.com/2006/08/09/technology/09aol.html (visited on 05/24/2023).

[2] Abbas Acar et al. "WACA: Wearable-Assisted Continuous Authentication". In: *2018 IEEE Security and Privacy Workshops (SPW)*. 2018 IEEE Security and Privacy Workshops (SPW). May 2018, pp. 264–269. DOI: 10.1109/SPW.2018.00042.

[3] Reza Akhavian and Amir H. Behzadan. "Smartphone-Based Construction Workers' Activity Recognition and Classification". In: *Automation in Construction* 71 (Nov. 1, 2016), pp. 198–209. DOI: 10.1016/j.autcon.2016.08.015.

[4] Hande Alemdar et al. "ARAS Human Activity Datasets in Multiple Homes with Multiple Residents". In: *2013 7th International Conference on Pervasive Computing Technologies for Healthcare and Workshops*. 2013 7th International Conference on Pervasive Computing Technologies for Healthcare and Workshops. May 2013, pp. 232–235. DOI: 10.4108/icst.pervasivehealth.2013.252120.

[5] James F. Allen. "Maintaining Knowledge about Temporal Intervals". In: *Communications of the ACM* 26.11 (Nov. 1, 1983), pp. 832–843. DOI: 10.1145/182.358434.

[6] Theodoros Anagnostopoulos et al. "Environmental Exposure Assessment Using Indoor/Outdoor Detection on Smartphones". In: *Personal and Ubiquitous Computing* 21.4 (Aug. 1, 2017), pp. 761–773. DOI: 10.1007/s00779-017-1028-y.

[7] Davide Anguita et al. "A Public Domain Dataset for Human Activity Recognition Using Smartphones". In: (2013). ISSN: 978-2-87419-081-0.

[8] Damla Arifoglu and Abdelhamid Bouchachia. "Activity Recognition and Abnormal Behaviour Detection with Recurrent Neural Networks". In: *Procedia Computer Science*. 14th International Conference on Mobile Systems and Pervasive Computing (MobiSPC 2017) / 12th International Conference on Future Networks and Communications (FNC 2017) / Affiliated Workshops 110 (Jan. 1, 2017), pp. 86–93. DOI: 10.1016/j.procs.2017.06.121.

[9] Siddharth Arora et al. "High Accuracy Discrimination of Parkinson's Disease Participants from Healthy Controls Using Smartphones". In: *2014 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*. 2014 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP). May 2014, pp. 3641–3644. DOI: 10.1109/ICASSP.2014.6854280.

[10] Imran Ashraf, Soojung Hur, and Yongwan Park. "Enhancing Performance of Magnetic Field Based Indoor Localization Using Magnetic Patterns from Multiple Smartphones". In: *Sensors* 20.9 (9 Jan. 2020), p. 2704. DOI: 10.3390/s20092704.

Cette thèse est accessible à l'adresse : https://theses.insa-lyon.fr/publication/2024ISAL0024/these.pdf
© [P. Lachat], [2024], INSA Lyon, tous droits réservés

[11] Yusra Asim et al. "Context-Aware Human Activity Recognition (CAHAR) in-the-Wild Using Smartphone Accelerometer". In: *IEEE Sensors Journal* 20.8 (Apr. 2020), pp. 4361–4371. DOI: 10.1109/JSEN.2020.2964278.

[12] Sara Atske. *Americans and Privacy: Concerned, Confused and Feeling Lack of Control Over Their Personal Information*. Pew Research Center: Internet, Science & Tech. Nov. 15, 2019. URL: https://web.archive.org/web/20230524004828/https://www.pewresearch.org/internet/2019/11/15/americans-and-privacy-concerned-confused-and-feeling-lack-of-control-over-their-personal-information/ (visited on 05/24/2023).

[13] Sangwon Bae et al. "Detecting Drinking Episodes in Young Adults Using Smartphone-based Sensors". In: *Proceedings of the ACM on interactive, mobile, wearable and ubiquitous technologies* 1.2 (June 2017), p. 5. DOI: 10.1145/3090051.

[14] Serkan Ballı, Ensar Arif Sağbaş, and Musa Peker. "A Mobile Solution Based on Soft Computing for Fall Detection". In: *Mobile Solutions and Their Usefulness in Everyday Life*. Ed. by Sara Paiva. EAI/Springer Innovations in Communication and Computing. Cham: Springer International Publishing, 2019, pp. 275–294. DOI: 10.1007/978-3-319-93491-4_14.

[15] Oresti Banos et al. "Design, Implementation and Validation of a Novel Open Framework for Agile Development of Mobile Health Applications". In: *BioMedical Engineering OnLine* 14.2 (Aug. 13, 2015), S6. DOI: 10.1186/1475-925X-14-S2-S6.

[16] Mohammad Mahdi Bejani and Mehdi Ghatee. "Convolutional Neural Network With Adaptive Regularization to Classify Driving Styles on Smartphones". In: *IEEE Transactions on Intelligent Transportation Systems* 21.2 (Feb. 2020), pp. 543–552. DOI: 10.1109/TITS.2019.2896672.

[17] Abderrahmen Belfkih, Claude Duvallet, and Bruno Sadeg. "A Survey on Wireless Sensor Network Databases". In: *Wireless Networks* 25.8 (Nov. 1, 2019), pp. 4921–4946. DOI: 10.1007/s11276-019-02070-y.

[18] Yoshua Bengio, Yann Lecun, and Geoffrey Hinton. "Deep Learning for AI". In: *Communications of the ACM* 64.7 (June 21, 2021), pp. 58–65. DOI: 10.1145/3448250.

[19] Joachim Biskup. "Dynamic Policy Adaptation for Inference Control of Queries to a Propositional Information System". In: *Journal of Computer Security* 20.5 (Jan. 1, 2012), pp. 509–546. DOI: 10.3233/JCS-2012-0450.

[20] Joachim Biskup. "Inference-Proof Monotonic Query Evaluation and View Generation Reconsidered". In: *Data and Applications Security and Privacy XXXIV*. Ed. by Anoop Singhal and Jaideep Vaidya. Lecture Notes in Computer Science. Cham: Springer International Publishing, 2020, pp. 79–99. DOI: 10.1007/978-3-030-49669-2_5.

[21] Vincent D. Blondel et al. "Fast Unfolding of Communities in Large Networks". In: *Journal of Statistical Mechanics: Theory and Experiment* 2008.10 (Oct. 2008), P10008. DOI: 10.1088/1742-5468/2008/10/P10008.

[22] Felix Bölz et al. "HUMMUS: A Linked, Healthiness-Aware, User-centered and Argument-Enabling Recipe Data Set for Recommendation". In: *Proceedings of the 17th ACM Conference on Recommender Systems*. RecSys '23. New York, NY, USA: Association for Computing Machinery, Sept. 14, 2023, pp. 1–11. DOI: 10.1145/3604915.3609491.

[23] A. G. Bonomi et al. "Aspects of Activity Behavior as a Determinant of the Physical Activity Level". In: *Scandinavian Journal of Medicine & Science in Sports* 22.1 (Feb. 2012), pp. 139–145. DOI: 10.1111/j.1600-0838.2010.01130.x.

Cette thèse est accessible à l'adresse : https://theses.insa-lyon.fr/publication/2024ISAL0024/these.pdf
© [P. Lachat], [2024], INSA Lyon, tous droits réservés

[24] Hendrio Bragança et al. "A Smartphone Lightweight Method for Human Activity Recognition Based on Information Theory". In: *Sensors* 20.7 (7 Jan. 2020), p. 1856. DOI: 10.3390/s20071856.

[25] A. Brodsky, C. Farkas, and S. Jajodia. "Secure Databases: Constraints, Inference Channels, and Monitoring Disclosures". In: *IEEE Transactions on Knowledge and Data Engineering* 12.6 (Nov. 2000), pp. 900–919. DOI: 10.1109/69.895801.

[26] Tânia Carvalho et al. "Survey on Privacy-Preserving Techniques for Microdata Publication". In: *ACM Computing Surveys* (Mar. 28, 2023). DOI: 10.1145/3588765.

[27] Fernando E. Casado et al. "Walking Recognition in Mobile Devices". In: *Sensors* 20.4 (4 Jan. 2020), p. 1189. DOI: 10.3390/s20041189.

[28] Abdelberi Chaabane, Gergely Acs, and Mohamed Ali Kaafar. "You Are What You Like! Information Leakage Through Users' Interests". In: NDSS Symposium 2012 - 19th Annual Network and Distributed System Security Symposium. Feb. 5, 2012, p. 1.

[29] Hao Chen, Seung Hyun Cha, and Tae Wan Kim. "A Framework for Group Activity Detection and Recognition Using Smartphone Sensors and Beacons". In: *Building and Environment* 158 (July 1, 2019), pp. 205–216. DOI: 10.1016/j.buildenv.2019.05.016.

[30] Xihui Chen et al. "Active Re-identification Attacks on Periodically Released Dynamic Social Graphs". In: *Computer Security – ESORICS 2020: 25th European Symposium on Research in Computer Security, ESORICS 2020, Guildford, UK, September 14–18, 2020, Proceedings, Part II*. Berlin, Heidelberg: Springer-Verlag, Sept. 14, 2020, pp. 185–205. DOI: 10.1007/978-3-030-59013-0_10.

[31] Yu Chen and Wesley W. Chu. "Protection of Database Security via Collaborative Inference Detection". In: *IEEE Transactions on Knowledge and Data Engineering* 20.8 (Aug. 2008), pp. 1013–1027. DOI: 10.1109/TKDE.2007.190642.

[32] Girija Chetty, Matthew White, and Farnaz Akther. "Smart Phone Based Data Mining for Human Activity Recognition". In: *Procedia Computer Science*. Proceedings of the International Conference on Information and Communication Technologies, ICICT 2014, 3-5 December 2014 at Bolgatty Palace & Island Resort, Kochi, India 46 (Jan. 1, 2015), pp. 1181–1187. DOI: 10.1016/j.procs.2015.01.031.

[33] Belkacem Chikhaoui and Frank Gouineau. "Towards Automatic Feature Extraction for Activity Recognition from Wearable Sensors: A Deep Learning Approach". In: *2017 IEEE International Conference on Data Mining Workshops (ICDMW)*. 2017 IEEE International Conference on Data Mining Workshops (ICDMW). Nov. 2017, pp. 693–702. DOI: 10.1109/ICDMW.2017.97.

[34] Daeseon Choi et al. "Private Attribute Inference from Facebook's Public Text Metadata: A Case Study of Korean Users". In: *Industrial Management & Data Systems* 117.8 (Jan. 1, 2017), pp. 1687–1706. DOI: 10.1108/IMDS-07-2016-0276.

[35] Kah Meng Chong. "Privacy-Preserving Healthcare Informatics: A Review". In: *ITM Web of Conferences* 36 (2021), p. 04005. DOI: 10.1051/itmconf/20213604005.

[36] Peter Christen. "A Survey of Indexing Techniques for Scalable Record Linkage and Deduplication". In: *IEEE Transactions on Knowledge and Data Engineering* 24.9 (Sept. 2012), pp. 1537–1555. DOI: 10.1109/TKDE.2011.127.

Cette thèse est accessible à l'adresse : https://theses.insa-lyon.fr/publication/2024ISAL0024/these.pdf
© [P. Lachat], [2024], INSA Lyon, tous droits réservés

[37] Shanice Clarke, Luis G. Jaimes, and Miguel A. Labrador. "mStress: A Mobile Recommender System for Just-in-Time Interventions for Stress". In: *2017 14th IEEE Annual Consumer Communications & Networking Conference (CCNC)*. 2017 14th IEEE Annual Consumer Communications & Networking Conference (CCNC). Jan. 2017, pp. 1–5. DOI: 10.1109/CCNC.2017.8015367.

[38] *Consumer Electronic Sensors Market Size, Share | Industry Report 2022*. 2022. URL: https://web.archive.org/web/20230603145142/https://www.grandviewresearch.com/industry-analysis/consumer-electronic-sensors-market (visited on 05/24/2023).

[39] Matthew Crain. "The Limits of Transparency: Data Brokers and Commodification". In: *New Media & Society* 20.1 (Jan. 1, 2018), pp. 88–104. DOI: 10.1177/1461444816657096.

[40] Federico Cruciani et al. "Personalizing Activity Recognition With a Clustering Based Semi-Population Approach". In: *IEEE Access* 8 (2020), pp. 207794–207804. DOI: 10.1109/ACCESS.2020.3038084.

[41] Julien Cumin. *Orange4Home: A Dataset of Routine Daily Activities in an Instrumented Home – Amiqual4Home*. May 12, 2017. URL: http://amiqual4home.inria.fr/en/orange4home/ (visited on 11/22/2023).

[42] Julien Cumin et al. "A Dataset of Routine Daily Activities in an Instrumented Home". In: *Ubiquitous Computing and Ambient Intelligence*. Ed. by Sergio F. Ochoa, Pritpal Singh, and José Bravo. Lecture Notes in Computer Science. Cham: Springer International Publishing, 2017, pp. 413–425. DOI: 10.1007/978-3-319-67585-5_43.

[43] Julien Cumin et al. "Human Activity Recognition Using Place-Based Decision Fusion in Smart Homes". In: *Modeling and Using Context*. Ed. by Patrick Brézillon, Roy Turner, and Carlo Penco. Lecture Notes in Computer Science. Cham: Springer International Publishing, 2017, pp. 137–150. DOI: 10.1007/978-3-319-57837-8_11.

[44] Julien Cumin et al. "Inferring Availability for Communication in Smart Homes Using Context". In: *2018 IEEE International Conference on Pervasive Computing and Communications Workshops (PerCom Workshops)*. 2018 IEEE International Conference on Pervasive Computing and Communications Workshops (PerCom Workshops). Mar. 2018, pp. 1–6. DOI: 10.1109/PERCOMW.2018.8480091.

[45] George Danezis et al. *Privacy and Data Protection by Design - from Policy to Engineering*. 2014. DOI: 10.2824/38623.

[46] Alberto Calatroni Daniel Roggen. *OPPORTUNITY Activity Recognition*. UCI Machine Learning Repository, 2010. DOI: 10.24432/C5M027.

[47] Delan Devakumar et al. "Racism and Discrimination in COVID-19 Responses". In: *The Lancet* 395.10231 (Apr. 11, 2020), p. 1194. DOI: 10.1016/S0140-6736(20)30792-3.

[48] Anton Dries et al. "ProbLog2: Probabilistic Logic Programming". In: *Machine Learning and Knowledge Discovery in Databases*. Ed. by Albert Bifet et al. Lecture Notes in Computer Science. Cham: Springer International Publishing, 2015, pp. 312–315. DOI: 10.1007/978-3-319-23461-8_37.

[49] Cynthia Dwork. "Differential Privacy". In: *Automata, Languages and Programming*. Ed. by Michele Bugliesi et al. Lecture Notes in Computer Science. Berlin, Heidelberg: Springer, 2006, pp. 1–12. DOI: 10.1007/11787006_1.

[50] Markus Ebner et al. "Recognition of Typical Locomotion Activities Based on the Sensor Data of a Smartphone in Pocket or Hand". In: *Sensors* 20.22 (22 Jan. 2020), p. 6559. DOI: 10.3390/s20226559.

Cette thèse est accessible à l'adresse : https://theses.insa-lyon.fr/publication/2024ISAL0024/these.pdf
© [P. Lachat], [2024], INSA Lyon, tous droits réservés

[51] Günther Eibl and Dominik Engel. "Influence of Data Granularity on Smart Meter Privacy". In: *IEEE Transactions on Smart Grid* 6.2 (Mar. 2015), pp. 930–939. DOI: 10.1109/TSG.2014.2376613.

[52] Jihane el Mokhtari et al. "Coupling of Inference and Access Controls to Ensure Privacy Protection". In: *International Journal of Safety and Security Engineering* 11 (Oct. 31, 2021), pp. 529–535. DOI: 10.18280/ijsse.110504.

[53] *ENISA Threat Landscape 2020 - Insider Threat*. ENISA. 2020. URL: https://web.archive.org/web/20230603135508/https://www.enisa.europa.eu/publications/insider-threat/ (visited on 07/04/2023).

[54] Aghil Esmaeili Kelishomi et al. "Mobile User Indoor-Outdoor Detection through Physical Daily Activities". In: *Sensors* 19.3 (3 Jan. 2019), p. 511. DOI: 10.3390/s19030511.

[55] Labiba Gillani Fahad, Syed Fahad Tahir, and Muttukrishnan Rajarajan. "Feature Selection and Data Balancing for Activity Recognition in Smart Homes". In: *2015 IEEE International Conference on Communications (ICC)*. 2015 IEEE International Conference on Communications (ICC). June 2015, pp. 512–517. DOI: 10.1109/ICC.2015.7248373.

[56] Hongqing Fang, Raghavendiran Srinivasan, and Diane Cook. "Feature Selections for Human Activity Recognition in Smart Home Environments". In: *International Journal of Innovative Computing, Information and Control* 8 (May 1, 2012), pp. 3525–3535. ISSN: 349-419.

[57] Csilla Farkas and Sushil Jajodia. "The Inference Problem: A Survey". In: *ACM SIGKDD Explorations Newsletter* 4.2 (Dec. 1, 2002), pp. 6–11. DOI: 10.1145/772862.772864.

[58] Müge Fazlioglu. *IAPP-EY Annual Privacy Governance Report 2021*. 2021. DOI: 10.2139/ssrn.4227244. preprint.

[59] Anna Ferrari et al. "On the Personalization of Classification Models for Human Activity Recognition". In: *IEEE Access* 8 (2020), pp. 32066–32079. DOI: 10.1109/ACCESS.2020.2973425.

[60] *Fitness Trackers | Shop Fitbit*. 2020. URL: https://web.archive.org/web/20230704031747/https://www.fitbit.com/global/us/products/trackers (visited on 05/24/2023).

[61] Denis Foo Kune and Yongdae Kim. "Timing Attacks on PIN Input Devices". In: *Proceedings of the 17th ACM Conference on Computer and Communications Security*. CCS '10. New York, NY, USA: Association for Computing Machinery, Oct. 4, 2010, pp. 678–680. DOI: 10.1145/1866307.1866395.

[62] Matt Fredrikson, Somesh Jha, and Thomas Ristenpart. "Model Inversion Attacks That Exploit Confidence Information and Basic Countermeasures". In: *Proceedings of the 22nd ACM SIGSAC Conference on Computer and Communications Security*. CCS '15. New York, NY, USA: Association for Computing Machinery, Oct. 12, 2015, pp. 1322–1333. DOI: 10.1145/2810103.2813677.

[63] Lorenzo Frigerio et al. "Differentially Private Generative Adversarial Networks for Time Series, Continuous, and Discrete Open Data". In: *ICT Systems Security and Privacy Protection*. Ed. by Gurpreet Dhillon et al. IFIP Advances in Information and Communication Technology. Cham: Springer International Publishing, 2019, pp. 151–164. DOI: 10.1007/978-3-030-22312-0_11.

[64] Biying Fu et al. "Fitness Activity Recognition on Smartphones Using Doppler Measurements". In: *Informatics* 5.2 (2 June 2018), p. 24. DOI: 10.3390/informatics5020024.

Cette thèse est accessible à l'adresse : https://theses.insa-lyon.fr/publication/2024ISAL0024/these.pdf
© [P. Lachat], [2024], INSA Lyon, tous droits réservés

[65] Mohamed Medhat Gaber, Arkady Zaslavsky, and Shonali Krishnaswamy. "Mining Data Streams: A Review". In: *ACM SIGMOD Record* 34.2 (June 1, 2005), pp. 18–26. DOI: 10.1145/1083784.1083789.

[66] João Gama. "A Survey on Learning from Data Streams: Current and Future Trends". In: *Progress in Artificial Intelligence* 1.1 (Apr. 1, 2012), pp. 45–55. DOI: 10.1007/s13748-011-0002-6.

[67] Bernhard Ganter and Rudolf Wille. *Formal Concept Analysis: Mathematical Foundations*. Springer Science & Business Media, Dec. 6, 2012. 289 pp. ISBN: 978-3-642-59830-2.

[68] Daniel Garcia-Gonzalez et al. "A Public Domain Dataset for Real-Life Human Activity Recognition Using Smartphone Sensors". In: *Sensors* 20.8 (8 Jan. 2020), p. 2200. DOI: 10.3390/s20082200.

[69] *General Data Protection Regulation*. 2016. URL: https://web.archive.org/web/20230425072253/https://eur-lex.europa.eu/eli/reg/2016/679/2016-05-04 (visited on 05/24/2023).

[70] Armin Gerl et al. "LPL, Towards a GDPR-Compliant Privacy Language: Formal Definition and Usage". In: *Transactions on Large-Scale Data- and Knowledge-Centered Systems XXXVII*. Ed. by Abdelkader Hameurlain and Roland Wagner. Lecture Notes in Computer Science. Berlin, Heidelberg: Springer, 2018, pp. 41–80. DOI: 10.1007/978-3-662-57932-9_2.

[71] Lise Getoor et al. "Learning Probabilistic Relational Models". In: *Relational Data Mining*. Ed. by Sašo Džeroski and Nada Lavrač. Berlin, Heidelberg: Springer, 2001, pp. 307–335. DOI: 10.1007/978-3-662-04599-2_13.

[72] Martin Gjoreski et al. "Classical and Deep Learning Methods for Recognizing Human Activities and Modes of Transportation with Smartphone Sensors". In: *Information Fusion* 62 (Oct. 1, 2020), pp. 47–62. DOI: 10.1016/j.inffus.2020.04.004.

[73] *Global Companies Collecting Personal Data by Region 2021*. Statista. 2023. URL: https://web.archive.org/web/20230719145505/https://www.statista.com/statistics/1172965/firms-collecting-personal-data/ (visited on 05/24/2023).

[74] Imad Gohar et al. "Person Re-Identification Using Deep Modeling of Temporally Correlated Inertial Motion Patterns". In: *Sensors* 20.3 (3 Jan. 2020), p. 949. DOI: 10.3390/s20030949.

[75] Lukasz Golab and M. Tamer Zsu. *Data Stream Management*. Morgan & Claypool Publishers, May 2010. 80 pp. ISBN: 978-1-60845-272-9.

[76] Piotr Grzesik and Dariusz Mrozek. "Comparative Analysis of Time Series Databases in the Context of Edge Computing for Low Power Sensor Networks". In: *Computational Science – ICCS 2020* 12141 (May 25, 2020), pp. 371–383. DOI: 10.1007/978-3-030-50426-7_28.

[77] Marco Guarnieri, Srdjan Marinovic, and David Basin. "Securing Databases from Probabilistic Inference". In: *2017 IEEE 30th Computer Security Foundations Symposium (CSF)*. 2017 IEEE 30th Computer Security Foundations Symposium (CSF). Aug. 2017, pp. 343–359. DOI: 10.1109/CSF.2017.30.

[78] Peeyush Gupta et al. "SmartBench: A Benchmark for Data Management in Smart Spaces". In: *Proceedings of the VLDB Endowment* 13.12 (July 1, 2020), pp. 1807–1820. DOI: 10.14778/3407790.3407791.

Cette thèse est accessible à l'adresse : https://theses.insa-lyon.fr/publication/2024ISAL0024/these.pdf
© [P. Lachat], [2024], INSA Lyon, tous droits réservés

[79] Abdul Hakim et al. "Smartphone Based Data Mining for Fall Detection: Analysis and Design". In: *Procedia Computer Science*. 2016 IEEE International Symposium on Robotics and Intelligent Sensors, IRIS 2016, 17-20 December 2016, Tokyo, Japan 105 (Jan. 1, 2017), pp. 46–51. DOI: 10.1016/j.procs.2017.01.188.

[80] Brian A. Harris-Kojetin et al. *Statistical Policy Working Paper 22: Report on Statistical Disclosure Limitation Methodology*. Research Report. U.S. Federal Committee on Statistical Methodology, 2005.

[81] Alexander Hart et al. "Using Smartphone Sensor Paradata and Personalized Machine Learning Models to Infer Participants' Well-being: Ecological Momentary Assessment". In: *Journal of Medical Internet Research* 24.4 (Apr. 28, 2022), e34015. DOI: 10.2196/34015.

[82] Mohammed Mehedi Hassan et al. "A Robust Human Activity Recognition System Using Smartphone Sensors and Deep Learning". In: *Future Generation Computer Systems* 81 (Apr. 1, 2018), pp. 307–313. DOI: 10.1016/j.future.2017.11.029.

[83] Peter Hevesi et al. "Monitoring Household Activities and User Location with a Cheap, Unobtrusive Thermal Sensor Array". In: *Proceedings of the 2014 ACM International Joint Conference on Pervasive and Ubiquitous Computing*. UbiComp '14. New York, NY, USA: Association for Computing Machinery, Sept. 13, 2014, pp. 141–145. DOI: 10.1145/2632048.2636084.

[84] Walt Hickey. *How Americans Like Their Steak*. FiveThirtyEight. May 16, 2014. URL: https://fivethirtyeight.com/features/how-americans-like-their-steak/ (visited on 11/05/2023).

[85] Ivan Homoliak et al. "Insight Into Insiders and IT: A Survey of Insider Threat Taxonomies, Analysis, Modeling, and Countermeasures". In: *ACM Computing Surveys* 52.2 (Apr. 2, 2019), 30:1–30:40. DOI: 10.1145/3303771.

[86] M. Shamim Hossain and Ghulam Muhammad. "Emotion Recognition Using Secure Edge and Cloud Computing". In: *Information Sciences* 504 (Dec. 1, 2019), pp. 589–601. DOI: 10.1016/j.ins.2019.07.040.

[87] Hongsheng Hu et al. "Membership Inference Attacks on Machine Learning: A Survey". In: *ACM Computing Surveys* 54 (11s Sept. 9, 2022), 235:1–235:37. DOI: 10.1145/3523273.

[88] Vijay S. Iyengar. "Transforming Data to Satisfy Privacy Constraints". In: *Proceedings of the Eighth ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*. KDD '02. New York, NY, USA: Association for Computing Machinery, July 23, 2002, pp. 279–288. DOI: 10.1145/775047.775089.

[89] Adel Jebali, Salma Sassi, and Abderrazak Jemai. "Inference Control in Distributed Environment: A Comparison Study". In: *Risks and Security of Internet and Systems*. Ed. by Slim Kallel et al. Lecture Notes in Computer Science. Cham: Springer International Publishing, 2020, pp. 69–83. DOI: 10.1007/978-3-030-41568-6_5.

[90] Adel Jebali et al. "Secure Data Outsourcing in Presence of the Inference Problem: A Graph-Based Approach". In: *Journal of Parallel and Distributed Computing* 160 (Feb. 1, 2022), pp. 1–15. DOI: 10.1016/j.jpdc.2021.09.006.

[91] Jinyuan Jia and Neil Zhenqiang Gong. "Defending Against Machine Learning Based Inference Attacks via Adversarial Examples: Opportunities and Challenges". In: *Adaptive Autonomous Secure Cyber Systems*. Ed. by Sushil Jajodia et al. Cham: Springer International Publishing, 2020, pp. 23–40. DOI: 10.1007/978-3-030-33432-1_2.

Cette thèse est accessible à l'adresse : https://theses.insa-lyon.fr/publication/2024ISAL0024/these.pdf
© [P. Lachat], [2024], INSA Lyon, tous droits réservés

[92] Mahdi Khosravy et al. "Model Inversion Attack by Integration of Deep Generative Models: Privacy-Sensitive Face Generation From a Face Recognition System". In: *IEEE Transactions on Information Forensics and Security* 17 (2022), pp. 357–372. DOI: 10.1109/TIFS.2022.3140687.

[93] Sunder Ali Khowaja, Bernardo Nugroho Yahya, and Seok-Lyong Lee. "Hierarchical Classification Method Based on Selective Learning of Slacked Hierarchy for Activity Recognition Systems". In: *Expert Systems with Applications* 88 (Dec. 1, 2017), pp. 165–177. DOI: 10.1016/j.eswa.2017.06.040.

[94] Jong Wook Kim et al. "A Survey Of Differential Privacy-Based Techniques and Their Applicability to Location-Based Services". In: *Computers & Security* 111 (Dec. 1, 2021), p. 102464. DOI: 10.1016/j.cose.2021.102464.

[95] Yong-Joong Kim et al. "Integrating Hidden Markov Models Based on Mixture-of-Templates and k-NN2 Ensemble for Activity Recognition". In: *2016 23rd International Conference on Pattern Recognition (ICPR)*. 2016 23rd International Conference on Pattern Recognition (ICPR). Dec. 2016, pp. 1636–1641. DOI: 10.1109/ICPR.2016.7899871.

[96] Itzik Klein. "Smartphone Location Recognition: A Deep Learning-Based Approach". In: *Sensors* 20.1 (1 Jan. 2020), p. 214. DOI: 10.3390/s20010214.

[97] J Kolter and Matthew Johnson. "REDD: A Public Data Set for Energy Disaggregation Research". In: *Artif. Intell.* 25 (Jan. 1, 2011).

[98] Jacob Kröger. "The Privacy-Invading Potential of Sensor Data". In: *SSRN Electronic Journal* (June 9, 2022). DOI: 10.2139/ssrn.4362987.

[99] Jacob Kröger. "Unexpected Inferences from Sensor Data: A Hidden Privacy Threat in the Internet of Things". In: *Internet of Things. Information Processing in an Increasingly Connected World*. Ed. by Leon Strous and Vinton G. Cerf. IFIP Advances in Information and Communication Technology. Cham: Springer International Publishing, 2019, pp. 147–159. DOI: 10.1007/978-3-030-15651-0_13.

[100] Yasuhiko Kubota et al. "Physical Activity and Lifetime Risk of Cardiovascular Disease and Cancer". In: *Medicine & Science in Sports & Exercise* 49.8 (Aug. 2017), p. 1599. DOI: 10.1249/MSS.0000000000001274.

[101] Lucia Kvapilova et al. "Continuous Sound Collection Using Smartphones and Machine Learning to Measure Cough". In: *Digital Biomarkers* 3.3 (2019), pp. 166–175. DOI: 10.1159/000504666.

[102] Emiro De-La-Hoz-Franco et al. "Sensor-Based Datasets for Human Activity Recognition – A Systematic Review of Literature". In: *IEEE Access* 6 (2018), pp. 59192–59210. DOI: 10.1109/ACCESS.2018.2873502.

[103] Paul Lachat, Veronika Rehn-Sonigo, and Nadia Bennani. "Towards an Inference Detection System Against Multi-database Attacks". In: *New Trends in Databases and Information Systems*. Ed. by Jérôme Darmont, Boris Novikov, and Robert Wrembel. Communications in Computer and Information Science. Cham: Springer International Publishing, 2020, pp. 199–209. DOI: 10.1007/978-3-030-54623-6_18.

[104] Paul Lachat et al. "Detecting Inference Attacks Involving Raw Sensor Data: A Case Study". In: *Sensors* 22.21 (21 Jan. 2022), p. 8140. DOI: 10.3390/s22218140.

[105] Paul Lachat et al. "Detecting Inference Attacks Involving Sensor Data in a Multi-Database Context: Issues & Challenges". In: *Internet Technology Letters* 5.6 (2022), e387. DOI: 10.1002/itl2.387.

Cette thèse est accessible à l'adresse : https://theses.insa-lyon.fr/publication/2024ISAL0024/these.pdf
© [P. Lachat], [2024], INSA Lyon, tous droits réservés

[106] Kangjae Lee and Mei-Po Kwan. "Physical Activity Classification in Free-Living Conditions Using Smartphone Accelerometer Data and Exploration of Predicted Results". In: *Computers, Environment and Urban Systems* 67 (Jan. 1, 2018), pp. 124–131. DOI: 10.1016/j.compenvurbsys.2017.09.012.

[107] Jiexing Li, Yufei Tao, and Xiaokui Xiao. "Preservation of Proximity Privacy in Publishing Numerical Sensitive Data". In: *Proceedings of the 2008 ACM SIGMOD International Conference on Management of Data*. SIGMOD '08. New York, NY, USA: Association for Computing Machinery, June 9, 2008, pp. 473–486. DOI: 10.1145/1376616.1376666.

[108] Xiao Li et al. "PSDRNN: An Efficient and Effective HAR Scheme Based on Feature Extraction and Deep Learning". In: *IEEE Transactions on Industrial Informatics* 16.10 (Oct. 2020), pp. 6703–6713. DOI: 10.1109/TII.2020.2968920.

[109] Jie Lian et al. "EchoSpot: Spotting Your Locations via Acoustic Sensing". In: *Proceedings of the ACM on Interactive, Mobile, Wearable and Ubiquitous Technologies* 5.3 (Sept. 14, 2021), 113:1–113:21. DOI: 10.1145/3478095.

[110] Gaoyang Liu et al. "SocInf: Membership Inference Attacks on Social Media Health Data With Machine Learning". In: *IEEE Transactions on Computational Social Systems* 6.5 (Oct. 2019), pp. 907–921. DOI: 10.1109/TCSS.2019.2916086.

[111] Jinyi Liu et al. "WiPhone: Smartphone-based Respiration Monitoring Using Ambient Reflected WiFi Signals". In: *Proceedings of the ACM on Interactive, Mobile, Wearable and Ubiquitous Technologies* 5.1 (Mar. 30, 2021), 23:1–23:19. DOI: 10.1145/3448092.

[112] Xiangyu Liu et al. "When Good Becomes Evil: Keystroke Inference with Smartwatch". In: *Proceedings of the 22nd ACM SIGSAC Conference on Computer and Communications Security*. CCS '15. New York, NY, USA: Association for Computing Machinery, Oct. 12, 2015, pp. 1273–1285. DOI: 10.1145/2810103.2813668.

[113] Miguel Martínez del Horno, Luis Orozco-Barbosa, and Ismael García-Varea. "A Smartphone-Based Multimodal Indoor Tracking System". In: *Information Fusion* 76 (Dec. 1, 2021), pp. 36–45. DOI: 10.1016/j.inffus.2021.05.001.

[114] Alban Maxhuni et al. "Unobtrusive Stress Assessment Using Smartphones". In: *IEEE Transactions on Mobile Computing* 20.6 (June 2021), pp. 2313–2325. DOI: 10.1109/TMC.2020.2974834.

[115] Javier E. Meseguer et al. "DrivingStyles: A Smartphone Application to Assess Driver Behavior". In: *2013 IEEE Symposium on Computers and Communications (ISCC)*. 2013 IEEE Symposium on Computers and Communications (ISCC). July 2013, pp. 000535–000540. DOI: 10.1109/ISCC.2013.6755001.

[116] Fen Miao et al. "Identifying Typical Physical Activity on Smartphone with Varying Positions and Orientations". In: *BioMedical Engineering OnLine* 14.1 (Apr. 13, 2015), p. 32. DOI: 10.1186/s12938-015-0026-4.

[117] Jalal Mostafa et al. "SciTS: A Benchmark for Time-Series Databases in Scientific Experiments and Industrial Internet of Things". In: *Proceedings of the 34th International Conference on Scientific and Statistical Database Management*. SSDBM '22. New York, NY, USA: Association for Computing Machinery, Aug. 23, 2022, pp. 1–11. DOI: 10.1145/3538712.3538723.

[118] Ali Ben Mrad et al. "An Explication of Uncertain Evidence in Bayesian Networks: Likelihood Evidence and Probabilistic Evidence". In: *Applied Intelligence* 43.4 (Dec. 1, 2015), pp. 802–824. DOI: 10.1007/s10489-015-0678-6.

Cette thèse est accessible à l'adresse : https://theses.insa-lyon.fr/publication/2024ISAL0024/these.pdf
© [P. Lachat], [2024], INSA Lyon, tous droits réservés

[119] Debadyuti Mukherjee et al. "EnsemConvNet: A Deep Learning Approach for Human Activity Recognition Using Smartphone Sensors for Healthcare Applications". In: *Multimedia Tools and Applications* 79.41 (Nov. 1, 2020), pp. 31663–31690. DOI: 10.1007/s11042-020-09537-7.

[120] Arvind Narayanan and Vitaly Shmatikov. *How To Break Anonymity of the Netflix Prize Dataset*. Nov. 22, 2007. DOI: 10.48550/arXiv.cs/0610105. URL: http://arxiv.org/abs/cs/0610105 (visited on 05/24/2023). preprint.

[121] Anubhav Natani, Abhishek Sharma, and Thinagaran Perumal. "Sequential Neural Networks for Multi-Resident Activity Recognition in Ambient Sensing Smart Homes". In: *Applied Intelligence* 51.8 (Aug. 1, 2021), pp. 6014–6028. DOI: 10.1007/s10489-020-02134-z.

[122] Tobias Nef et al. "Evaluation of Three State-of-the-Art Classifiers for Recognition of Activities of Daily Living from Smart Home Ambient Data". In: *Sensors (Basel, Switzerland)* 15.5 (May 21, 2015), pp. 11725–11740. DOI: 10.3390/s150511725.

[123] Amir Noury and Morteza Amini. "An Access and Inference Control Model for Time Series Databases". In: *Future Generation Computer Systems* 92 (Mar. 1, 2019), pp. 93–108. DOI: 10.1016/j.future.2018.09.057.

[124] Rafael Garcia Oresti Banos. *MHEALTH Dataset*. UCI Machine Learning Repository, 2014. DOI: 10.24432/C5TW22.

[125] Primal Pappachan et al. "Don't Be a Tattle-Tale: Preventing Leakages through Data Dependencies on Access Control Protected Data". In: *Proceedings of the VLDB Endowment* 15.11 (July 1, 2022), pp. 2437–2449. DOI: 10.14778/3551793.3551805.

[126] Yangheran Piao, Kai Ye, and Xiaohui Cui. "Privacy Inference Attack Against Users in Online Social Networks: A Literature Review". In: *IEEE Access* 9 (2021), pp. 40417–40431. DOI: 10.1109/ACCESS.2021.3064208.

[127] Meikel Poess et al. "Analysis of TPCx-IoT: The First Industry Standard Benchmark for IoT Gateway Systems". In: *2018 IEEE 34th International Conference on Data Engineering (ICDE)*. 2018 IEEE 34th International Conference on Data Engineering (ICDE). Apr. 2018, pp. 1519–1530. DOI: 10.1109/ICDE.2018.00170.

[128] Apostolos Pyrgelis, Carmela Troncoso, and Emiliano De Cristofaro. "Measuring Membership Privacy on Aggregate Location Time-Series". In: *Proceedings of the ACM on Measurement and Analysis of Computing Systems* 4.2 (June 12, 2020), 36:1–36:28. DOI: 10.1145/3392154.

[129] X. Qian et al. "Detection and Elimination of Inference Channels in Multilevel Relational Database Systems". In: *Proceedings 1993 IEEE Computer Society Symposium on Research in Security and Privacy*. Proceedings 1993 IEEE Computer Society Symposium on Research in Security and Privacy. May 1993, pp. 196–205. DOI: 10.1109/RISP.1993.287632.

[130] Xiaolei Qian and T.F. Lunt. "A Semantic Framework of the Multilevel Secure Relational Model". In: *IEEE Transactions on Knowledge and Data Engineering* 9.2 (Mar. 1997), pp. 292–301. DOI: 10.1109/69.591453.

[131] Jing Qiu et al. "A Survey on Access Control in the Age of Internet of Things". In: *IEEE Internet of Things Journal* 7.6 (June 2020), pp. 4682–4696. DOI: 10.1109/JIOT.2020.2969326.

Cette thèse est accessible à l'adresse : https://theses.insa-lyon.fr/publication/2024ISAL0024/these.pdf
© [P. Lachat], [2024], INSA Lyon, tous droits réservés

[132] Sad Rafik et al. "Towards a Distributed Inference Detection System in a Multi-Database Context". In: *2022 IEEE 46th Annual Computers, Software, and Applications Conference (COMPSAC)*. 2022 IEEE 46th Annual Computers, Software, and Applications Conference (COMPSAC). June 2022, pp. 1550–1554. DOI: 10.1109/COMPSAC54236.2022.00246.

[133] Daniele Ravì et al. "A Deep Learning Approach to On-Node Sensor Data Analytics for Mobile or Wearable Devices". In: *IEEE Journal of Biomedical and Health Informatics* 21.1 (Jan. 2017), pp. 56–64. DOI: 10.1109/JBHI.2016.2633287.

[134] *RDC - Confidentiality Training - Disclosure*. Jan. 14, 2019–. URL: https://web.archive.org/web/20230719150828/https://www.cdc.gov/rdc/b4confidisc/training/Confident413.htm (visited on 07/13/2023).

[135] Charissa Ann Ronao and Sung-Bae Cho. "Human Activity Recognition with Smartphone Sensors Using Deep Learning Neural Networks". In: *Expert Systems with Applications* 59 (Oct. 15, 2016), pp. 235–244. DOI: 10.1016/j.eswa.2016.04.032.

[136] Farida Sabry et al. "Towards On-Device Dehydration Monitoring Using Machine Learning from Wearable Device's Data". In: *Sensors* 22.5 (5 Jan. 2022), p. 1887. DOI: 10.3390/s22051887.

[137] Ensar Arif Sağbaş, Serdar Korukoglu, and Serkan Balli. "Stress Detection via Keyboard Typing Behaviors by Using Smartphone Sensors and Machine Learning Techniques". In: *Journal of Medical Systems* 44.4 (Feb. 17, 2020), p. 68. DOI: 10.1007/s10916-020-1530-z.

[138] P. Samarati. "Protecting Respondents Identities in Microdata Release". In: *IEEE Transactions on Knowledge and Data Engineering* 13.6 (Nov. 2001), pp. 1010–1027. DOI: 10.1109/69.971193.

[139] Mokhtar Sellami, Mohand-Said Hacid, and Mohamed Mohsen Gammoudi. "A FCA Framework for Inference Control in Data Integration Systems". In: *Distributed and Parallel Databases* 37.4 (Dec. 1, 2019), pp. 543–586. DOI: 10.1007/s10619-018-7241-5.

[140] Ahmad Shahi, Brendon J. Woodford, and Hanhe Lin. "Dynamic Real-Time Segmentation and Recognition of Activities Using a Multi-feature Windowing Approach". In: *Trends and Applications in Knowledge Discovery and Data Mining*. Ed. by U. Kang et al. Lecture Notes in Computer Science. Cham: Springer International Publishing, 2017, pp. 26–38. DOI: 10.1007/978-3-319-67274-8_3.

[141] Anshul Sharma et al. "Early Transportation Mode Detection Using Smartphone Sensing Data". In: *IEEE Sensors Journal* 21.14 (July 2021), pp. 15651–15659. DOI: 10.1109/JSEN.2020.3009312.

[142] Seyed Vahab Shojaedini and Mohamad Javad Beirami. "Mobile Sensor Based Human Activity Recognition: Distinguishing of Challenging Activities by Applying Long Short-Term Memory Deep Learning Modified by Residual Network Concept". In: *Biomedical Engineering Letters* 10.3 (Aug. 2020), pp. 419–430. DOI: 10.1007/s13534-020-00160-x.

[143] Raphael Spreitzer. "PIN Skimming: Exploiting the Ambient-Light Sensor in Mobile Devices". In: *Proceedings of the 4th ACM Workshop on Security and Privacy in Smartphones & Mobile Devices*. SPSM '14. New York, NY, USA: Association for Computing Machinery, Nov. 7, 2014, pp. 51–62. DOI: 10.1145/2666620.2666622.

[144] Jessica Staddon. "Dynamic Inference Control". In: *Proceedings of the 8th ACM SIGMOD Workshop on Research Issues in Data Mining and Knowledge Discovery*. DMKD '03. New York, NY, USA: Association for Computing Machinery, June 13, 2003, pp. 94–100. DOI: 10.1145/882082.882103.

Cette thèse est accessible à l'adresse : https://theses.insa-lyon.fr/publication/2024ISAL0024/these.pdf
© [P. Lachat], [2024], INSA Lyon, tous droits réservés

[145] Marcin Straczkiewicz, Peter James, and Jukka-Pekka Onnela. "A Systematic Review of Smartphone-Based Human Activity Recognition Methods for Health Research". In: *npj Digital Medicine* 4.1 (1 Oct. 18, 2021), pp. 1–15. DOI: 10.1038/s41746-021-00514-4.

[146] T.-A. Su and G. Ozsoyoglu. "Controlling FD and MVD Inferences in Multilevel Relational Database Systems". In: *IEEE Transactions on Knowledge and Data Engineering* 3.4 (Dec. 1991), pp. 474–485. DOI: 10.1109/69.109108.

[147] Priyank Sunhare, Rameez R. Chowdhary, and Manju K. Chattopadhyay. "Internet of Things and Data Mining: An Application Oriented Survey". In: *Journal of King Saud University - Computer and Information Sciences* 34 (6, Part B June 1, 2022), pp. 3569–3590. DOI: 10.1016/j.jksuci.2020.07.002.

[148] Latanya Sweeney. "K-Anonymity: A Model for Protecting Privacy". In: *International Journal of Uncertainty, Fuzziness and Knowledge-Based Systems* 10.5 (Oct. 1, 2002), pp. 557–570. DOI: 10.1142/S0218488502001648.

[149] "The Best Smart Home Devices to Help Aging in Place". In: *The New York Times* (Mar. 30, 2023). ISSN: 0362-4331.

[150] Tyrone S. Toland, Csilla Farkas, and Caroline M. Eastman. "The Inference Problem: Maintaining Maximal Availability in the Presence of Database Updates". In: *Computers & Security* 29.1 (Feb. 1, 2010), pp. 88–103. DOI: 10.1016/j.cose.2009.07.004.

[151] Nick Tsalis et al. "A Taxonomy of Side Channel Attacks on Critical Infrastructures and Relevant Systems". In: *Critical Infrastructure Security and Resilience: Theories, Methods, Tools and Technologies*. Ed. by Dimitris Gritzalis, Marianthi Theocharidou, and George Stergiopoulos. Advanced Sciences and Technologies for Security Applications. Cham: Springer International Publishing, 2019, pp. 283–313. DOI: 10.1007/978-3-030-00024-0_15.

[152] Yonatan Vaizman, Katherine Ellis, and Gert Lanckriet. "Recognizing Detailed Human Context in the Wild from Smartphones and Smartwatches". In: *IEEE Pervasive Computing* 16.4 (Oct. 2017), pp. 62–74. DOI: 10.1109/MPRV.2017.3971131.

[153] Dinusha Vatsalan, Peter Christen, and Vassilios S. Verykios. "A Taxonomy of Privacy-Preserving Record Linkage Techniques". In: *Information Systems* 38.6 (Sept. 1, 2013), pp. 946–969. DOI: 10.1016/j.is.2012.11.005.

[154] Parvathy Venkatachalam and Sanjog Ray. "How Do Context-Aware Artificial Intelligence Algorithms Used in Fitness Recommender Systems? A Literature Review and Research Agenda". In: *International Journal of Information Management Data Insights* 2.2 (Nov. 1, 2022), p. 100139. DOI: 10.1016/j.jjimei.2022.100139.

[155] Robert-Andrei Voicu et al. "Human Physical Activity Recognition Using Smartphone Sensors". In: *Sensors* 19.3 (3 Jan. 2019), p. 458. DOI: 10.3390/s19030458.

[156] Bao Wang, Linjie Gao, and Zhicai Juan. "Travel Mode Detection Using GPS Data and Socioeconomic Attributes Based on a Random Forest Classifier". In: *IEEE Transactions on Intelligent Transportation Systems* 19.5 (May 2018), pp. 1547–1558. DOI: 10.1109/TITS.2017.2723523.

[157] Liang Wang et al. "Recognizing Multi-User Activities Using Wearable Sensors in a Smart Home". In: *Pervasive and Mobile Computing*. Knowledge-Driven Activity Recognition in Intelligent Environments 7.3 (June 1, 2011), pp. 287–298. DOI: 10.1016/j.pmcj.2010.11.008.

Cette thèse est accessible à l'adresse : https://theses.insa-lyon.fr/publication/2024ISAL0024/these.pdf
© [P. Lachat], [2024], INSA Lyon, tous droits réservés

[158] Gary Weiss. *WISDM Smartphone and Smartwatch Activity and Biometrics Dataset*. UCI Machine Learning Repository, 2019. DOI: 10.24432/C5HK59.

[159] Gary M. Weiss and Jeffrey W. Lockhart. "Identifying User Traits by Mining Smart Phone Accelerometer Data". In: *Proceedings of the Fifth International Workshop on Knowledge Discovery from Sensor Data*. SensorKDD '11. New York, NY, USA: Association for Computing Machinery, Aug. 21, 2011, pp. 61–69. DOI: 10.1145/2003653.2003660.

[160] Raymond K. Wong and B. S. Vidyalakshmi. "Privacy Leakage via Attribute Inference in Directed Social Networks". In: *Information and Communications Security*. Ed. by Kwok-Yan Lam, Chi-Hung Chi, and Sihan Qing. Lecture Notes in Computer Science. Cham: Springer International Publishing, 2016, pp. 333–346. DOI: 10.1007/978-3-319-50011-9_26.

[161] Philip Woodall and Pearl Brereton. "A Systematic Literature Review of Inference Strategies". In: *International Journal of Information and Computer Security* 4.2 (Jan. 2010), pp. 99–117. DOI: 10.1504/IJICS.2010.034813.

[162] Yang Yang, Qiang Cao, and Hong Jiang. "EdgeDB: An Efficient Time-Series Database for Edge Computing". In: *IEEE Access* 7 (2019), pp. 142295–142307. DOI: 10.1109/ACCESS.2019.2943876.

[163] Qussai Yaseen and Brajendra Panda. "Insider Threat Mitigation: Preventing Unauthorized Knowledge Acquisition". In: *International Journal of Information Security* 11.4 (Aug. 1, 2012), pp. 269–280. DOI: 10.1007/s10207-012-0165-6.

[164] Darren Yates and Md Zahidul Islam. "Data Mining on Smartphones: An Introduction and Survey". In: *ACM Computing Surveys* 55.5 (Dec. 3, 2022), 101:1–101:38. DOI: 10.1145/3529753.

[165] Darren Yates and Md. Zahidul Islam. "Readiness of Smartphones for Data Collection and Data Mining with an Example Application in Mental Health". In: *Data Mining*. Ed. by Thuc D. Le et al. Communications in Computer and Information Science. Singapore: Springer, 2019, pp. 235–246. DOI: 10.1007/978-981-15-1699-3_19.

[166] Mingyang Zhang et al. "Indoor Localization Fusing WiFi With Smartphone Inertial Sensors Using LSTM Networks". In: *IEEE Internet of Things Journal* 8.17 (Sept. 2021), pp. 13608–13623. DOI: 10.1109/JIOT.2021.3067515.

[167] Xiao Zhang et al. "MoodExplorer: Towards Compound Emotion Detection via Smartphone Sensing". In: *Proceedings of the ACM on Interactive, Mobile, Wearable and Ubiquitous Technologies* 1.4 (Jan. 8, 2018), 176:1–176:30. DOI: 10.1145/3161414.

[168] Beidi Zhao et al. "A Framework of Combining Short-Term Spatial/Frequency Feature Extraction and Long-Term IndRNN for Activity Recognition". In: *Sensors* 20.23 (23 Jan. 2020), p. 6984. DOI: 10.3390/s20236984.

[169] Elena Zheleva and Lise Getoor. "To Join or Not to Join: The Illusion of Privacy in Social Networks with Mixed Public and Private User Profiles". In: *Proceedings of the 18th International Conference on World Wide Web*. WWW '09. New York, NY, USA: Association for Computing Machinery, Apr. 20, 2009, pp. 531–540. DOI: 10.1145/1526709.1526781.

[170] Yong Zhong et al. "A Systematic Survey of Data Mining and Big Data Analysis in Internet of Things". In: *The Journal of Supercomputing* 78.17 (Nov. 1, 2022), pp. 18405–18453. DOI: 10.1007/s11227-022-04594-1.

[171] Xiaolu Zhou, Wei Yu, and William C. Sullivan. "Making Pervasive Sensing Possible: Effective Travel Mode Sensing Based on Smartphones". In: *Computers, Environment and Urban Systems* 58 (July 1, 2016), pp. 52–59. DOI: 10.1016/j.compenvurbsys.2016.03.001.

Cette thèse est accessible à l'adresse : https://theses.insa-lyon.fr/publication/2024ISAL0024/these.pdf
© [P. Lachat], [2024], INSA Lyon, tous droits réservés

FOLIO ADMINISTRATIF

THESE DE L'INSA LYON, MEMBRE DE L'UNIVERSITE DE LYON

NOM : Lachat
(avec précision du nom de jeune fille, le cas échéant)

DATE de SOUTENANCE : 12/04/2024

Prénoms : Paul

TITRE : Detecting Inference Attack Involving Sensor Data

NATURE : Doctorat

Numéro d'ordre : 2024ISAL0024

Ecole doctorale : InfoMaths

Spécialité : Informatique

RESUME : Le partage et le traitement des informations personnelles avec les organisations sont devenus de plus en plus essentiels pour les interactions sociales. En Europe, selon le RGPD (Règlement Général sur la Protection des Données), elles doivent protéger par conception et par défaut les données collectées. Les Mécanismes de Contrôle d'Accès (MCA) sont traditionnellement utilisés pour sécuriser les systèmes d'information contre l'accès non autorisé à des données sensibles. Cependant, la disponibilité croissante de données de capteurs motive de nouveaux services à fournir des informations sur les individus. Divers algorithmes d'exploration de données ont été proposés pour inférer de nouvelles données personnelles à partir de données de capteurs. Les données sensibles peuvent être accédées indirectement à l'aide de données non sensibles, telles que des informations inférées à partir de données de capteurs. Par conséquent, le risque de contourner les MCAs pour obtenir des informations à l'aide des données de capteur existe. Nous proposons un système de détection d'inférence basé sur l'analyse des requêtes émises sur une base de données de capteurs. Les connaissances obtenu via ces requêtes, et les canaux d'inférence correspondant à l'utilisation des algorithmes d'exploration de données sur des données de capteur pour inférer les informations des individus, sont décrits grâce à Raw sensor data based Inference ChannEl Model (RICE-M). La détection est effectuée par RICE-M based inference detection System (RICE-Sy). RICE-Sy considère au moment de la requête, la connaissance qu'un utilisateur obtient via une nouvelle requête et a obtenues via son historique de requête, et détermine si cela suffit pour permettre à cet utilisateur d'exploiter un canal. Ainsi, les systèmes de protection de la vie privée peuvent tirer parti des inférences détectées par RICE-Sy, en prenant en compte les informations des individus obtenues par les attaquants via une base de données de capteurs, pour davantage protéger les données collectées.

MOTS-CLÉS : Confidentialité des données, Données de capteurs, Système de détection d'inférence, Modélisation des requêtes, Métadonnées

Laboratoire (s) de recherche : LIRIS

Directeur de thèse : Lionel Brunie et Harald Kosch

Président de jury :

Composition du jury : Frédéric Cuppens, Alexander Felfernig, Mario Döller, Salma Sassi, Michael Granitzer, Lionel Brunie, Harald Kosch, Nadia Bennani

Cette thèse est accessible à l'adresse : https://theses.insa-lyon.fr/publication/2024ISAL0024/these.pdf
© [P. Lachat], [2024], INSA Lyon, tous droits réservés