Aix*Marseille université
Socialement engagée

NNT : 2024AIXM****

THÈSE DE DOCTORAT

Soutenue à Aix-Marseille Université
le 16 décembre 2024 par

Alba MARTINEZ ANTON

Protection de la Vie Privée à travers la Formalisation de Modèles Basés sur la Provenance.

Discipline
Informatique

École doctorale
ED 184 MATHEMATIQUES ET INFORMATIQUE

Laboratoire/Partenaires de recherche
Laboratoire d'Informatique et Systèmes (LIS)

Composition du jury

Benjamin NGUYEN PR, INSA CVL, Université d'Orléans	Rapporteur
Abdessamad IMINE MCF HDR, LORIA, Université de Lorraine	Rapporteur
Mohand-Saïd HACID PR, LIRIS, Université Lyon 1	Examinateur
Yannick CHEVALIER MCF HDR, IRIT, Université Toulouse III	Examinateur
Florence SÈDES PR, IRIT, Université Toulouse III	Présidente du jury
Clara BERTOLISSI MCF HDR, LIS, Aix-Marseille Université	Directrice de thèse
Jean-Marc TALBOT PR, LIS, Aix-Marseille Université	Co-directeur de thèse

LIS
LABORATOIRE D'INFORMATIQUE & SYSTÈMES

Affidavit

Je soussigné, Alba Martinez Anton, déclare par la présente que le travail présenté dans ce manuscrit est mon propre travail, réalisé sous la direction scientifique de Clara Bertolissi et Jean-Marc Talbot, dans le respect des principes d'honnêteté, d'intégrité et de responsabilité inhérents à la mission de recherche. Les travaux de recherche et la rédaction de ce manuscrit ont été réalisés dans le respect à la fois de la charte nationale de déontologie des métiers de la recherche et de la charte d'Aix-Marseille Université relative à la lutte contre le plagiat.

Ce travail n'a pas été précédemment soumis en France ou à l'étranger dans une version identique ou similaire à un organisme examinateur.

Fait à Marseille, le 11/10/2024,

Cette œuvre est mise à disposition selon les termes de la Licence Creative Commons Attribution - Pas d'Utilisation Commerciale - Pas de Modification 4.0 International.

Bibliographie personnelle

Publications acceptées et publiées :

1. **Clara Bertolissi, Alba Martinez Anton, Romain Testud, and Nicola Zannone**. 2022. « Towards Secure Content Sharing in Social Networks ». In *CYBER 2022, The Seventh International Conference on Cyber-Technologies and Cyber-Systems, IARIA, Valencia, Spain, November 13-17, 2022*, pp. 34-35. ISBN : 978-1-61208-996-6.
2. **Clara Bertolissi, Alba Martinez Anton, and Nicola Zannone**. 2023. « Data Sharing in Social Networks. » In Proceedings of the *28th ACM Symposium on Access Control Models and Technologies (SACMAT '23)*. Association for Computing Machinery, New York, NY, USA, 181–192. https://doi.org/10.1145/3589608.3593833
3. **Alba Martinez Anton, Clara Bertolissi, and Jean-Marc Talbot**. 2024. « Solving Access Control Conflicts in Multi-User Systems ». In Proceedings of *the 21st International Conference on Security and Cryptography (SECRYPT 2024)*, S. De Capitani di Vimercati and P. Samarati (Eds.). SCITEPRESS, Dijon, France, 492–499. https://doi.org/10.5220/0012768500003767.
4. **Pauline Di Salvo-Cilia, Alba Martinez Anton, and Clara Bertolissi**. 2024.« Towards Automated Checking of GDPR Compliance ». In *CYBER 2024, The Ninth International Conference on Cyber-Technologies and Cyber-Systems, IARIA, Venice, Italy, September 29-October 03, 2024.*

Articles en cours de finition :

1. **Clara Bertolissi, Alba Martinez Anton, and Nicola Zannone.** 2024. « Data Sharing in Social Networks ». *Version journal en cours de finition.*

Résumé et mots clés

Avec l'essor d'Internet, de nombreux systèmes proposent des services numérisés, utilisant ainsi des données personnelles, ce qui suscite des préoccupations croissantes concernant la vie privée. Ce défi est accentué par la popularité de certains systèmes collaboratifs, tels que les réseaux sociaux, où la collaboration entre utilisateurs complexifie la protection des données. Ceci souligne un dilemme entre la nécessité de protéger les données personnelles et le désir de partage et d'interaction. Face à cette situation, des lois et réglementations sur la protection des données ont vu le jour pour encadrer l'utilisation de ces informations. Parallèlement, il y a une recherche croissante de modèles adaptés pour assurer la sécurisation des données dans ces environnements collaboratifs, où la multiplicité des acteurs complique la mise en place de mécanismes de protection efficaces.

L'objectif de ce travail est de proposer des mécanismes de protection adaptés basés sur la provenance des données. La provenance permet de retracer l'origine et l'historique des données, offrant ainsi un contexte précieux pour leur gestion sécurisée. Le travail s'oriente sur deux axes principaux : d'abord, le développement d'un contrôle d'accès collaboratif et d'une stratégie de gestion des conflits, conçus spécifiquement pour les systèmes de type réseaux sociaux. Ensuite, nous proposons une méthode d'audit de la conformité d'un système aux réglementations existantes, permettant de vérifier le respect à ces dernières.

Pour démontrer la faisabilité et la pertinence des modèles proposés, nous avons développé des prototypes. Ils permettent de valider concrètement nos approches et d'illustrer leur application dans des simulations d'environnements réels.

Mots clés : Vie privée, Protection des données, Provenance, Contrôle d'accès multipartite, Résolution de conflit de politiques, Conformité

Abstract and keywords

With the rise of the Internet, many systems offer digitized services, thereby using personal data and raising growing concerns about privacy. This challenge is exacerbated by the popularity of certain collaborative systems, such as social networks, where user collaboration complicates data protection. This highlights a dilemma between the need to protect personal data and the desire for sharing and interaction.

In response to this situation, data protection laws and regulations have emerged to regulate the use of this information. At the same time, there is increasing research into suitable models to ensure data security in these collaborative environments, where the multiplicity of actors complicates the implementation of effective protection mechanisms.

The aim of this work is to propose suitable protection mechanisms based on data provenance. Provenance allows for tracing the origin and history of data, thus providing valuable context for secure management. The work is focused on two main areas: first, the development of a collaborative access control system and a conflict management strategy, specifically designed for social network-type systems. Second, we propose a method for auditing a system's compliance with existing regulations, enabling verification of adherence to these regulations.

To demonstrate the feasibility and the relevence of the proposed models, we have developed prototypes. These prototypes validate our approaches in practice and illustrate their application in simulations of real-world environments.

Keywords: Privacy, Data protection, Provenance, Multi-party Access Control, Policy conflict resolution, Compliance

Remerciements

Je tiens tout d'abord à exprimer ma plus profonde gratitude à ma directrice de thèse, Madame Clara Bertolissi. Je la remercie sincèrement d'avoir su proposer un sujet de recherche en lien avec mon stage de fin de master. Son implication et sa disponibilité tout au long de ces trois dernières années ont été essentielles à l'aboutissement de cette thèse. J'ai particulièrement apprécié sa bienveillance, ses précieux conseils et l'attention qu'elle a portée à chacun de mes questionnements. Je lui suis également reconnaissante pour le temps considérable consacré à la relecture attentive de ce mémoire. Je souhaite également exprimer toute ma reconnaissance à mon co-directeur de thèse, Monsieur Jean-Marc Talbot. Son accompagnement, empreint de bienveillance et de conseils avisés, a été d'une grande valeur tout au long de ce travail. Je tiens également à le remercier pour le temps qu'il a consacré à la relecture de cette thèse.

Je tiens à exprimer toute ma gratitude à Monsieur Benjamin Nguyen et à Monsieur Abdessamad Imine pour l'honneur qu'ils m'ont fait en acceptant d'être les rapporteurs de cette thèse. Je remercie également Madame Florence Sèdes, Monsieur Mohamed-Saïd Hacid et Monsieur Yannick Chevalier pour avoir généreusement accepté de partie de mon jury de soutenance.

Je tiens à remercier le Laboratoire d'Informatique et Systèmes (LIS), en particulier l'équipe MOVE, pour leur accueil au sein de leur structure. Leur soutien et l'environnement de travail stimulant qu'ils m'ont offert ont grandement contribué à l'avancement de cette recherche. Je remercie également l'Institut Archimède pour le financement de mon contrat doctoral, sans lequel la réalisation de cette thèse n'aurait pas été possible. Je souhaite également remercier chaleureusement Monsieur Nicola Zannone pour sa collaboration dans la rédaction de plusieurs articles. Ses contributions m'ont offert un regard extérieur éclairé sur mes pistes de travail, enrichissant ainsi mes réflexions et orientant mon approche de manière constructive.

Merci à mon compagnon pour son soutien indéfectible tout au long de cette aventure. Sa présence, ses encouragements et sa compréhension ont été d'une grande aide et m'ont permis de surmonter les défis rencontrés durant cette thèse.
Enfin, merci à mes amis et ma famille, qui ont été d'un soutien inestimable tout au long de ce parcours. Leur présence réconfortante, leurs encouragements et les moments de convivialité partagés ont été essentiels pour me changer les idées et me motiver dans les moments difficiles. Merci d'avoir été là, de m'avoir soutenu(e) et d'avoir cru en moi.

Table des matières

Affidavit 2

Bibliographie personnelle 3

Résumé et mots clés 4

Abstract and keywords 5

Remerciements 6

Table des matières 7

Table des figures 10

Liste des tableaux 12

Liste des acronymes 13

Introduction Générale 15

1. Préliminaires 18
- 1.1. Introduction . . . 19
  - 1.1.1. Vie privée dans le monde numérique . . . 19
  - 1.1.2. Données personnelles . . . 21
  - 1.1.3. Défis liés au respect de la vie privée et aux données personnelles 22
- 1.2. Contexte . . . 25
  - 1.2.1. Un systèmes collaboratif basé sur la communauté : les réseaux sociaux . . . 25
  - 1.2.2. Applications web . . . 30
- 1.3. Représentation de la provenance des données . . . 33
  - 1.3.1. Modèle Ouvert de Provenance . . . 33
  - 1.3.2. Extension du Modèle Ouvert de Provenance avec la notion de propriétaire . . . 40
  - 1.3.3. Extension pour des systèmes dynamiques et multi-propriétaires 41
  - 1.3.4. Utilisations du Modèle Ouvert de Provenance . . . 44
- 1.4. Représentation de la communauté des utilisateurs . . . 45
  - 1.4.1. Graphe des relations . . . 45
  - 1.4.2. Définition de communauté . . . 48
  - 1.4.3. Pondération des arêtes pour modéliser la confiance . . . 49

1.5. Capture de la provenance et des relations pour la construction du graphe 53
1.5.1. Conversion de journaux en graphe . . . 54
1.5.2. Capture de la provenance . . . 54
1.6. Conclusion et discussion . . . 55

2. Contrôle d'accès pour le partage de données multi-propriétaires 56
2.1. Introduction . . . 56
2.2. Contrôles d'accès dans la littérature . . . 57
2.2.1. Modèles classiques . . . 58
2.2.2. Contrôles d'accès collaboratifs . . . 59
2.3. Défis des contrôles d'accès collaboratifs dans les réseaux sociaux . . . 62
2.3.1. Défis liés aux relations dynamiques . . . 64
2.3.2. Défis liés aux données multi-propriétaires . . . 65
2.3.3. Défis liés au contenu dynamique . . . 66
2.3.4. Discussion . . . 67
2.4. Modèle de contrôle d'accès collaboratif basé sur les attributs et la provenance . . . 69
2.4.1. Spécification des politiques d'accès . . . 70
2.4.2. Expression de politiques raffinées . . . 77
2.4.3. Sémantique pour l'évaluation des politiques . . . 81
2.4.4. Évaluation d'accès d'objets composés . . . 85
2.5. Prototype et validation . . . 88
2.5.1. Architecture . . . 88
2.5.2. Générateurs de graphes et de politiques . . . 89
2.5.3. Expérimentations . . . 91
2.5.4. Résultats et discussion . . . 93
2.6. Conclusion et Discussion . . . 98

3. Gestion de conflits dans les systèmes multi-utilisateurs 100
3.1. Introduction . . . 101
3.2. Stratégies de résolution de conflits dans la littérature . . . 101
3.2.1. Algorithmes de combinaison . . . 102
3.2.2. Stratégies de majorité et de consensus . . . 102
3.2.3. Résolution basée sur la théorie des jeux . . . 103
3.2.4. Résolution de conflits dans les photos . . . 103
3.2.5. Résolution par seuil . . . 104
3.2.6. Facteurs de décision . . . 105
3.3. Défis de la gestion des conflits . . . 106
3.4. Modèle de résolution de conflits basé sur un seuil . . . 109
3.4.1. Sensibilité . . . 110
3.4.2. Intérêt à partager . . . 113
3.4.3. Résolution du conflit . . . 117
3.5. Validation du modèle . . . 120
3.5.1. Variations des facteurs . . . 120

3.5.2. Analyse comparative par rapport aux modèles basés sur le seuil 121
3.5.3. Prototype . . . 123
3.6. Conclusion et Discussion . . . 126

4. Vérification automatisée de lois **128**
4.1. Introduction . . . 128
4.2. Lois relatives aux systèmes informatiques . . . 129
4.2.1. Règlement Général sur la Protection des Données . . . 130
4.2.2. Protection de données à travers le monde . . . 134
4.3. Conformité et sa vérification dans la littérature . . . 135
4.3.1. Du langage juridique au langage logique . . . 136
4.3.2. Conformité dès la conception . . . 136
4.3.3. Vérification de la conformité : audit et surveillance . . . 137
4.4. Défis de la conformité du traitement des données . . . 138
4.4.1. Traçabilité des finalités . . . 139
4.4.2. Automatisation de la vérification . . . 140
4.4.3. Réplication des données . . . 141
4.5. Utilisation de la provenance pour vérifier la conformité . . . 142
4.5.1. Graphe de provenance : nœuds et attributs pour la conformité . 143
4.5.2. Patterns pour vérifier la conformité . . . 145
4.6. Prototype et validation . . . 154
4.6.1. Architecture . . . 155
4.6.2. Validation . . . 160
4.6.3. Améliorations et travaux futurs . . . 163
4.7. Conclusion et discussion . . . 165

Conclusion **166**

ANNEXES **180**

A. Fichier d'une politique **181**

B. Prolog : syntaxe et patterns **182**
B.1. Syntaxe Prolog . . . 182
B.2. Patterns . . . 183
B.2.1. Représentation des attributs . . . 183
B.2.2. Représentation des arrêtes . . . 183

Table des figures

1.1. Réseau de relations 28
1.2. Contenu utilisateur dans un réseau social 29
1.3. Contenu d'un site web 32
1.4. Dépendances causales de l'OPM 39
1.5. Dépendance notAvailable 43
1.6. Graphe de provenance d'un graphe social 44
1.7. Graphe des relations 47
1.8. Exemple de communautés en suivant la Définition 1 49
1.9. Graphe des relations étendu avec la confiance 52

2.1. Graphe de provenance pour les exemples Sections 2.3 et 2.4 63
2.2. Différents cas pour une publication partagée 67
2.3. Architecture du prototype 89
2.4. Temps d'exécution moyen pour résoudre une requête dans le modèle sans extension temporelle 94
2.5. Temps d'exécution moyen pour résoudre une requête dans le modèle avec extension temporelle 94
2.6. Résultats de performance pour les requêtes refusées avec des politiques temporelles 95
2.7. Moyenne du temps d'évaluation des requêtes autorisées pour un nombre variable d'éléments associés par artefact 96
2.8. Distribution des coefficients de performance 97
2.9. Distribution du ratio entre le temps de récupération et d'évaluation d'une politique 98
2.10. Évolution du temps d'évaluation pour un nombre variable de conditions temporelles 98

3.1. Graphe de provenance utilisé dans le scénario illustrant les défis et notre modèle de gestion des conflits 108
3.2. Variations de la sensibilité et de lintérêt à partager en fonction de leur composantes 122
3.3. Performances de notre solution de résolution de conflits 126

4.1. Extrait de graphe de provenance d'un forum en ligne. 146
4.2. Architecture de l'outil de vérification de la conformité 155
4.3. Interface du prototype 157
4.4. Interface du prototype 161
4.5. Brique sendMail 162

4.6. Représentation Neo4j d'un extrait de graphe de provenance généré par le générateur avec 20 utilisateurs et facteur 1 163

Liste des tableaux

1.1. Dépendances causales du modèle retenu 35
1.2. Confiance entre utilisateurs 53

2.1. Définition formelle d'une politique en langage $\mathscr{P}ol_{\mathscr{A}}$ 71
2.2. Table de vérité des contraintes de temps 82

3.1. Récapitulatif des formules pour la résolution de conflits 110
3.2. Confiance entre les utilisateurs de la Figure 1.1 112

4.1. Correspondance entre les principes du RGPD, les articles et les considérants associés 133
4.2. Traces d'exécution illustrant la limitation des finalités 139
4.3. Traces d'exécution illustrant la licéité du traitement 140
4.4. Comparaison des traces conforme et non conforme concernant la gestion des comptes utilisateurs 141
4.5. Comparaison des traces conforme et non conforme concernant la gestion des modifications des copies 141

Liste des acronymes

ABAC
Attribute-Based Access Control. 58, 67, 68

CCPA
Californian Consumer Protection Act. 18

CNIL
Commission Nationale de l'Informatique et des Libertés. 29

DAC
Discretionary Access Control. 56

DOV
Deny-override. 71, 100, 101, 105, 122, 123

DUDH
Déclaration Universelle des Droit de l'Homme. 17

FA
First-applicable. 71

HIPAA
Health Insurance Portability and Accountability Act. 132, 159

HOV
Host-override. 100, 122, 123

MAC
Mandatory Access Control. 56

NIST
National Institute of Standards and Technology. 18, 19

NSA
National Security Agency. 18

OCDE
Organisation de Coopération et de Développement Économiques. 127, 132

OOV
Owner-override. 100

OPM
Open Provenance Model ou Modèle Ouvert de Provenance. 31–36, 38, 41–43, 51, 58, 67, 159, 161

PBAC
Provenance-Based Access Control. 42

PII
Personally Identifiable Information. 19, 59, 101

POV
Permit-override. 71, 100, 105, 122

RBAC
Role-Based Access Control. 56

ReBAC
Relationship-Based Access Control. 57–59

RGPD
Règlement Général sur la Protection des Données. 18–20, 29, 127–130, 132–144, 147, 151, 152, 154, 155, 159–162

TaskBAC
Task-Based Access Control. 56

TeamBAC
Team-Based Access Control. 56

UE
Union Européenne. 18, 29, 128, 129, 132

XACML
eXtensible Access Control Markup Language. 58, 68, 71, 97, 99, 100

Introduction Générale

Les avancées technologiques, telles que l'amélioration des capacités de calcul et de stockage, ainsi que la démocratisation d'Internet, ont permis à de nombreux services et organisations de dématérialiser leurs opérations. Cela a entraîné un traitement, une collecte et un stockage massifs de données. Souvent, ces données sont à caractère personnel, c'est-à-dire des informations permettant d'identifier directement ou indirectement une personne, ce qui peut mettre en danger la vie privée des utilisateurs. Un paradoxe émerge : d'un côté, l'apparition de services visant à renforcer les relations sociales par le partage d'informations personnelles apporte des avantages, particulièrement dans des contextes d'isolement, comme lors de la pandémie de Covid-19. En effet, comme décrit dans le travail présenté par (KRASNOVA et al. 2010), partager sur les réseaux sociaux offre la commodité de garder à jour ses relations et d'en bâtir de nouvelles. Ceci peut procurer du plaisir aux utilisateurs et peut aussi améliorer son image de soi. D'un autre côté, cette évolution suscite de nouvelles préoccupations quant aux dérives et aux problèmes de sécurité que la collecte massive de données peut engendrer ainsi que la perception du risque de violation de la vie privée, avec des niveaux d'inquiétude variables selon les utilisateurs. Ce paradoxe peut être expliqué par plusieurs facteurs, comme le souligne KOKOLAKIS 2017. Par exemple, les avantages perçus de la participation aux réseaux sociaux surpassent souvent les risques observés. De plus, les individus divulguent des informations afin d'acquérir du capital social, mais aussi à cause de l'intégration croissante des réseaux sociaux dans la vie quotidienne.

Différents défis émergent, nécessitant soit l'adaptation des mécanismes de sécurisation des données, soit la création de nouveaux. Cela est dû, d'une part, à la diversité des utilisateurs de ces systèmes, certains étant des « moldus » de technologie, aux profils, expériences et parcours divers et d'autre part, à la nature même des données et au contenu des systèmes. Enfin, des législations et textes juridiques encadrent le traitement des données personnelles, imposant certaines mesures de sécurité. Les systèmes manipulant des données personnelles nécessitent donc une attention particulière en termes de sécurisation. En particulier les systèmes collaboratifs, comme les drives partagés ou les réseaux sociaux, possédant des données multi-propriétaires, nécessitent des mécanismes bien adaptés pour protéger au mieux leurs utilisateurs.

La sécurisation de ces systèmes peut passer par de la sécurité pro-active ou de la sécurité réactive. La sécurité réactive est une approche de la sécurité qui consiste à répondre aux incidents ou aux menaces après qu'ils se soient produits. La sécurité proactive, au contraire de la sécurité réactive, est l'ensemble de techniques et mécanismes

mis en œuvre pour la prévention des risques et la protection d'un système et de ses actifs. Différents types de mécanismes existent comme le chiffrement des données, la détection de vulnérabilités, les formations et sensibilisations des utilisateurs aux menaces, les sauvegardes de sécurité, la segmentation des réseaux ainsi que le contrôle des accès et les audits pour l'identification et l'atténuation de risques.

Le contrôle d'accès est un domaine assez large qui peut comprendre la gestion des identités, l'authentification des utilisateurs, ainsi que la gestion des autorisations. Nous définissons ici le contrôle d'accès comme un mécanisme permettant aux utilisateurs d'exprimer des politiques correspondant à leurs souhaits concernant l'accès à leurs données ou l'impact potentiel sur ces données. Ces politiques seront évaluées pour prendre une décision lorsque qu'une requête d'accès est émise par un utilisateur du système. Cette évaluation peut mener à des conflits si deux ou plusieurs utilisateurs ont des souhaits incompatibles.
La sécurité pro-active peut aussi se présenter sous la forme d'audits. Ces audits peuvent être variés et cibler l'identification de différents problèmes comme les audits de surveillance, les audits des systèmes et des infrastructures, les audits sur les processus ou encore les audits de conformité. Ces derniers permettent de vérifier que les pratiques d'un système ou organisation sont conformes aux exigences légales et réglementaires et ainsi permettre une garantie de sécurité vis-à-vis des utilisateurs.

Avec l'émergence de ces nouveaux systèmes et pratiques, il est nécessaire d'adapter au contexte actuel l'ensemble des mécanismes afin de répondre à ces défis spécifiques. La problématique qui en découle est la suivante : comment ajuster les mécanismes existants pour répondre aux besoins de ces nouveaux systèmes ?

Au cœur de ces défis se trouvent les données personnelles et le respect de la vie privée. Une modélisation centrée sur les données peut être une piste envisageable pour concevoir des mécanismes adaptés. La provenance des données permet de retracer les transformations, les sources et les étapes de traitement qu'une donnée a traversées. Modéliser cette provenance permet de maintenir un lien avec les données et de connaître les différents acteurs impliqués dans leur historique. La problématique étudiée dans ce mémoire est la suivante : comment la provenance des données peut-elle contribuer à proposer des modèles adaptés de contrôle d'accès et de vérification, afin de protéger la vie privée des utilisateurs ?

L'objectif de ce mémoire est de présenter différentes stratégies permettant de répondre aux défis générés par les systèmes collaboratifs et les systèmes manipulant des données personnelles à l'aide de la représentation de la provenance des données. En particulier, nous proposons un modèle de contrôle d'accès pour des systèmes centrés sur la communauté, tels que les réseaux sociaux, permettant de prendre en compte différentes politiques et une résolution de conflit plus précise et adaptée si besoin. Il est important de noter que les relations dans ce type de système imitent les relations dans le monde physique. Il est donc crucial de gérer les conflits de manière proche du comportement humain. Nous proposons également un outil de vérification de la

conformité du traitement des données dans un système afin de faciliter le travail des auditeurs.

Ce manuscrit s'articule autour de l'utilisation de la provenance pour la protection des données, en explorant deux axes de recherche de la sécurité pro-active : le contrôle d'accès et l'audit. Le Chapitre 1 introduit ainsi le contexte de ce travail, ainsi que le modèle ouvert de provenance et nos extensions nécessaires à la modélisation des systèmes étudiés. Dans les Chapitres 2 et 3 nous nous intéressons à la sécurité à priori. Dans le Chapitre 2, nous proposons un modèle de contrôle d'accès avec une granularité élevée adapté aux challenges des systèmes collaboratifs, en incluant la présentation un prototype afin de valider notre approche en termes de performance et de pertinence. Nous poursuivons dans le Chapitre 3 avec une stratégie de gestion de conflit fine, venant compléter le modèle de contrôle d'accès. Cette solution est accompagnée d'une validation de notre approche à l'aide d'un prototype permettant de simuler une version simplifiée d'un système collaboratif. Enfin, dans le Chapitre 4, nous abordons la provenance des objets du point de vue de l'audit de conformité en proposant une vérification automatique mais partielle de la conformité à un règlementation. En particulier, nous formalisons les principes principaux du RGPD et proposons le prototype outil dans le but de vérifier la conformité d'un système à certains des aspects de ce règlement.

1. Préliminaires

Sommaire

1.1. Introduction . . . 19
1.1.1. Vie privée dans le monde numérique . . . 19
1.1.2. Données personnelles . . . 21
1.1.3. Défis liés au respect de la vie privée et aux données personnelles 22
1.2. Contexte . . . 25
1.2.1. Un systèmes collaboratif basé sur la communauté : les réseaux sociaux . . . 25
1.2.1.1. Réseau de relations . . . 27
1.2.1.2. Les données utilisateurs . . . 27
1.2.1.3. Les actions . . . 28
1.2.2. Applications web . . . 30
1.2.2.1. Les données . . . 31
1.2.2.2. Les actions . . . 32
1.3. Représentation de la provenance des données . . . 33
1.3.1. Modèle Ouvert de Provenance . . . 33
1.3.1.1. Nœuds du graphe . . . 34
1.3.1.2. Arêtes et dépendances causales . . . 35
1.3.1.3. Dépendances à étapes multiples . . . 36
1.3.1.4. Extension temporelle . . . 37
1.3.2. Extension du Modèle Ouvert de Provenance avec la notion de propriétaire . . . 40
1.3.2.1. Dépendances pour représenter la propriété . . . 40
1.3.2.2. Ajout des horodotages . . . 41
1.3.3. Extension pour des systèmes dynamiques et multi-propriétaires 41
1.3.3.1. Nœuds avec des attributs . . . 41
1.3.3.2. Dépendances notAvailable et belongsTo . . . 42
1.3.3.3. Dépendances spécifiques . . . 43
1.3.4. Utilisations du Modèle Ouvert de Provenance . . . 44
1.4. Représentation de la communauté des utilisateurs . . . 45
1.4.1. Graphe des relations . . . 45
1.4.2. Définition de communauté . . . 48
1.4.3. Pondération des arêtes pour modéliser la confiance . . . 49
1.5. Capture de la provenance et des relations pour la construction du graphe 53
1.5.1. Conversion de journaux en graphe . . . 54

1.5.2. Capture de la provenance 54
1.6. Conclusion et discussion 55

1.1. Introduction

Avec l'omniprésence des interactions numériques, les utilisateurs sont de plus en plus exposés, ce qui entraîne une montée des préoccupations concernant le respect de la vie privée. À travers l'histoire, la définition de la vie privée a évolué au même rythme que la société. Avec la démocratisation d'Internet, l'informatisation des données et le partage de celles-ci à travers les réseaux sociaux, de nouvelles menaces ont vu le jour et les risques pour le viol de la vie privée ont augmenté. Les mécanismes liés à la protection de celle-ci et des données personnelles nécessitent prendre en compte les nouveaux défis relevés par ces nouveaux systèmes informatiques. Une formalisation de ces systèmes plus fine est donc nécessaire pour traiter ces problèmes. Dans ce chapitre nous abordons les éléments essentiels pour comprendre les concepts nécessaires liés à notre travail, en particulier la notion de vie privée et de son respect, ainsi que les formalismes nécessaires pour proposer des mécanismes pour les protéger. Dans cette section, nous nous centrons sur le concept de vie privée (Section 1.1.1) ainsi sur celui de donnée personnelle (Section 1.1.2). Nous aborderons ensuite les défis introduits par ces concepts dans les systèmes utilisant les données personnelles (Section 1.1.3).

1.1.1. Vie privée dans le monde numérique

Définie comme *« ce qui concerne la vie personnelle d'un individu, par opposition à sa vie publique »* (*Larousse - Vie privée* 2024) ou *« a state in which one is not observed or disturbed by other people" et "the state of being free from public attention" »* (*Oxford - Privacy* 2024) dans les dictionnaires, la notion de vie privée, ou en anglais « privacy », est une composante essentielle des droits humains. En effet, la Déclaration Universelle des Droit de l'Homme (DUDH) explicite dans l'Article 12 que *« Nul ne sera l'objet d'immixtions arbitraires dans sa vie privée, sa famille, son domicile ou sa correspondance, ni d'atteintes à son honneur et à sa réputation. Toute personne a droit à la protection de la loi contre de telles immixtions ou de telles atteintes. »*[1] . Cette notion a évolué considérablement au fil du temps, surtout avec l'avènement de l'ère numérique.

Existant déjà dans le monde romain, le terme « privatus » faisait référence à « qui ne concerne qu'un individu » et était utilisé dans le droit romain en opposition à ce qui est public. Au long des siècles, le concept de vie privée ou « privacy » a évolué vers un sens de contrôle des informations personnelles par les individus eux-mêmes. Cela a été formalisé pour la première fois en 1890 par les juristes Samuel Warren et Louis

1. En anglais « No one shall be subjected to arbitrary interference with his privacy, family, home or correspondence, nor to attacks upon his honour and reputation. Everyone has the right to the protection of the law against such interference or attack. »

Brandeis dans leur article « The Right to Privacy », publié dans la Harvard Law Review (WARREN et al. 1890), où ils décrivaient le droit de chacun à « être laissé tranquille ». Dans WESTIN 1967, l'auteur revisite la définition en mettant l'accent sur le contrôle de ce qui est divulgué, ainsi que du moment et de la manière dont cela est fait : « claim of individuals, groups, or institutions to determine for themselves when, how, and to what extent information about them is communicated to others »[2]. C'est cette même idée qui est reprise dans MUELLER 2006 où il décrit la « privacy » comme la possibilité de contrôler la distribution et l'utilisation de ses données (« *the possibility to control the distribution and use of personal data"* »). Cette prise de conscience précoce s'est intensifiée au cours des dernières décennies avec les avancées technologiques et l'omniprésence d'Internet.

L'évolution de la définition de la vie privée a été marquée par plusieurs scandales, dont les révélations d'Edward Snowden en 2013 (WILTON 2017), mettant en évidence le programme de surveillance « *Prism Project* » mis en place par la National Security Agency (NSA). Ces révélations ont mis en lumière l'ampleur des programmes de surveillance gouvernementaux, tandis que des affaires comme celles de Cambridge Analytica, avec pour but d'influencer des élections (ISAAK et al. 2018), ont démontré comment les entreprises peuvent exploiter les données personnelles à des fins commerciales ou politiques. Ces incidents ont suscité des débats mondiaux sur la protection des données et ont conduit à la création de cadres réglementaires stricts comme le Règlement Général sur la Protection des Données (RGPD) (UNION 2016) dans l'Union Européenne (UE) ou la Californian Consumer Protection Act (CCPA) (*California Consumer Privacy Act (CCPA)* 2018) en Californie ont été mis en place pour protéger les droits des individus vis à vis du respect de la vie privée.

Dans le domaine de l'informatique, la « digital privacy » ou la vie privée numérique est devenue un enjeu crucial, en particulier sur Internet où les données personnelles sont constamment collectées, analysées et souvent monétisées. De nombreuse menaces pèsent sur ce type de données et donc sur la vie privée des utilisateurs. HO 2012 les classe en trois catégories : la sécurité (incluant le vol d'identité, la duplication de profil, le phishing et le cyberharcèlement), la réputation et la crédibilité (comme les licenciements d'employés dus à leur exposition sur les réseaux sociaux), et le profilage (qui englobe le spam et la sollicitation non désirée). La vie privée numérique se réfère à la protection des informations personnelles partagées en ligne, allant des informations d'identification personnelles aux comportements de navigation. Des organisations comme le National Institute of Standards and Technology (NIST) ont formalisé la « privacy » comme étant « l'assertion des droits et des attentes des individus concernant la collecte, l'utilisation, la divulgation et la gestion de leurs données personnelles »[3].

2. En français, « la revendication des individus, des groupes ou des institutions de décider eux-mêmes quand, comment et dans quelle mesure les informations les concernant sont communiquées à d'autres ».

3. *Security and Privacy Controls for Information Systems and Organizations* 2020.

Les réseaux sociaux illustrent parfaitement l'enjeu de la vie privée numérique, se situant au cœur d'un paradoxe entre le partage d'informations et la protection des données personnelles, où la vie privée des utilisateurs peut être compromise par des pratiques de collecte de données intrusives, des partages non autorisés et des failles de sécurité.

Avec l'explosion des technologies numériques, la protection de la vie privée est devenue un défi majeur. Les réseaux sociaux comme Facebook, Twitter, TikTok [4], qui collectent des quantités massives de données personnelles, sont souvent au centre des préoccupations. Des scandales tels que l'exploitation des données par Cambridge Analytica démontrent les risques potentiels liés à une gestion inappropriée des informations privées.

La protection de la vie privée, en particulier dans le domaine numérique, nécessite une vigilance constante et une régulation rigoureuse pour garantir que les droits des individus soient respectés et protégés contre les abus potentiels des gouvernements et des entreprises. Pour parvenir à cela, des modèles formels et technologiques sont développés pour une meilleure sécurisation.

1.1.2. Données personnelles

Les données personnelles sont des informations qui permettent d'identifier directement ou indirectement une personne physique. En effet le RGPD définit les données à caractère personnel comme « *toute information se rapportant à une personne physique identifiée ou identifiable* » [RGPD, article 4]. Ces informations peuvent être le nom, l'adresse, l'adresse e-mail, le numéro de téléphone, l'adresse IP, les données de géolocalisation, les informations médicales, les données de comportement en ligne, etc. Le NIST complète cette définition en incluant dans les données personnelles (Personally Identifiable Information (PII)) les facteurs spécifiques à l'identité physique, physiologique, génétique, mentale, économique, culturelle ou sociale de cette personne (McCallister et al. 2010).

Ces définitions et règlementations ont accentué certains challenges liées aux données personnelles. Par exemple, le RGPD permet d'établir des droits robustes pour les individus en matière de contrôle et de protection de leurs données. Cependant, il peut parfois être difficile de distinguer ce qui constitue une donnée personnelle d'une donnée non personnelle (P. Schwartz et al. 2011). Par exemple, en effectuant des référencements croisés, il est possible de déduire à qui appartiennent certaines données en utilisant d'autres informations. Les auteurs de Narayanan et al. 2008 le montrent en identifiant des utilisateurs d'une base de données anonymisée, rendue publique par Netflix lors de la compétition Netflix Price, en comparant avec d'autres donnés trouvées librement sur Internet. Un autre exemple est le projet surnommé « Gaydar » de l'université de Standford (Y. Wang et al. 2018), où ils déduisent l'orientation sexuelle de personnes à travers des expressions du visage sur une photo.
En outre le stockage et le traitement à grande échelle introduisent des défis de sécuri-

4. *TikTok : Recent Data Privacy and National Security Concerns* 2023.

sation et de transparence du traitement. De plus, ces données peuvent être partagées sur des systèmes sans que les utilisateurs soient pleinement conscients de l'impact de ces partages et sans avoir un contrôle clair sur l'utilisation de leurs informations.

1.1.3. Défis liés au respect de la vie privée et aux données personnelles

Avec l'explosion des systèmes informatiques et les progrès en terme de stockage et de puissance de traitement, énormément de données sont stockées par les organisations afin de fournir des services rapides. Ces avantages introduisent des problématiques lorsque les données stockées et traitées sont des données à caractère personnel. Les défis associés à la protection de la vie privée et des données personnelles sont nombreux, mais ils peuvent être regroupés en trois grandes catégories : les défis liés au traitement des données, ceux relatifs à leur stockage, et enfin ceux concernant le partage des données.

Collecte et traitement des données Grâce à la facilité de collecte et de traitement des données, des abus peuvent survenir lorsque des organisations utilisent ou réclament des données personnelles non essentielles. Par exemple, de grandes entreprises technologiques comme Google, Facebook, et Amazon ont été critiquées pour leur collecte excessive de données personnelles, souvent sans le consentement explicite des utilisateurs, voire en dépit de leurs choix [5]. De plus, certaines pratiques, comme le programme Vitality de John Hancock Financial aux États-Unis, imposent de manière peu éthique l'utilisation de données sensibles. Dans l'exemple du programme Vitality, l'assureur exige des nouveaux adhérents qu'ils partagent leurs données médicales via des dispositifs connectés pour souscrire à une assurance décès [6]. De plus, des données peuvent être collectées et transmises à des tiers sans l'autorisation ni la notification des utilisateurs. Ce problème est particulièrement préoccupant lorsque ces données sont sensibles, comme le sont les informations médicales. Par exemple, en 2019, la révélation d'un partenariat entre Google et Ascension, l'un des plus grands systèmes de santé aux États-Unis, a suscité une attention considérable. Ce partenariat, connu sous le nom de Project Nightingale, permettait à Google de collecter et d'analyser les données médicales de millions de patients sans que ceux-ci en soient informés, soulevant des questions cruciales sur la confidentialité et la transparence dans la collecte et la gestion des données (SCHNEBLE et al. 2020). Un autre exemple tristement connu est celui de Cambridge Analytica, cité plus haut.
Pour protéger les utilisateurs et leurs données, il est nécessaire de mettre en place de règles strictes, comme le RGPD, mais aussi de développer des technologies permettant de vérifier la conformité du traitement, et répondre aux défis de transparence et de respect vis à vis des utilisateurs.

5. https://apnews.com/article/828aefab64d4411bac257a07c1af0ecb, consulté en août 2024
6. https://www.bbc.com/news/technology-45590293, consulte en août 2024.

Stockage des données Outre la collecte abusive et le manque de transparence dans le traitement des données, il est crucial d'assurer un stockage sécurisé des informations personnelles. Un stockage inadéquat peut entraîner des fuites et des vols de données, affectant ainsi la vie privée des utilisateurs des systèmes, comme en témoignent plusieurs incidents récents. Par exemple, l'attaque de Equifax en 2017[7], la violation de données du groupe hôtelier Marriott de 2014 à 2018[8], ou encore la fuite de données chez Capital One en 2019[9] sont autant d'exemples illustrant les conséquences graves de telles lacunes.

Tout comme pour le traitement des données, il est essentiel d'établir des règles claires et des mécanismes de vérification pour assurer leur application. Ces règles doivent spécifier quelles données doivent être stockées, dans quelles conditions, et comment les protéger. Par exemple, définir des politiques strictes concernant la durée de conservation des données peut considérablement réduire les risques associés aux fuites de données.

Partage des données Le partage de données personnelles peut être vu sous deux angles : la réindentification de données partagées qui avaient été anonymisées (PAWAR et al. 2018) et le partage volontaire des données personnelles sur des plateformes de types réseaux sociaux. Ici nous nous centrons sur les défis liés au partage volontaire.

Les systèmes collaboratifs centrés sur les communautés, c'est à dire les systèmes régis par les relations interpersonnelles des utilisateurs, sont au cœur des défis liés au partage des données et au respect de la vie privée. De par leur données multi-propriétaires (une donnée peut appartenir à plus d'un utilisateur) et de par leur but, les réseaux sociaux illustrent bien ces défis. Par exemple, sur une photo de groupe partagée dans un réseau social en ligne, chaque personne apparaissant dans cette photo voit sa vie privée menacée.

En particulier, avec l'essor des réseaux sociaux en ligne, des quantités massives de données personnelles sont partagées quotidiennement, suscitant de nombreuses préoccupations concernant la vie privée de leurs utilisateurs, qui est exposée à de nombreuses menaces. Certaines de ces menaces sont communes à tout type de systèmes comme le vol d'identité ou les attaque par malware (GAO et al. 2011), mais d'autres sont plus spécifiques à ce type de réseau comme l'attaque par clonage de profil ou le cyberharcèlement (JAIN et al. 2021). En effet, au-delà des risques de fuite et de divulgation des informations personnelles par les plateformes à des tiers, cette exposition peut également conduire à des abus potentiels ou à des pertes de données. Cela renforce le désir des utilisateurs de contrôler leurs informations, bien que ce contrôle ne soit pas toujours facile à obtenir, comme le souligne DAMEN et ZANNONE 2014. Leur travail met en évidence comment des choix apparemment innocents, tels que le fait de « taguer » une personne sur une photo, peuvent avoir un impact négatif

7. https://www.nytimes.com/2017/09/07/business/equifax-cyberattack.html, consulté en août 2024

8. https://www.nytimes.com/2018/11/30/business/marriott-data-breach.html?smid=url-share, consulté en août 2024

9. https://www.capitalone.com/digital/facts2019/, consulté en août 2024

sur la gestion de la vie privée.

Le partage de contenu étant devenu plus facile, rapide et direct expose également les utilisateurs à divers risques. Par exemple, un premier projet « Gaydar », développé par deux étudiants du MIT, permettait en se basant sur les relations d'un utilisateur prédire son orientation sexuelle [10]. Ces problèmes ont été accentués par le fort développement des intelligences artificielles comme le prouve le projet homonyme Y. WANG et al. 2018 cité précédemment.

La protection de la vie privée est un enjeu majeur dans les réseaux sociaux, notamment la protection contre la divulgation de données sur les utilisateurs et leurs relations sociales. Ce défi est d'autant plus compliqué que ces systèmes ont pour objectif principal de partager des informations personnelles, alors que la perception et l'importance de la confidentialité des données personnelles diffèrent selon les individus et affectent la façon dont les utilisateurs gèrent leur vie privée en ligne. Dans S. H. SCHWARTZ 2012, l'auteur base cette affirmation sur dix valeurs fondamentales et montre que cette perception dépend de plusieurs facteurs tels que la culture et les contextes sociaux, les expériences et objectifs personnels et la personnalité de chaque utilisateur.

En résumé, la gestion de la « privacy » sur les réseaux sociaux est un enjeu complexe, nécessitant des solutions robustes pour protéger les données personnelles et les relations sociales des utilisateurs. Un des mécanismes de protection est le contrôle d'accès, en particulier le contrôle d'accès collaboratif. Cependant, comme mis en avant dans PACI et al. 2018, il est très compliqué d'adapter le contrôle d'accès aux défis des systèmes collaboratifs basés sur la communauté. C'est dans ce contexte là que s'inscrit le travail présenté en Chapitre 2.

Ces défis sont abordés dans les Chapitres 2 et 3, en particulier ceux concernant les systèmes qui partagent des informations personnelles. Le Chapitre 4 traitera des défis liés au traitement et au stockage des données, en mettant un accent particulier sur la conformité de ces pratiques avec la législation en vigueur.

Dans ce mémoire, nous proposons des solutions aux défis évoqués précédemment. Nous concentrons notre approche sur deux types de systèmes : les réseaux sociaux et les sites web. Au cœur des ces défis se trouvent les données personnelles et les relations entre utilisateurs. Pour représenter ces systèmes de manière formelle et intuitive, nous utilisons un formalisme fondé à la fois sur la provenance des données et sur les relations interpersonnelles. Les informations nécessaires peuvent être recueillies à partir des événements du système ou des journaux d'activité. En effet, les traces d'exécution ou journaux permettent de suivre les processus ainsi que les données qu'ils manipulent, et d'enregistrer les actions des utilisateurs lors de la création de nouvelles relations. Dans la suite de ce chapitre, nous présentons donc en détail le contexte de notre travail, en nous focalisant sur les systèmes manipulant des don-

10. https://cacmb4.acm.org/news/42191-project-gaydar-computes-orientation/fulltext, consulté en août 2024

nées personnelles, comme les sites web et les systèmes collaboratifs centrés sur la communauté, en particulier les réseaux sociaux (Section 1.2). Les Sections 1.3 et 1.4 sont consacrées à la formalisation de la représentation de ces systèmes, en s'appuyant sur nos extensions de la modélisation de la provenance introduites dans les articles BERTOLISSI, MARTINEZ ANTON et ZANNONE 2023 ; MARTINEZ ANTON et al. 2024, ainsi que dans une version journal en cours de finalisation. Le chapitre présente une démonstration de la faisabilité de notre formalisation (Section 1.5) , et conclut par une brève discussion de notre modélisation en Section 1.6.

1.2. Contexte

Les systèmes informatiques manipulent un volume croissant de données en raison de l'explosion des sources d'information, de l'augmentation des capacités de stockage et de traitement, ainsi que de la nécessité d'analyser ces données pour prendre des décisions éclairées dans divers domaines. Cette tendance est renforcée par la diversification des systèmes, tels que les boutiques en ligne, l'informatisation des démarches administratives, la gestion des données médicales, et les plateformes de communication qui génèrent et manipulent des quantités toujours plus importantes d'informations. De plus, ces données sont souvent des données personnelles, rendant leur protection une priorité. Ici, nous nous centrons sur deux types de systèmes : les systèmes collaboratifs basés sur la communauté, en particulier les réseaux sociaux et les sites web plus classiques, comme les boutiques en lignes.

1.2.1. Un systèmes collaboratif basé sur la communauté : les réseaux sociaux

Un type de système a particulièrement prospéré ces dernières décennies : les systèmes collaboratifs, notamment ceux centrés sur la communauté. Ces systèmes, également appelés « groupware », sont des applications logicielles conçues pour aider un groupe de personnes à accomplir une tâche commune, en les mettant en relation. Ces systèmes facilitent la communication, la coordination et la collaboration entre les membres du groupe, souvent en partageant des outils et des plateformes qui simplifient la gestion de projets, la co-création de documents et la prise de décisions, même lorsque les utilisateurs sont géographiquement dispersés comme expliqué dans BAFOUTSOU et al. 2002. Ces systèmes sont dynamiques, car ils sont conçus pour être évolutifs, permettant de gérer des tâches susceptibles de changer au fil du temps. Ils facilitent l'ajout ou la suppression de participants, la réallocation des tâches, ainsi que l'ajustement des processus en temps réel pour répondre aux besoins de l'équipe ou du projet. De plus, ces systèmes sont également multipartites ou multipropriétaires, nécessitant de partager le contrôle entre plusieurs utilisateurs des tâches et des ressources communes. Il existe deux principaux types de systèmes collaboratifs : les systèmes centrés sur les tâches et les systèmes centrés sur la communauté (en anglais « community-centered collaborative systems »). Les premiers se concentrent

sur l'accomplissement de tâches spécifiques, dont ses utilisateurs ne sont en relation que pour la durée de la tâche, tandis que les seconds mettent en relation des communautés d'utilisateurs pour partager des informations et des ressources d'intérêt commun.

Au cours des dernières décennies, ces derniers ont gagné en popularité, permettant de mettre en relation des utilisateurs pour créer et partager des ressources de manière collaborative, comme par exemple les réseaux sociaux. Ils permettent à des communautés d'utilisateurs en ligne de partager des informations et des ressources (BARAB et al. 2004). Régulés par des relations interpersonnelles entre utilisateurs, ces derniers coopèrent pour créer, gérer et protéger les ressources au sein de la communauté. Les relations établies au sein de ces systèmes et formant les communautés calquent les relations interpersonnelles de la vie réelle des utilisateurs. Cependant, un problème majeur se pose : la collaboration implique nécessairement le partage de données, qu'elles soient personnelles ou non. La question de la gestion de la vie privée des utilisateurs et du contrôle des éléments créés de manière collaborative devient alors cruciale. Les utilisateurs doivent savoir comment leurs données sont protégées et qui a le pouvoir de contrôler et de modifier ces éléments partagés. De plus, ces systèmes peuvent réunir des utilisateurs de divers horizons et expertises. Tout comme les systèmes d'information traditionnels et les systèmes collaboratifs centrés sur les tâches, l'interaction entre des utilisateurs non spécialistes exige des moyens simples et transparents pour gérer les ressources partagées. En plus de contenir les problématiques des systèmes multipartites et dynamiques, les sytèmes centrés sur la communauté introduisent également de nouveaux défis liés aux relations interpersonnelles.

Un exemple très répandu et utilisé de système collaboratif basé sur la communauté sont les réseaux sociaux. En effet, ils sont devenus très populaires et comptent des millions d'utilisateurs partageant souvent des informations personnelles.

Un réseau social est une plateforme en ligne où les différents utilisateurs peuvent interagir les uns avec les autres en créant de nouvelles relations et en partageant du contenu. Comme défini dans BOYD et al. 2013, les réseaux sociaux permettent aux utilisateurs de créer un profil, d'entrer en contact avec d'autres utilisateurs pour partager des informations, des idées, des messages, des photos, des vidéos et d'autres contenus. Les réseaux sociaux facilitent les interactions et la communication entre utilisateurs, souvent en permettant la création de groupes d'intérêt, de pages publiques et de discussions en temps réel. Ils jouent un rôle crucial dans la formation de communautés en ligne, en permettant aux utilisateurs de maintenir et de développer des relations personnelles et professionnelles. Différents réseaux sociaux existent avec des objectifs variés et spécifiques. Dans le rapport BEYE et al. 2010, les auteurs les classent selon leur objectif (leur type de connexion entre les utilisateurs) ainsi que selon le type de contenu. Un réseau social peut être un site de rencontres, servir à créer et maintenir un réseau professionnel, permettre de garder contact avec les relations existantes dans la vie réelle ou encore socialiser et rencontrer de nouvelles personnes. Ces deux dernières catégories sont les plus répandues.

En ce qui concerne le type de contenu, les auteurs les divisent en six catégories : partage, recommandations, divertissement, conseils, loisirs et actualités.

Dans ce travail nous considérerons les réseaux sociaux qui favorisent le partage de l'information (Facebook, Instagram, ...). Nous décrivons ici les principales fonctionnalités. Les composants principaux de ce type de réseau social sont un ensemble d'utilisateurs, un réseau de relations et des données utilisateurs pouvant être de différents types. Différentes actions peuvent être réalisées sur les données partagées par les utilisateurs.

1.2.1.1. Réseau de relations

Le réseau de relations représente les connexions sociales entre les utilisateurs. Il peut naturellement être représenté à l'aide d'une structure de graphe où chaque nœud du graphe désigne un utilisateur, et chaque arête entre deux nœuds représente une interaction sociale entre deux utilisateurs. L'étiquette associée à chaque arête indique le type d'interaction, qui peut varier en fonction du réseau social spécifique et de ses objectifs. Dans un réseau de relations semblable à Facebook, les étiquettes incluent, par exemple, *ami* et *ami d'ami*. Les relations peuvent être asymétriques, par exemple si un utilisateur suit quelqu'un (« *follower* »), mais peut ne pas être suivi en retour.

Les réseaux sociaux étant des systèmes dynamiques, les relations entre utilisateurs peuvent évoluer avec le temps, car les utilisateurs peuvent nouer de nouvelles relations ou décider de quitter des groupes et mettre fin à des amitiés existantes. On peut choisir de représenter uniquement les relations actuelles, celles qui existent encore, ou bien de garder une trace de toutes les relations qui ont existé dans le passé. Pour ce faire, les étiquettes des arêtes incluent l'intervalle de temps pendant lequel la relation existe, afin de représenter la nature dynamique des relations.

La Figure 1.1 montre un réseau de relations simple comprenant sept utilisateurs (Alice, Bob, Christopher, Daphne, Fiona, Finn et Edward) et leurs relations interpersonnelles. Les arêtes (symétriques) étiquetées *areRelated* sont annotées avec le type de relation (par exemple *ami*) et une période de temps. Les horodatages sont utilisés pour représenter la date de début et de fin d'une relation. Par exemple, *Alice* et *Fiona* sont devenues amies dans le réseau social le 12 novembre 2023. Comme l'amitié est en cours, le symbole ∞ est utilisé comme second horodatage pour indiquer que la relation n'est pas encore terminée (cf. Section1.4).

1.2.1.2. Les données utilisateurs

Dans les réseaux sociaux en ligne, les utilisateurs se voient généralement attribuer un espace web (également appelé *profil utilisateur*) où ils peuvent stocker et gérer des données personnelles telles que les informations de profil et le contenu utilisateur. La Figure 1.2 présente les différents types de contenus utilisateur (désignés ci-après par le terme *objets*) qui peuvent être créés et modifiés dans un profil utilisateur ainsi que leurs relations.

Un *post* est une entrée dans le profil et peut inclure du contenu sous forme de texte,

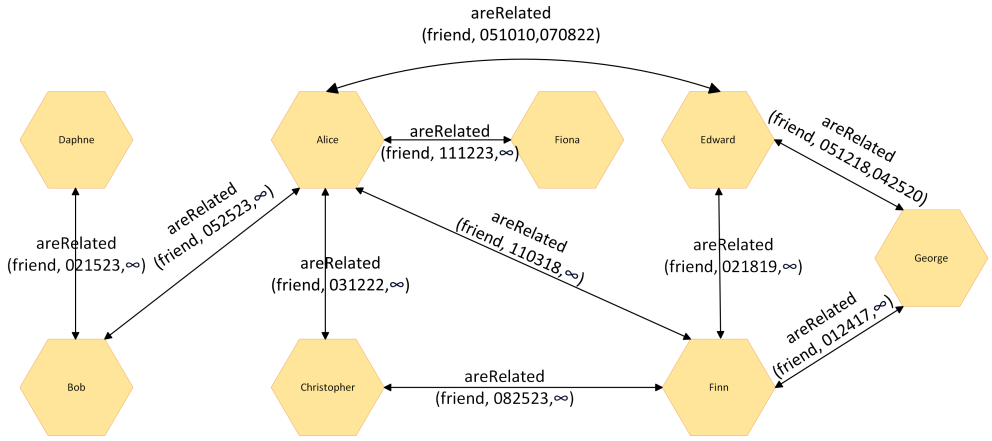


FIGURE 1.1. – Réseau de relations

photos, infographies ou vidéos. Les utilisateurs peuvent également télécharger des *photos* dans leurs *albums*, qui sont des dossiers dans le profil utilisateur dédiés au stockage et à la gestion des photos [11].

Les posts, les photos et les albums peuvent avoir zéro, un ou plusieurs *commentaires*, c'est-à-dire des notes commentant un objet cible. Les commentaires peuvent également être associés à d'autres commentaires, par exemple en répondant à un commentaire existant. Une photo peut également être associée à des *tags*, qui sont des étiquettes liant la photo au profil de l'utilisateur tagué.

Les utilisateurs peuvent créer un nouveau contenu (sous forme d'un post) sur leur profil ou le profil d'un autre utilisateur et des photos dans leurs albums, ou ajouter une nouvelle donnée (par exemple, un *commentaire*) à un contenu existant. Par la suite, nous désignons l'utilisateur qui crée l'objet comme le *propriétaire* de l'objet et l'utilisateur sur le profil duquel l'objet est publié comme l'*hôte* de l'objet. Il est à noter que le propriétaire de l'objet et l'hôte sont les mêmes utilisateurs si un utilisateur publie un objet sur son propre profil. Nous appelons ces utilisateurs les contrôleurs impliqués dans l'existence de l'objet.

1.2.1.3. Les actions

Les actions possibles dépendent du réseau social observé, mais restent les mêmes pour la plupart : ajouter du contenu (cette action peut avoir différents noms : poster, charger, ...), supprimer, modifier, partager, commenter ou encore tagger.

Les actions que les utilisateurs peuvent effectuer sur un objet dépendent du type d'objet et de leurs permissions sur cet objet. Sur Facebook, l'hôte des données possède

11. Notez que les photos téléchargées dans un album et les photos publiées sur le profil utilisateur en tant que posts sont considérées comme des types d'objets différents. Dans le texte, nous utilisons le terme *photo* pour désigner une photo téléchargée dans un album et nous nous référons à une photo publiée sur le profil utilisateur en tant que *post* dans le Chapitre 2. Par abus de langage et pour simplifier les notations, dans le Chapitre 3 nous employons le terme *photo* dans les deux cas.

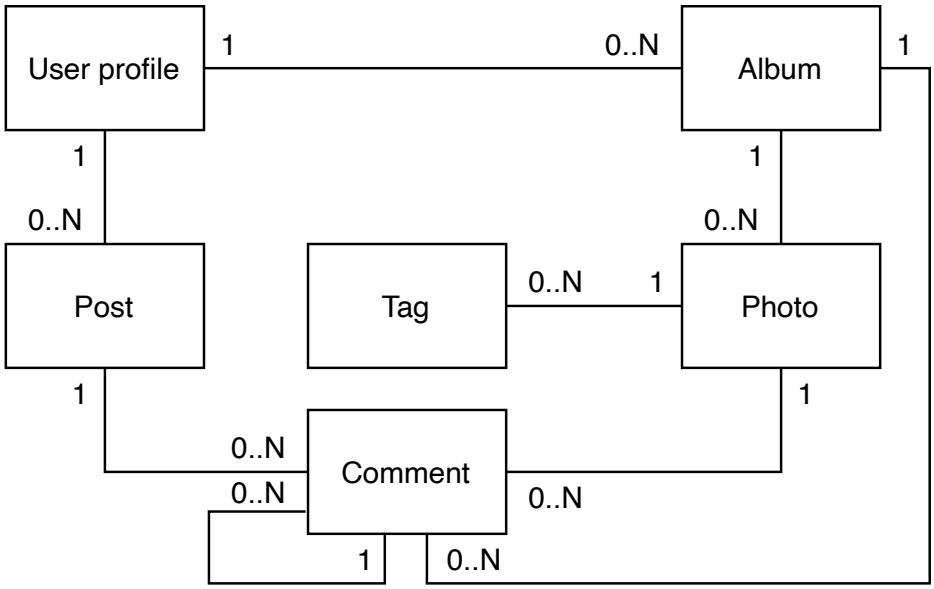


FIGURE 1.2. – Contenu utilisateur dans un réseau social

toutes les permissions (à l'exception de `modifier`) sur son profil et les objets qui s'y trouvent. Seul le propriétaire de l'objet peut `modifier` l'objet [12]. En se basant encore sur un réseau social proche de Facebook, les utilisateurs peuvent `poster` du nouveau contenu sur le profil d'un autre utilisateur uniquement s'ils peuvent `voir` ce profil.

Pour les objets dans le profil d'autres utilisateurs, les permissions sont déterminées par la relation de l'utilisateur avec l'objet (par exemple, propriétaire) ainsi que par la visibilité de l'objet (DAMEN et ZANNONE 2014), déterminée par les paramètres de confidentialité spécifiés par l'hôte des données (ou par le propriétaire si l'objet est un commentaire). En particulier, l'hôte des données peut spécifier dans ses paramètres les groupes (définis en termes d'interactions sociales comme modélisées dans le réseau de relations) qui peuvent `voir` les posts et les photos dans son profil. Un utilisateur peut `voir` le contenu dans le profil d'un autre utilisateur uniquement s'il appartient à un groupe auquel le propriétaire du profil a accordé la permission de `voir` le contenu. Par la suite, nous considérons les groupes pris en charge sur Facebook (par exemple, amis, amis des amis, public). De plus, ils peuvent `commenter` (c'est-à-dire ajouter un commentaire) des posts, des albums et des photos uniquement s'ils ont la permission de `voir` l'objet.

Un utilisateur peut également `taguer` (c'est-à-dire ajouter un tag à) une photo uniquement s'il a la permission de `poster` sur le profil ou sur un post contenant une photo.

Le tag d'une photo a deux effets : un post contenant la photo est automatiquement ajouté au profil de l'utilisateur tagué (désigné également comme *cible*), et la visibilité de la photo est élargie (DAMEN et ZANNONE 2014). Plus précisément, la visibilité de la photo est modifiée pour inclure le même groupe que celui spécifié par l'hôte des

12. Seuls les photos, les posts et les commentaires peuvent être modifiés sur Facebook.

données dans ses paramètres, mais maintenant pour l'utilisateur tagué. Par exemple, si l'hôte des données spécifie que ses amis peuvent `voir` la photo, la visibilité de la photo taguée inclura les amis de l'hôte des données *et* les amis de l'utilisateur tagué.

Les utilisateurs peuvent `partager` des posts ou des photos qui existent déjà sur les réseaux sociaux avec d'autres utilisateurs [13]. Cela aboutit à un nouveau post contenant un lien vers le post/photo original. Il est à noter que si un post partagé précédemment est partagé à nouveau par un autre utilisateur, le nouveau partage est une référence au post original (et non à la première copie). Par exemple, si Bob aime un post sur le profil d'Alice et décide de le `partager`, une copie du post faisant référence au post original d'Alice est ajoutée au profil de Bob. Si un troisième utilisateur, Christopher, voit le post sur le profil de Bob et décide de le (re)`partager`, le résultat sera un post sur le profil de Christopher contenant une référence au post original d'Alice.

Un objet peut être `supprimé` par son *propriétaire*, par son *hôte* et, si l'objet est un tag, par la *cible* du tag. Lorsqu'un *tag* est supprimé, la photo n'est plus disponible sur le profil de l'utilisateur tagué, et la visibilité originale est restaurée. En supprimant un objet, tous les objets associés à l'objet supprimé sont également supprimés, sauf si les objets associés ont été créés via une action de `partage`. Par exemple, en supprimant une photo ou un post, tous les commentaires et tags associés sont supprimés, ou en supprimant un commentaire, toutes les réponses au commentaire sont également supprimées.

On peut ainsi étendre la définition de *contrôleurs* d'un objet aux utilisateurs donnant les permissions d'accès pour cet objet. Par exemple, sur Facebook l'hôte est le seul contrôleur de la plupart des objets pour l'action `view` mais seul le propriétaire peut `modifier` l'objet. Mais on peut imaginer que quelqu'un taggé sur un objet pourrait aussi donner ses préférences et donc devenir un *contrôleur*.

1.2.2. Applications web

Dans cette sous-section nous nous intéressons aux sites web plus génériques, ne prenant pas compte uniquement les systèmes collaboratifs. Un site web est un ensemble de pages web accessibles sur Internet via un navigateur. Il peut contenir divers types de contenu, tels que du texte, des images, des vidéos et des fonctionnalités interactives. Chaque site web a des objectifs différents. On peut observer grands types de sites : des sites de contenu (site d'actualité, site éducatif), des sites d'entreprise ou institutionnels, des sites de divertissement comme les plateformes de vidéos ou de jeux en ligne, des sites de e-commerce, des sites de service comme les messageries ou les sites de réservation de voyages, ménage, etc. Chaque type de site web contiendra des données et proposera des actions qui lui sont spécifiques, cependant certains éléments sont présents dans la majorité des sites, peu importe leur objectif.

13. Notez que sur Facebook, le `partage` fonctionne actuellement uniquement sur les posts avec visibilité publique.

1.2.2.1. Les données

Une application ou un site web contient et utilise différentes catégories de données. Les *données du système* représentent toutes les informations de base nécessaires à son fonctionnement ou spécifiques à l'activité du système. Par exemple, un site de commerce électronique contiendra des données sur les produits qu'il vend, sur ses stocks, etc.

Dès qu'un utilisateur interagit avec un site web, que ce soit par une simple navigation ou en créant un compte, des *données personnelles* peuvent être stockées par le système. Comme défini en Section 1.1.2, les données personnelles sont les données qui peuvent identifier directement ou indirectement un utilisateur. Ces données peuvent être *fournies* par l'utilisateur, comme son numéro de carte bancaire lors d'un achat, ou bien *collectées* directement par le système, comme l'adresse IP de son ordinateur.

À partir des données du système et des données personnelles, le système peut générer de nouvelles informations, qu'on appelle *données générées*. Par exemple, lorsqu'un utilisateur effectue un achat sur un site de commerce électronique, un bon de commande est généré. Ce document contiendra le nom des articles achetés (données système) ainsi que l'adresse d'expédition et le nom de l'utilisateur ayant passé la commande (données personnelles). Un type de données générées correspond aux *cookies*.

Les *cookies* sont définis par la Commission Nationale de l'Informatique et des Libertés (CNIL) comme étant de petits fichiers stockés sur le terminal d'un utilisateur, associés au site web où ils ont été générés. Certains cookies, appelés *cookies non personnels*, stockent des informations génériques telles que les préférences de langue ou les paramètres de navigation, sans révéler d'informations spécifiques sur l'utilisateur. D'autres, appelés *cookies personnels*, peuvent stocker des informations liées à un utilisateur identifié ou identifiable, tels que des identifiants de connexion, des historiques de navigation ou des informations de suivi. Ces cookies peuvent être utilisés pour suivre les activités de l'utilisateur à travers différents sites et sessions, et ont de multiples usages : mémoriser l'identifiant de l'utilisateur pour simplifier ses connexions ultérieures sur un même appareil, stocker la langue choisie sur un site web, ou encore à des fins statistiques en analysant les choix de navigation, etc.
Parmi ces cookies, qu'ils soient personnels ou non, on distingue deux catégories : les cookies internes et les cookies tiers. Les cookies internes sont utilisés uniquement par le site consulté par l'utilisateur. Les cookies tiers, en revanche, sont partagés avec d'autres sites web tiers, permettant ainsi à ces derniers de savoir quelles pages ont été visitées par l'utilisateur, souvent à des fins publicitaires.

L'utilisation de données personnelles est régie dans l'UE par le RGPD stipulant que le consentement doit être donné pour des finalités précises. Les sites web sont donc tenus de demander aux utilisateurs comment les données collectées peuvent être utilisées, en particulier quels types de cookies peuvent être générés. Cet ensemble de choix est représenté par un *modèle de politique de confidentialité*, qui sera instancié pour chaque utilisateur en fonction de ses choix.

La Figure 1.3 représente les différentes données présentes dans un site web et les liens qu'elles peuvent avoir.

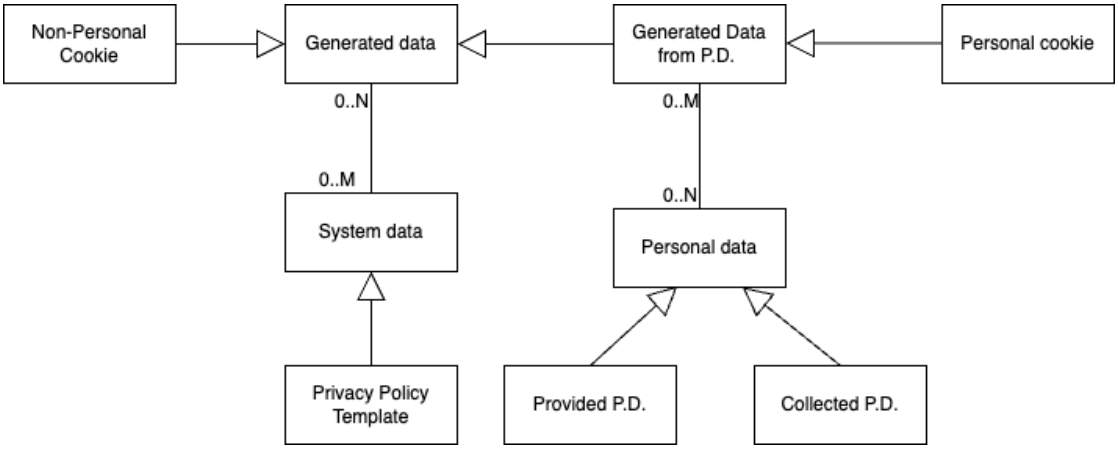


FIGURE 1.3. – Contenu d'un site web

1.2.2.2. Les actions

Les actions réalisables sur un site ou une application web dépendent de la nature du site. Cependant, il existe des actions communes à la plupart des sites web.

Tout d'abord, un utilisateur peut *naviguer* dans les différentes pages et sections d'un site web. La fonction de *recherche* est également souvent disponible pour permettre de trouver des informations spécifiques plus rapidement. Ces deux actions sont fréquemment utilisées pour générer des cookies.

De plus en plus, quel que soit l'objectif du site, il est possible de *créer un compte* en tant qu'utilisateur. Cela permet de retrouver plus facilement ses préférences, de gérer ses achats ou de s'abonner à un service. Les utilisateurs peuvent également *interagir* de manière plus active. Par exemple, ils peuvent soumettre un formulaire pour contacter le service client, laisser des avis sur des produits ou des articles, ou encore *consulter* des produits, des descriptions de services ou lire du contenu.

Dans le cas des sites de commerce électronique ou de services, l'utilisateur peut également *acheter* directement sur le site. Enfin, l'action de *partager* du contenu sur une autre plateforme, pour diffuser des informations ou donner un avis, est de plus en plus répandue. Cette action est particulière car son effet n'est pas directement lié au site web consulté.

On retrouve au cœur des problématiques les données personnelles et les actions qui sont réalisées sur ces données. Pour pouvoir proposer des mécanismes de protection des données dans ces systèmes, il est nécessaire de formaliser la représentation du système. Représenter les dépendances entre les données et les actions est une solution intuitive permettant d'avoir une vision formelle. Nous proposons ainsi d'utiliser la pro-

venance des données. Au cœur des systèmes basés sur des communautés, on retrouve aussi les différentes relations entre les utilisateurs, nécessitant une modélisation aussi.

1.3. Représentation de la provenance des données

La provenance est définie comme « *endroit d'où quelque chose vient ou provient* » (LAROUSSE n.d.) ou « *The history of the ownership of a work of art or an antique, used as a guide to authenticity or quality; a documented record of this.* » (OXFORD UNIVERSITY PRESS n.d.). Le terme « provenance » désigne donc l'origine ou la source d'un objet, d'une œuvre ou d'une information. Il fait référence au lieu, à la personne, au processus ou aux circonstances à partir desquels quelque chose a été acquis ou produit. La provenance permet la traçabilité des données, en garantissant leur intégrité, leur qualité et leur fiabilité grâce à un historique clair de leur manipulation et de leur gestion.

Ce concept est largement utilisé dans le monde numérique, que ce soit pour la gestion des flux de données, des bibliothèques numériques ou la documentation de la vie des objets et données numériques. La provenance est employée à plusieurs fins, telles que la gestion des accès aux données, la réalisation d'audits de conformité ou l'analyse des données.

Dans ce contexte, nous avons choisi de modéliser le contenu des systèmes collaboratifs centrés sur la communauté en utilisant la provenance de ce contenu, en particulier la représentation proposée par l'Open Provenance Model ou Modèle Ouvert de Provenance (OPM) (MOREAU et al. 2011). En raison de sa normalisation et de sa formalisation intuitive, ce modèle constitue une base solide pour relever les défis posés par ces systèmes complexes, tels que la protection de la vie privée et la gestion des données personnelles.

Dans cette section, nous présentons d'abord l'OPM, ainsi que les dépendances associées et la notion de propriétaire pour la provenance introduite dans BERTOLISSI, HARTOG et al. 2019. Nous introduisons aussi notre extension pour modéliser la provenance dans les systèmes collaboratifs. Afin de justifier l'utilisation de ce modèle, nous explorons les différents champs d'application des graphes de provenance.

1.3.1. Modèle Ouvert de Provenance

Le Modèle Ouvert de Provenance (OPM) (MOREAU et al. 2011) est un modèle permettant de représenter formellement la provenance des données d'un système. Ce modèle a eté conçu pour satisfaire plusieurs exigences essentielles : faciliter l'échange d'informations de provenance entre différents systèmes, soutenir la création et le partage d'outils pour les développeurs basés sur ce modèle, et offrir une définition précise et indépendante de la technologie. De plus, il permet une représentation numérique de la provenance pour tout type d'objet (issu de systèmes informatiques ou non). Il permet également la coexistence de plusieurs niveaux de description et

définit un ensemble de règles fondamentales pour déterminer les déductions logiques valides à partir des représentations de provenance.

Développé dans le but de standardiser la représentation de la provenance dans le cadre de différents défis, le OPM représente les actions passées au sein d'un système sous la forme d'un graphe orienté et étiqueté. Ce graphe, appelé *graphe de provenance*, contient trois types de nœuds et des arêtes représentant les dépendances causales entre les différents nœuds. Il exprime comment les objets évoluent dans le système en utilisant des relations de causalité. Ce graphe est un triplet $G = (\mathcal{N}, \mathcal{E}, L)$, où $\mathcal{N}$ est l'ensemble de nœuds, $\mathcal{E}$ est l'ensemble d'arêtes orientées entre les nœuds, et L est l'ensemble d'étiquettes. Étant donné que la provenance fournit une sorte d'historique du système, le graphe de provenance ne peut être modifié qu'en ajoutant de nouveaux nœuds et dépendances.

1.3.1.1. Nœuds du graphe

Les nœuds du graphe représentent les entités du système, c'est à dire les données ou objets, les processus qui manipulent ces éléments et les entités contrôlant les processus. Les nœuds se divisent donc naturellement en trois sous-ensembles disjoints : les artefacts, les processus et les agents.

Artefacts Les artefacts représentent des données ou des objets immuables. En effet, comme mentionné précédemment, le graphe ne peut être modifié qu'en ajoutant de nouveaux éléments. Les objets modifiés sont immédiatement représentés par un nouvel artefact. Ces éléments peuvent être des objets physiques ou des représentations numériques. Dans le graphe de provenance, les artefacts sont illustrés par des cercles verts.

Processus Les processus représentent les actions ou les groupes d'actions qui sont réalisées sur les artefacts. Ces actions peuvent être menées sur les artefacts ou être déclenchées par d'autres processus. Des nouveaux artefacts peuvent être crées. Par exemple, un processus peut créer un *rapport* (artefact créé) en traitant les *données d'un capteur* (artefact utilisé). Les processus sont représentés par des rectangles bleus dans le graphe de provenance.

Agents Un agent est une entité dont le rôle et la fonction sont spécifiques au contexte. Il agit comme un catalyseur d'un processus en contrôlant, affectant, modifiant ou facilitant son exécution. Par exemple, le processus générant le rapport a été exécuté par un analyste. Les agents sont représentés par des hexagones jaunes dans le graphe de provenance.

On dénote les ensembles d'artefacts, processus et agents $\mathcal{O}$, $\mathcal{P}$ et $\mathcal{A}$ respectivement. On obtient ainsi l'ensemble des nœuds $\mathcal{N} = \mathcal{O} \cup \mathcal{P} \cup \mathcal{A}$ avec $\mathcal{O}, \mathcal{P}, \mathcal{A}$ deux à deux disjoints.

Dépendances Causales du Modèle Ouvert de Provenance (MOREAU et al. 2011)
$\text{used}(p:\mathscr{P}, d:\mathscr{O}, r:\mathscr{R}, ts_u:\mathscr{T})$
$\text{wasGeneratedBy}(d:\mathscr{O}, p:\mathscr{P}, r:\mathscr{R}, ts_g:\mathscr{T})$
$\text{wasControlledBy}(p:\mathscr{P}, s:\mathscr{A}, r:\mathscr{R}, ts_b \in \mathscr{T}, ts_e:\mathscr{T})$
$\text{wasTriggeredBy}(p_1:\mathscr{P}, p_2:\mathscr{P}, ts_u:\mathscr{T}) = \exists d \in \mathscr{O}, r_1, r_2 \in \mathscr{R}, ts_g \in \mathscr{T} : \text{used}(p_1, d, r_1, ts_u) \wedge \text{wasGeneratedBy}(d, p_2, r_2, ts_g)$
$\text{wasDerivedFrom}(d_1:\mathscr{O}, d_2:\mathscr{O}, r:\mathscr{R}, ts_g:\mathscr{T}) = \exists p \in \mathscr{P}, r_i \in \mathscr{R}, ts_u \in \mathscr{T} : \text{wasGeneratedBy}(d_1, p, r_i, ts_g) \wedge \text{used}(p, d_2, r, ts_u)$
Dépendances à Étapes Multiples
$\text{used}^+(p:\mathscr{P}, d:\mathscr{O}, ts_b:\mathscr{T}, ts_e:\mathscr{T}) = \exists p_i \in \mathscr{P}, r \in \mathscr{R}, ts_{b'} \in \mathscr{T} : \text{wasTriggeredBy}^+(p, p_i, ts_{b'}, ts_e) \wedge \text{used}(p_i, d, r, ts_b)$
$\text{wasGeneratedBy}^+(d:\mathscr{O}, p:\mathscr{P}, ts_b:\mathscr{T}, ts_e:\mathscr{T}) = \exists p_i \in \mathscr{P}, r \in \mathscr{R}, ts'_e \in \mathscr{T} : \text{wasGeneratedBy}(d, p_i, r, ts_e) \wedge \text{wasTriggeredBy}^+(p_i, p, ts_b, ts'_e)$
$\text{wasTriggeredBy}^+(p_1:\mathscr{P}, p_2:\mathscr{P}, ts_b:\mathscr{T}, ts_e:\mathscr{T}) = (\text{wasTriggeredBy}(p_1, p_2, ts_e) \wedge (ts_b = ts_e)) \vee (\exists p_i \in \mathscr{P}, ts_{e'} \in \mathscr{T} : \text{wasTriggeredBy}(p_1, p_i, ts_e) \wedge \text{wasTriggeredBy}^+(p_i, p_2, ts_b, ts'_e))$
$\text{wasDerivedFrom}^+(d_1:\mathscr{O}, d_2:\mathscr{O}, ts_b:\mathscr{T}, ts_e:\mathscr{T}) = (\text{wasDerivedFrom}(d_1, d_2, r, ts_e) \wedge (ts_b = ts_e)) \vee (\exists d_i \in \mathscr{O}, r_i \in \mathscr{R}, ts_{e'} \in \mathscr{T} : \text{wasDerivedFrom}(d_1, d_i, r_i, ts_e) \wedge \text{wasDerivedFrom}^+(d_i, d_2, ts_b, ts_{e'}))$
Dépendances pour Modéliser la Propriété (BERTOLISSI, HARTOG et al. 2019)
$\text{owns}(s:\mathscr{A}, d:\mathscr{O}, ts_g:\mathscr{T}) = \exists p \in \mathscr{P}, r \in \mathscr{R}, ts_b, ts_e \in \mathscr{T} : \text{wasGeneratedBy}(d, p, r, ts_g) \wedge \text{wasControlledBy}(p, s, \texttt{owner}, ts_b, ts_e)$
$\text{contributedTo}(s:\mathscr{A}, d:\mathscr{O}, r:\mathscr{R}, ts_g:\mathscr{T}) = \exists p \in \mathscr{P}, ts_b, ts_e \in \mathscr{T} : \text{wasGeneratedBy}^+(d, p, ts_g, \wedge) \text{wasControlledBy}(p, s, r, ts_b, ts_e)$
Dépendances pour les systèmes dynamiques et multi-propriétaires
$\text{belongsTo}(d_1:\mathscr{O}, d_2:\mathscr{O}, ts_g:\mathscr{T}) = \exists p \in \mathscr{P}, r_i, r_j \in \mathscr{R}, ts_u \in \mathscr{T} : \text{wasGeneratedBy}(d_1, p, r_i, ts_g) \wedge \text{used}(p, d_2, r_j, ts_u) \wedge \text{action}(p) \in \mathscr{V}_{action_{inclusion}}$
$\text{notAvailable}(d:\mathscr{O}, t:\mathscr{T}) = \exists p \in \mathscr{P}, r \in \mathscr{R} : \text{used}(p, d, r, t) \wedge \text{action}(p) = \texttt{delete}$
Dépendances des Relations Inter-agents
$\text{isRelated}(s_1:\mathscr{A}, s_2:\mathscr{A}, r:\mathscr{R}, ts_b:\mathscr{T}, ts_e:\mathscr{T})$
$\text{areRelated}(s_1:\mathscr{A}, s_2:\mathscr{A}, r:\mathscr{R}, ts_b:\mathscr{T}, ts_e:\mathscr{T}) = \exists ts_{b1}, ts_{e1}, ts_{b2}, ts_{e2}$ s.t. $\text{isRelated}(s_1, s_2, r, ts_{b1}, ts_{e1}) \wedge \text{isRelated}(s_2, s_1, r, ts_{b2}, ts_{e2}) \wedge [ts_b, ts_e] = [ts_{e1}, ts_{e2}] \cap [ts_{b2}, ts_{e2}]$
$\text{isRelated}^k(s_2:\mathscr{A}, s_1:\mathscr{A}, r:\mathscr{R}, ts_b:\mathscr{T}, ts_e:\mathscr{T}) = \exists u_1, ts_{b1}, ts_{e1}, \ldots, u_{k+1}, ts_{bk+1}, ts_{ek+1}$ s.t. $\forall i \in \{1,\ldots k+1\}\ \text{isRelated}(u_i, u_{i+1}, r, ts_{bi}, ts_{ei}) \wedge u_1 = s_1 \wedge u_{k+1} = s_2 \wedge [ts_b, ts_e] = \bigcap_{i=1}^{k+1}[ts_{bi}, ts_{ei}]$
$\text{areRelated}^k(s_2:\mathscr{A}, s_1:\mathscr{A}, r:\mathscr{R}, ts_b:\mathscr{T}, ts_e:\mathscr{T}) = \exists u_1, ts_{b1}, ts_{e1}, \ldots, u_{k+1}, ts_{bk+1}, ts_{ek+1}$ s.t. $\forall i \in \{1,\ldots k+1\}\ \text{areRelated}(u_i, u_{i+1}, r, ts_{bi}, ts_{ei}) \wedge u_1 = s_1 \wedge u_{k+1} = s_2 \wedge [ts_b, ts_e] = \bigcap_{i=1}^{k+1}[ts_{bi}, ts_{ei}]$

TABLEAU 1.1. – Dépendances causales du modèle retenu

1.3.1.2. Arêtes et dépendances causales

Les arêtes dirigées du graphe de provenance modélisent les dépendances entre les nœuds, où le nœud source de l'arête représente la conséquence et le nœud de destination la cause. Les arêtes sont étiquetées pour préciser le type de causalité. Les dépendances causales introduites dans le OPM peuvent donc être divisées en deux catégories : les dépendances basiques et les dépendances composées.

Dépendances basiques Les dépendances basiques représentent les principes fondamentaux de la provenance d'un objet, répondant aux questions : par quel processus a-t-il été créé (wasGeneratedBy), par quel processus a-t-il été utilisé (used) et qui contrôle l'action (wasControlledBy). Afin de préciser la fonction d'un artefact ou d'un agent dans un processus, un rôle optionnel est introduit. On dénote $r \in \mathscr{R} \subset L$ un rôle et, comme définit dans MOREAU et al. 2011, il est spécifique au système. Ceci est utile pour différencier les artefacts ou les agents lorsque plusieurs entités interagissent via la même dépendance avec un processus.

Les trois dépendances basiques sont définies :
— wasGeneratedBy($d : \mathscr{O}, p : \mathscr{P}, r : \mathscr{R}$) liant l'artefact d au processus p qui l'a crée avec la fonction r.
— used($p : \mathscr{P}, d : \mathscr{O}, r : \mathscr{R}$) montrant la relation entre un processus p et un artefact d utilisé en tant que r pour la bonne exécution du processus.
— wasControlledBy($p : \mathscr{P}, s : \mathscr{A}, r : \mathscr{R}$) liant un processus p à un agent s qui a été nécessaire au bon déroulé de l'exécution du processus. L'agent peut avoir un rôle actif (lancer le processus, arrêter le processus) ou un rôle passif sans réelle interaction. Ce rôle est précisé par r.

Ces dépendances permettent de représenter dans le graphe les différentes évolutions des artefacts ainsi que les agents qui ont été nécessaires à l'existence d'un artefact, en participant de loin ou de près à l'exécution du processus.

Dépendances composées Pour avoir plus de précision sur le lien entre les objets et les processus, deux dépendances causales permettent de représenter visuellement dans le graphe si un artefact est dérivé d'un autre wasDerivedFrom ou si un processus a été déclenché par un autre wasTriggeredBy. Dans MOREAU et al. 2011, ces deux dépendances ne contenaient pas de rôle. En utilisant la définition du rôle donnée dans (BERTOLISSI, HARTOG et al. 2019) pour wasDerivedFrom , la définition des dépendances est la suivante :
— wasDerivedFrom($d_1 : \mathscr{O}, d_2 : \mathscr{O}, r : \mathscr{R}$) apparaît entre deux artefacts lors qu'un artefact $d1$ a été créé par un processus utilisant un second artefact $d2$ et l'artefact généré $d1$ n'a pu être généré que parce que l'artefact $d2$ existe (par exemple, un rapport ne peut exister que si des données ont été prélevées précédemment). Cette dépendance peut être formalisée à partir des dépendances de base : wasDerivedFrom$(d_1, d_2, r) = \exists p \in \mathscr{P}, r_i \in \mathscr{R}$: wasGeneratedBy$(d_1, p, r_i) \wedge$ used(p, d_2, r), signifiant $d1$ a été dérivé de $d2$. Le rôle r fait référence au rôle que l'artefact d_2 a joué dans la création de d_1.
— wasTriggeredBy($p_1 : \mathscr{P}, p_2 : \mathscr{P}$) représente que l'exécution d'un processus a été déclenchée par l'exécution d'un autre. La formalisation de cette dépendance est wasTriggeredBy$(p_1, p_2) = \exists d \in \mathscr{O}, r_1, r_2 \in \mathscr{R}$: used$(p_1, d, r_1) \wedge$ wasGeneratedBy(d, p_2, r_2).

1.3.1.3. Dépendances à étapes multiples

Afin de représenter les différents utilisateurs pouvant définir les conditions d'accès à un objet ou déterminer si un objet dérive d'une donnée personnelle, il est nécessaire de prendre en compte certains chemins spécifiques. C'est pour ces raisons que les auteurs de BERTOLISSI, HARTOG et al. 2019 ont étendu les dépendances présentes dans l'OPM avec des dépendances à étapes multiples.

Les dépendances de base sont ainsi étendues pour caractériser les liens entre des nœuds situés à une distance supérieure à 1. Ces dépendances à étapes multiples sont définies comme une conjonction de dépendances similaires, sans limite minimale ou

maximale quant à la longueur du chemin formé. On obtient ainsi les dépendances causales suivantes :

— La dépendance $\mathsf{wasGeneratedBy}^+ d : \mathscr{O}\, p : \mathscr{P}$ capture tous les processus ayant conduit à la génération de l'artefact d : $\mathsf{wasGeneratedBy}^+(d, p) = \exists p_i \in \mathscr{P}, r \in \mathscr{R} : \mathsf{wasGeneratedBy}(d, p_i, r) \wedge \mathsf{wasTriggeredBy}^+(p_i, p)$.

— La dépendance used^+ est définie comme $\mathsf{used}^+(p : \mathscr{P}, d : \mathscr{O}) = \exists p_i \in \mathscr{P}, r \in \mathscr{R} : \mathsf{wasTriggeredBy}^+(p, p_i) \wedge \mathsf{used}(p_i, d, r)$. Cette dépendance permet de retrouver tous les artefacts dont la bonne exécution du processus p dépend directement ou indirectement.

— $\mathsf{wasDerivedFrom}^+(d_1 : \mathscr{O}, d_2 : \mathscr{O})$ capture, pour un artefact, tous les artefacts utilisés pour le dériver, de manière directe ou indirecte. La définition formelle est $\mathsf{wasDerivedFrom}^+(d_1, d_2) = \mathsf{wasDerivedFrom}(d_1, d_2, r) \vee \exists d_i \in \mathscr{O}, r_i \in \mathscr{R} : \mathsf{wasDerivedFrom}(d_1, d_i, r_i) \wedge \mathsf{wasDerivedFrom}^+(d_i, d_2)$.

— La dépendance $\mathsf{wasTriggeredBy}^+$, définie comme $\mathsf{wasTriggeredBy}^+(p_1 : \mathscr{P}, p_2 : \mathscr{P}) = \mathsf{wasTriggeredBy}(p_1, p_2) \vee \exists p_i \in \mathscr{P} : \mathsf{wasTriggeredBy}(p_1, p_i) \wedge \mathsf{wasTriggeredBy}^+(p_i, p_2)$, retrouve tous les processus p_2 ayant mené au déclenchement de p_1.

Ces dépendances, au contraire de dépendances à une étape, ne contiennent pas de rôle. Pour des raisons de lisibilité, ces dépendances ne sont pas représentées graphiquement dans le graphe de provenance.

1.3.1.4. Extension temporelle

L'information de provenance donne un ordre partiel de l'exécution des processus ainsi que de la génération et de l'utilisation des artefacts. Lorsque l'on regarde un graphe de provenance, on peut savoir si un artefact a été créé avant un autre si un chemin de wasGeneratedBy et de used les relie. Autrement, en général, l'ordre n'est pas total. Dans certains contextes comme l'évaluation d'accès ou la vérification de la conformité, un ordre total peut être nécessaire.

Afin d'obtenir l'ordre total, une extension des arêtes avec des horodatages est proposée dans l'OPM. Chaque dépendance est étendue avec un ou des horodatages dans le cas de wasControlledBy. Nous supposons que le temps est mesuré par un observateur selon une horloge unique (ou des horloges synchronisées), de manière que les observations temporelles soient comparables. De plus, le temps observé doit être compatible avec les dépendances causales, par exemple, un artefact doit exister avant d'être utilisé, un processus génère des artefacts avant de se terminer, toutes les actions sont réalisées avant la fin de vie du système, etc. En d'autres termes, la causalité implique la précédence temporelle.

Pour représenter les horodatages, nous considérons une série de paramètres temporels ordonnés notés $ts_1, \ldots, ts_n \in \mathscr{T}$, avec $ts_1 < ts_2 < \cdots < ts_n$, qui représentent des points spécifiques dans le temps. De plus, nous introduisons deux symboles spéciaux pour désigner des temps non spécifiés : le symbole infini ∞) représentant un temps futur non spécifié, et le symbole astérisque $*$, représentant un temps non spécifié et non pertinent.

On note ts_{esy} le temps de fin de vie du système. Lorsque le système est toujours

en cours, on définit $ts_{esy} = \infty$. L'intervalle de temps $\mathcal{T}$ représente la durée de vie du système. Un processus s'exécute pendant un certain intervalle de temps $[ts_b, ts_e]$, avec $ts_b, ts_e \in \mathcal{T}$. Tous les artefacts utilisés ou générés le sont durant la fenêtre temporelle du processus.

Horodatages des dépendances causales de l'OPM Dans le modèle de provenance avec observations temporelles, l'arête wasControlledBy est étendue avec l'intervalle de temps $[ts_b, ts_e]$ (correspondant à la fenêtre d'exécution du processus). L'arête used est étendue avec un horodatage $ts_u \in [ts_b, ts_e]$ indiquant que l'artefact a été utilisé par le processus à l'instant ts_u. L'arête wasGeneratedBy est étendue avec $ts_g \in [ts_b, ts_e]$ indiquant que l'artefact a été créé par le processus à un instant donné ts_g.
Les dépendances composées sont également étendues avec un horodatage. La dépendance wasDerivedFrom est étendue avec l'horodatage $ts_g \in \mathcal{T}$ correspondant au moment où l'artefact a été généré. En d'autres termes, ts_g est le temps présent dans la composante wasGeneratedBy de la dépendance. Dans le cas de wasTriggeredBy, l'horodatage retenu est $ts_u \in \mathcal{T}$. Il indique le moment où l'artefact communiqué a été utilisé par la source (c'est-à-dire, ts_u correspond au paramètre temporel dans la composante used).

La Figure 1.4 présente un récapitulatif graphique des dépendances introduites dans l'OPM, tandis que la définition formelle est résumée dans la section *« Dépendances du Modèle Ouvert de Provenance »* du Tableau 1.1.

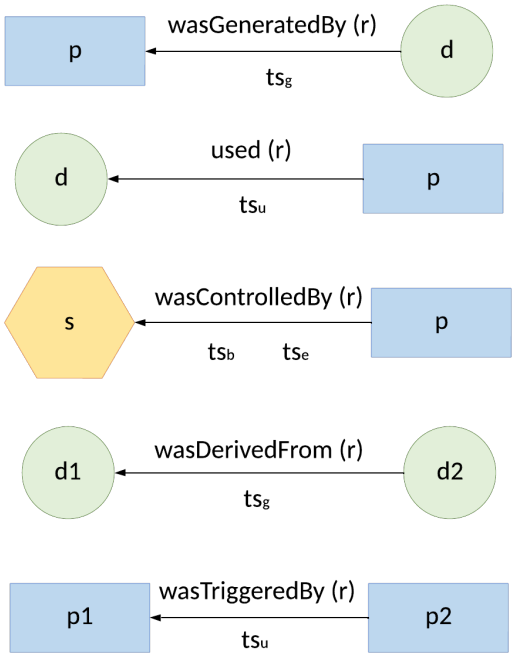


FIGURE 1.4. – Dépendances causales de l'OPM

Extension des horodatages pour les dépendances à étapes multiples Pour les dépendances à étapes multiples, nous considérons un intervalle qui indique la durée de la dépendance. Rappelons que la provenance est représentée dans le graphe par des arcs reliant un effet à sa cause. Lorsqu'on parcourt un chemin dans le graphe, on le fait en partant de l'effet le plus récent dans le temps pour remonter jusqu'à la cause la plus ancienne. L'horodatage de fin correspond à celui de la dernière étape (c'est-à-dire la plus ancienne dans le temps), tandis que l'horodatage de début est celui de la première étape (c'est-à-dire la plus récente dans le temps). Par exemple, les horodatages de $\mathsf{wasDerivedFrom}^+$ sont définis comme suit :

$$\begin{aligned}&\mathsf{wasDerivedFrom}^+(d_1:\mathscr{O}, d_2:\mathscr{O}, ts_b:\mathscr{T}, ts_e:\mathscr{T}) = (\mathsf{wasDerivedFrom}(d_1, d_2, r, ts_e)\\ &\wedge (ts_b = ts_e)) \vee (\exists d_i \in \mathscr{O}, r_i \in \mathscr{R}, ts_{e'} \in \mathscr{T} : \mathsf{wasDerivedFrom}(d_1, d_i, r_i, ts_e)\\ &\wedge \mathsf{wasDerivedFrom}^+(d_i, d_2, ts_b, ts_{e'}))\end{aligned}$$

Si l'on décompose cette écriture, nous avons : soit k un entier naturel, $\mathsf{wasDerivedFrom}^+(d_1, d_2, ts_b, ts_e) = \mathsf{wasDerivedFrom}(d_1, d_{i1}, r_{i1}, ts_e) \wedge \mathsf{wasDerivedFrom}(d_{i1}, d_{i2}, r_{i2}, ts_{b_{i2}}) \wedge \ldots \wedge \mathsf{wasDerivedFrom}(d_{ik}, d_2, r_{ik}, ts_b)$
Le temps donné par ts_b est le plus petit, car il appartient à l'objet créé en premier. On observe ainsi que l'intervalle donné par les horodatages correspond à la dernière étape pour celui de fin et à la première étape pour celui de début. Le Tableau 1.1 récapitule, dans la partie « *Dépendances à Étapes Multiples* » la définition formelle des dépendances à étapes multiples.

1.3.2. Extension du Modèle Ouvert de Provenance avec la notion de propriétaire

Dans le but de représenter un système collaboratif et utiliser la provenance, les auteurs de BERTOLISSI, HARTOG et al. 2019 proposent deux nouvelles dépendances owns et contributedTo. Ces dépendances sont définies en utilisant les dépendances causales de l'OPM, ainsi que l'extension multi-étapes présentée dans la Section 1.3.1.3.

1.3.2.1. Dépendances pour représenter la propriété

Lors de l'utilisation de données personnelles dans un système, il est nécessaire de connaître le propriétaire de ces informations. De même, dans les systèmes collaboratifs, certaines données sont conjointement détenues par plusieurs agents du systèmes (nommés contrôleurs). Afin de capturer la propriété et la contribution des agents directe ou indirecte à un processus, nous incluons les dépendances owns et contributedTo à notre modèle, construites à partir de la dépendance wasControlledBy :

— La dépendance owns lie un agent à une donnée créée par un processus contrôlé par cet agent en tant que propriétaire (« owner »). La définition formelle est $\mathsf{owns}(s:\mathscr{A}, d:\mathscr{O}) = \exists p \in \mathscr{P}, r \in \mathscr{R} : \mathsf{wasGeneratedBy}(d, p, r) \wedge \mathsf{wasControlledBy}(p, s, \texttt{owner})$. Cette définition repose sur l'hypothèse qu'un agent est propriétaire d'un artefact qu'il crée.

— Pour capturer l'implication d'un agent dans la création d'un objet, les auteurs de BERTOLISSI, HARTOG et al. 2019 définissent la dépendance $\mathsf{contributedTo}(s:\mathscr{A}, d:\mathscr{O}, \texttt{owner}) = \exists p \in \mathscr{P} : \mathsf{wasGeneratedBy}^{+}(d, p) \wedge \mathsf{wasControlledBy}(p, s, \texttt{owner})$. En d'autres termes, cette dépendance permet d'identifier les acteurs ayant exécuté un processus en tant que propriétaire impliqué indirectement dans la création d'un artefact. Ils sont considérés comme des contributeurs.

Dans l'article précédemment cité, la dépendance contributedTo ne capture que les agents ayant participé activement à la création de l'un des artefacts ayant conduit à l'artefact d, car seule la dépendance wasControlledBy avec le rôle « owner » est prise en compte. Cependant, dans le cadre des systèmes collaboratifs basés sur la communauté, en particulier sur les réseaux sociaux, certains utilisateurs peuvent contribuer de manière indirecte à un objet. Par exemple, la cible d'un tag n'est pas activement impliquée dans la création du tag, mais elle est nécessaire à la création du tag. Pour ce faire, nous étendons contributedTo à n'importe quel rôle $r \in \mathscr{R}$, afin de lier un processus à un agent impliqué passivement dans la création d'un artefact. Le rôle explicitement indiqué dans contributedTo précise de quelle manière un agent a été impliqué dans la création. Dans l'exemple du tag, la cible aura pour rôle *cible* (« *tag target* » en anglais). Comme indiqué dans la définition, si le rôle dans contributedTo est « owner », la dépendance causale a le même sens que celle dans BERTOLISSI, HARTOG et al. 2019.

1.3.2.2. Ajout des horodotages

Pour obtenir un ordre total, nous utilisons les définitions des horodatages fournis dans MOREAU et al. 2011 afin d'étendre les dépendances introduites dans cette section avec des observations temporelles. Nous supposons qu'un agent possède un artefact depuis sa génération, l'horodatage associé alors à la dépendance owns correspond au paramètre temporel dans la dépendance wasGeneratedBy marquant la création de l'artefact. De même, pour la dépendance contributedTo, son horodatage coïncide avec le temps associé à la génération de l'artefact.

Le Tableau 1.1 récapitule « *Dépendances pour Modéliser la Propriété* », les dépendances pour représenter la propriété, avec notre extension temporelle.

1.3.3. Extension pour des systèmes dynamiques et multi-propriétaires

Les systèmes collaboratifs basés sur la communauté se caractérisent par la dynamique des objets, des données et des agents, ainsi que par la complexité de leurs données. Cette complexité ne réside pas dans leur type (souvent du texte ou des images), mais dans leur manière d'être créés ou modifiés. Par exemple, des données peuvent être supprimées, et un utilisateur peut effacer son profil. De plus, dans un document créé de manière collaborative, il est difficile de définir un propriétaire global, ce qui peut rendre pertinent de désigner des propriétaires pour différentes parties du document (chapitre, section, paragraphe, etc.). De même, dans le cadre d'un réseau social, il est utile de lier un commentaire à la photo à laquelle il est associé, car sans cette photo, le commentaire perd son sens.

Pour répondre à ces problématiques, nous avons proposé une extension du modèle de provenance, présenté dans l'article BERTOLISSI, MARTINEZ ANTON et ZANNONE 2023 ainsi que dans une version journal qui est en cours de préparation. Nous étendons la provenance avec deux dépendances supplémentaires, dérivées des dépendances existantes : notAvailable et belongsTo. Pour ce faire, nous enrichissons les nœuds avec des attributs, puis nous définissons ces nouvelles dépendances. Nous présentons aussi les dépendances spécifiques au système.

1.3.3.1. Nœuds avec des attributs

Pour obtenir davantage d'informations sur les nœuds, nous supposons que ces nœuds peuvent contenir des attributs. Par exemple, afin de déterminer le type d'action d'un processus ou le type d'un artefact, les nœuds représentant les processus seront enrichis d'un attribut désignant l'action, tandis que les artefacts seront dotés d'un attribut indiquant leur type. Les attributs sont spécifiques aux systèmes et peuvent varier en fonction de l'utilisation de la modélisation.

Un attribut est défini comme une paire (a, v), où a représente le type d'attribut (par

exemple : *action*, *type*, *identité*, etc.), et v est la valeur associée à cet attribut pour un nœud particulier. Par exemple, si le processus pr_{908} est une instance du processus *supprimer*, ce processus possédera un attribut (*action*, *delete*). On note $\mathscr{A}$ l'ensemble fini d'attributs du système et $\mathscr{V}_a$ l'ensemble de valeurs que l'attribut a peut prendre. Pour l'attribut action l'ensemble $\mathscr{V}_{\text{action}}$ sera l'ensemble des actions définies dans le système.

Pour des raisons de lisibilité, ces attributs ne sont pas représentés graphiquement. Dans la suite de ce mémoire, un attribut a appartenant à un nœud n avec valeur v peut être noté $a = v$ ou (a, v) s'il n'y a pas de confusion quant à quel nœud il appartient. Sinon, le nœud n est inclus dans la notation comme suit : $n_a = v$ ou $a(n, v)$.

1.3.3.2. Dépendances notAvailable et belongsTo

Un système dynamique contient des données qui peuvent être supprimées avant la fin de vie de celui-ci. La provenance représentant un historique du système, il est impossible de venir supprimer sans violer ce principe des artefacts n'étant plus accessibles. Afin de modéliser ceci, nous introduisons la dépendance notAvailable pour indiquer qu'un artefact précédemment créé n'est plus disponible dans le système (c'est-à-dire que l'artefact a été supprimé). Cette dépendance est modélisée par un cycle de taille un dans le graphe de provenance. Cette dépendance apparaît si un processus dont son action est de *supprimer* utilise un artefact. On formalise ceci par

$$\text{notAvailable}(d : \mathscr{O}) = \exists p \in \mathscr{P}, r \in \mathscr{R} : \text{used}(p, d, r) \wedge \text{action}(p) = \texttt{delete}$$

où $\text{action}(p) = \texttt{delete}$ signifie que l'attribut $action$ du processus p vaut `delete`. La Figure 1.5 montre la représentation graphique de cette dépendance, où l'artefact d a été supprimé par le processus p le 2 février 2024.

Si l'on considère les objets comme étant des compositions de sous-objets (par exemple, un document est composé de sections, un post de texte principal et de commentaires, etc.), il peut être utile de modéliser ce lien entre sous-objets.

Pour caractériser explicitement la relation entre un artefact et ses sous-artefacts, nous introduisons donc un cas particulier de la dépendance wasDerivedFrom. Plus précisément, nous utilisons la dépendance belongsTo pour désigner l'inclusion d'un artefact dans un autre (comme un commentaire dans un post, une étiquette dans une figure, etc.). Un artefact sera inclut dans un autre selon l'action du processus qui crée le lien entre les deux, et donc spécifique au système. On dénote ce sous ensemble d'actions $\mathscr{V}_{action_bT} \subset \mathscr{V}_{action}$
Par exemple, dans le contexte des réseaux sociaux, les processus qui causeront l'existence de belongsTo sont *poster* un post sur un profil, *charger* une photo dans un album, *commenter* un post ou une photo ou *tagger* une photo. On aurait donc la formalisation

suivante avec $\mathcal{V}_{action_bT} = \{\texttt{post}, \texttt{upload}, \texttt{comment}, \texttt{tag}\}$:

$$\begin{aligned}&\mathsf{belongsTo}(d_1 : \mathcal{O}, d_2 : \mathcal{O}) = \exists p \in \mathcal{P}, r_i, r_j \in \mathcal{R}, : \mathsf{wasGeneratedBy}(d_1, p, r_i)\\ &\wedge \mathsf{used}(p, d_2, r_j) \wedge \mathsf{action}(p) \in \mathcal{V}_{action_bT}\end{aligned}$$

Comme pour les autres dépendances, nous étendons notAvailable et belongsTo avec des observations temporelles. La dépendance notAvailable peut être étendue avec à l'horodatage de la composante used de la dépendance. Il indique le moment où l'artefact a été supprimé. Comme pour la dépendance wasDerivedFrom, belongsTo prend l'horodatage de la composante wasGeneratedBy.



FIGURE 1.5. – Dépendance notAvailable

Le Tableau 1.1 résume la formalisation de ces deux dépendances avec les horodatages, dans la partie « *Dépendances pour les systèmes dynamiques et multi-propriétaires* ».

1.3.3.3. Dépendances spécifiques

En combinant de façons différentes les dépendances du OPM et les extensions présentées précédemment, il est possible d'établir des dépendances spécifiques à un système. Dans le contexte des systèmes collaboratifs centrés sur la communauté, il peut être bénéfique de capturer à l'aide d'une dépendance les différents détenteurs d'une donnée ou dans le cadre du contrôle d'accès de ces systèmes identifier toutes les personnes dont l'avis doit être pris en compte pour accorder l'accès à une donnée particulière. On appelle ces derniers les contrôleurs de la donnée (« controllers »).

Par exemple, considérons un réseau social avec deux types de contrôleurs c pour les photos : le propriétaire de la photo et les utilisateurs identifiés. Nous pouvons définir le prédicat controller pour une photo comme suit :

$$\mathsf{controller}(c, photo) = (\mathsf{owns}(c, photo)) \vee (\mathsf{contributedTo}(c, photo, "tag\ target"))$$

La Figure 1.6 est un exemple d'un graphe de provenance modélisant un petit réseau social contenant trois utilisateurs Alice, Bob et Christopher. Chacun possède un profil. Alice a chargé une photo $p1$ le 20 novembre 2023 dans son album et elle a identifié (« taggé ») ces deux amis Bob et Christopher. Les contrôleurs capturés par controller pour la photo $p1$ sont Alice en tant que *propriétaire*, et Christopher et Bob en tant que *utilisateurs identifiés* (« tag target »).

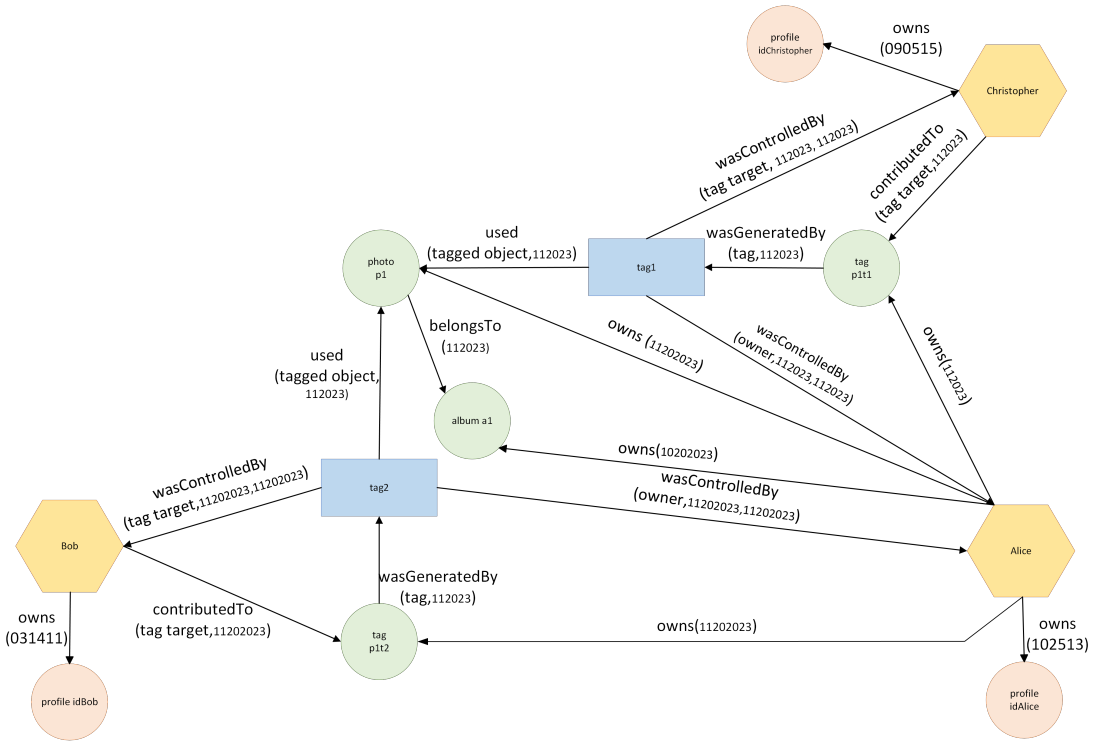


FIGURE 1.6. – Graphe de provenance d'un graphe social

1.3.4. Utilisations du Modèle Ouvert de Provenance

Le Modèle Ouvert de Provenance (OPM) a fait ses preuves en démontrant son efficacité à travers diverses applications dans de nombreux domaines. Sa flexibilité et sa robustesse le rendent adapté pour tracer et vérifier les processus dans des secteurs tels que l'analyse, l'audit, la traçabilité des données, et les contrôles d'accès.

Par exemple, dans BUTIN et al. 2016, les auteurs proposent d'utiliser la provenance pour assurer la traçabilité des données et garantir la conformité du traitement aux politiques d'utilisation dans le cadre des grandes organisations. La notion de traçabilité, rendue possible par la provenance, est également appliquée dans les workflows et les bases de données pour suivre l'origine des données, vérifier leur intégrité, et exécuter des requêtes complexes pour analyser les processus scientifiques comme expliqué dans LIM et al. 2011. Ces aspects sont particulièrement importants en fonction de la nature des données, comme dans le cas des données médicales ou expérimentales.

La provenance des données est également un sujet de recherche important dans l'Internet des objets (IoT), où elle aide à résoudre des problèmes liés à la gestion des identités, la fiabilité des données, la détection de comportements anormaux, et le contrôle d'accès comme illustré par le travail R. HU et al. 2020.

En outre, l'utilisation de la provenance pour les contrôles d'accès se retrouve dans d'autres domaines, tels que les systèmes collaboratifs où les données proviennent de sources variées (BERTOLISSI, HARTOG et al. 2019), ainsi que dans des contrôles d'accès plus génériques comme le Provenance-Based Access Control (PBAC) (PARK

et al. 2012). Ce dernier modèle utilise les activités et les usages passés des utilisateurs et des données pour évaluer les accès, et peut être étendu pour supporter la séparation dynamique des fonctions (NGUYEN et al. 2013).

Étant donné son utilisation répandue, de nombreux travaux se sont également concentrés sur les méthodes de sécurisation et de protection de ce type de données (HASAN et al. 2007 ; LEE et al. 2015).

1.4. Représentation de la communauté des utilisateurs

Une partie centrale des systèmes collaboratifs centrés sur la communauté est la communauté formée par les différents utilisateurs. On parle de communauté car, comme défini dans la Section 1.2, les relations interpersonnelles sont au cœur des interactions et du déroulé des actions. Afin de raisonner sur ces relations pour protéger au mieux la vie privée de ces utilisateurs, il est nécessaire de définir un modèle pour les formaliser. En raison de l'utilisation d'un graphe pour modéliser l'information en lien avec la provenance, nous avons fait le choix d'utiliser aussi un graphe dirigé et étiqueté pour modéliser les relations utilisateurs, comme présenté dans BERTOLISSI, MARTINEZ ANTON et ZANNONE 2023 et MARTINEZ ANTON et al. 2024. On note ce graphe dirigé et étiqueté $S = (\mathscr{A}, \mathscr{E}_{social}, L_{social})$ où $\mathscr{A}$ sont les nœuds correspondant à l'ensemble d'agents du graphe de provenance, $\mathscr{E}_{social}$ les arêtes et L_{social} un ensemble d'étiquettes. Tout comme pour la provenance, les étiquettes permettent de préciser la nature des relations, tandis que les horodatages offrent une vision plus fine de l'historique des relations.
Dans cette section, nous détaillons les composantes de ce graphe, les différentes étiquettes qu'il peut contenir, ainsi que leur possible généralisation.

1.4.1. Graphe des relations

Le graphe représentant les relations inter-personnelles des utilisateurs est nommé le graphe des relations.

Les nœuds de ce graphe représentent les utilisateurs du système. Ces utilisateurs correspondent aux agents du OPM dans le cadre des systèmes collaboratifs centrés sur la communauté. On les représente donc par des hexagones jaunes dans le graphe et on note l'ensemble des utilisateurs/agents $\mathscr{A}$.

Lorsqu'une arête existe entre deux nœuds, cela signifie qu'il existe une relation personnelle entre les agents représentés par les nœuds. Afin de préciser la nature de cette relation, nous utilisons différents types d'arêtes et d'étiquettes, pouvant représenter des relations symétriques ou asymétriques. En plus des étiquettes des arêtes, comme pour les dépendances définies dans la Section 1.3, ces prédicats comportent un rôle, qui est spécifique au sytème et varie selon les objectifs. Par exemple, l'ensemble de roles d'un réseau social $\mathscr{R} \subset L_s$ peut contenir des rôles tels que ami, ami proche, famille, abonné, etc.

Relation asymétrique Au sein de certains systèmes, on peut voir l'existence de relations à sens unique. En effet, sur certains réseaux sociaux un agent/utilisateur peut être *abonné* à un autre utilisateur, sans que l'autre soit abonné en retour (« *follower* »). Nous introduisons donc le prédicat isRelated où le label de rôle $r \in \mathscr{R}$ qualifie la relation sociale entre les deux agents dans $\mathscr{A}$. Comme pour les dépendances de la provenance, on peut vouloir capturer des agents via un prédicat qui sont à distance deux ou plus d'un utilisateur en suivant une même relation. On définit pour cela le prédicat $\mathsf{isRelated}^k$ utilisés pour capturer des relations composées du même type, limitées à une profondeur donnée k. Formellement,

$$\mathsf{isRelated}^k(s_2 : \mathscr{A}, s_1 : \mathscr{A}, r : \mathscr{R}) = \exists u_1, \ldots, u_{k+1} \in \mathscr{A} \text{ s.t. } \forall i \in \{1, \ldots k+1\}$$
$$\mathsf{isRelated}(u_i, u_{i+1}, r) \wedge u_1 = s_1 \wedge u_{k+1} = s_2$$

Par exemple, $\mathsf{isRelated}^2(s_1, s_2, \text{abonné})$ capture tous les agents s_2 tels que s_1 est abonné à s_2, où il existe un agent s tel que s_1 est abonné à s et s est abonné à s_2. De même $\mathsf{isRelated}^+$ capture les relations composées de même type sans limite de profondeur.

Relation symétrique Lorsqu'une relation est partagé entre les deux nœuds on appelle cela une relation symétrique. Le prédicat areRelated est la clôture symétrique de isRelated, ce qui nous permet de capturer ces relations bidirectionnelles, comme par exemple la relation ami sur Facebook. Nous obtenons ainsi

$$\mathsf{areRelated}(s_1 : \mathscr{A}, s_2 : \mathscr{A}, r : \mathscr{R}) = \mathsf{isRelated}(s_1, s_2, r) \wedge \mathsf{isRelated}(s_2, s_1, r)$$

Comme pour $\mathsf{isRelated}^k$ et $\mathsf{isRelated}^+$, $\mathsf{areRelated}^k$ est utilisé pour capturer des relations composées de même type et limitées à une profondeur k et $\mathsf{areRelated}^+$ pour celles sans limite de profondeur. Par exemple, $\mathsf{areRelated}^2(s_1, s_2, \text{ami})$ indique une relation d'ami d'ami entre s_1 et s_2.

Durée de vie d'une relation De même que pour les dépendances causales, nous introduisons les horodatages pour suivre l'évolution des interactions entre les utilisateurs. Pour l'arête isRelated, nous ajoutons un intervalle de temps $[ts_b, ts_e]$ avec $ts_b, ts_e \in \mathscr{T}$, $\mathscr{T}$ étant la période de vie du système, décrivant la fenêtre pendant laquelle la relation entre les agents a eu lieu. Nous utilisons l'intervalle de temps $[ts_b, \infty]$ pour indiquer qu'une relation est en cours.

Pour les autres prédicats, les intervalles de temps sont définis comme l'intersection de tous les intervalles associés à chaque composante. Par exemple,

$$\mathsf{areRelated}(s_1 : \mathscr{A}, s_2 : \mathscr{A}, r : \mathscr{R}, ts_b : \mathscr{T}, ts_e : \mathscr{T}) = \exists ts_{b1}, ts_{e1} \ldots, ts_{bk+1}, ts_{ek+1} \text{ s.t.}$$
$$\forall i \in \{1, \ldots k+1\} \mathsf{isRelated}(s_1, s_2, r, ts_{b1}, ts_{e1}) \wedge \mathsf{isRelated}(s_2, s_1, r, ts_{b1}, ts_{e1})$$
$$\wedge [ts_b, ts_e] = \bigcap_{i=1}^{k+1} [ts_{bi}, ts_{ei}]$$

Cela signifie que le prédicat areRelated est annoté avec l'intervalle de temps $[ts_b, ts_e]$ indiquant la période pendant laquelle les deux relations unidirectionnelles composantes isRelated sont actives. Si nous considérons une relation ami $\in \mathscr{R}$ entre deux utilisateurs s_1 et s_2, les agents s_1 et s_2 sont liés en tant qu'amis par une dépendance areRelated pendant la période où à la fois s_1 est ami avec s_2 et s_2 est ami avec s_1.
La même idée s'applique aux dépendances à étapes multiples isRelated^k et areRelated^k (ainsi qu'à isRelated$^+$ et areRelated$^+$) où l'intervalle de temps associé à la dépendance à étapes multiples est défini comme l'intervalle de temps pendant lequel toutes les étapes du prédicat coexistent. Lorsque des conditions par rapport au temps ne sont pas nécessaires, il est possible de ne considérer que les relations ayant des intervalles infinis, permettant ainsi d'avoir une capture instantanée des relations du système. Les horodatages permettent cependant de pourvoir créer des politiques d'accès plus fines (cf. Chapitre 2).

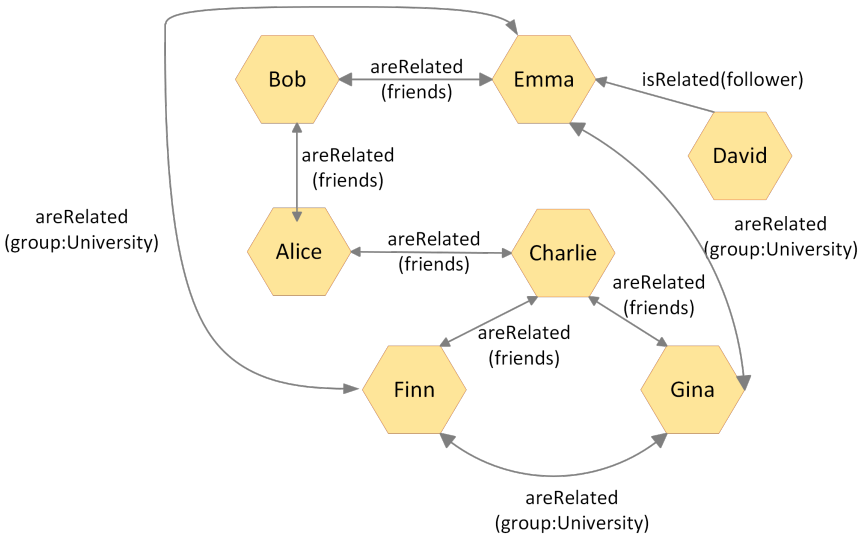


FIGURE 1.7. – Graphe des relations

La définition formelle de ces dépendances est synthétisée dans la dernière section du Tableau 1.1 et la Figure 1.7 donne une représentation graphique de isRelated et de areRelated dans le contexte d'un réseau social. On y voit que David suit Emma, ce qui est représenté par la relation asymétrique isRelated, tandis que toutes les autres relations (par exemple, l'amitié entre Alice et Bob) sont symétriques. Dans ce cas, les horodatages ne sont pas précisés, indiquant que ce graphe représente une capture instantanée des relations existantes à ce moment-là. Cependant, la Figure 1.1 comporte des horodatages. Par exemple, Alice et Edward ne sont plus amis depuis le 7 août 2022.

On remarque également que les membres d'un groupe forment une clique, comme c'est le cas pour le groupe universitaire formé par Finn, Gina et Charlie, représenté par les arêtes areRelated(*group :University*).

1.4.2. Définition de communauté

Dans la littérature, une communauté virtuelle est définie comme étant *« un réseau socio-technique persistant et durable d'individus qui partagent et développent une base de connaissances, un ensemble de croyances, de valeurs, d'histoires et d'expériences qui se chevauchent. »*[14] (BARAB et al. 2004). En d'autres mots, une communauté d'utilisateurs au sein d'un système collaboratif est un groupe de personnes qui partagent des intérêts, des objectifs ou des activités communs et échangent des informations, des idées et des expériences. Pour définir ces groupes, on peut utiliser les relations précédemment introduites. Ainsi, nous distinguons deux types de communautés : les *communautés de groupe* et les *communautés centrées sur un utilisateur*. Une communauté de groupe est constituée d'agents/utilisateurs appartenant à un même groupe et partageant un point commun. Ces communautés sont représentées par une clique dans le graphe des relations. Une communauté centrée sur un utilisateur forme un groupe d'utilisateurs partageant un même type de relation avec un utilisateur central. Graphiquement, cela est représenté par un sous-graphe en étoile, dont le centre est l'utilisateur.

La définition formelle de ces deux types de communauté est donnée en Définition 1.

Définition 1 *(Communauté)* Soit rel une relation binaire entre utilisateurs et $\mathscr{A}$ l'ensemble des utilisateurs.

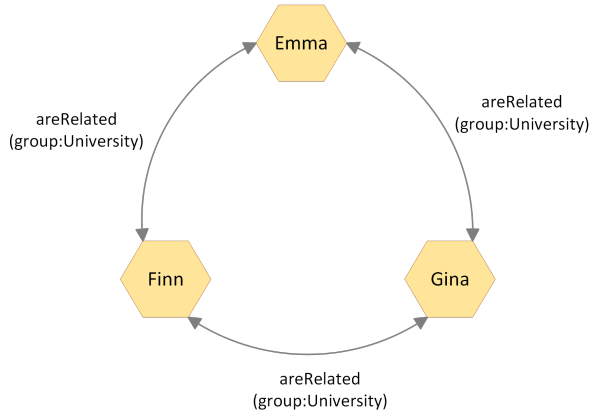
— La **communauté de groupe** associée à rel est définie comme $Com(rel) = \{u, u' \in \mathscr{A} \mid rel(u, u')\}$.

— Une **communauté centrée sur un utilisateur** de rayon k est définie comme $Com_u(rel, k) = \{u' \in \mathscr{A} \mid rel(u, u')$ et $|u, u'| \leq k\}$ où $|u, u'|$ est la longueur du plus court chemin entre u et u'.

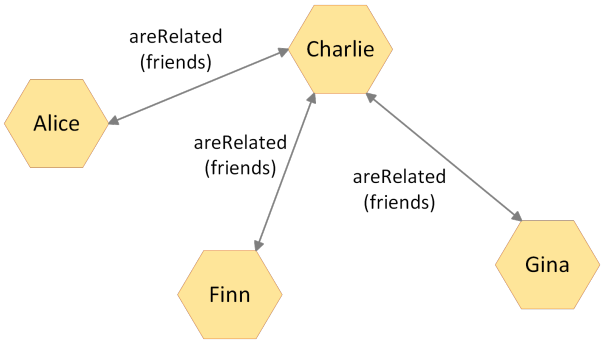
La Figure 1.8 représente une communauté de groupe (graphe a) et une communauté centrée sur un utilisateur (graphe b). Ces graphes sont extraits du réseau social de la Figure 1.7. Par exemple, la communauté de groupe avec la relation $rel =$ areRelated$(group : univ)$ est définie par $Com(rel) = \{Emma, Finn, Gina\}$. La communauté centrée sur Charlie avec un rayon de $k = 1$, construite en utilisant la relation $rel =$ isRelated(ami), est définie par $Com_{Charlie}(friends, 1) = \{Alice, Finn, Gina\}$.

Étant donné que les communautés impliquent intuitivement une proximité entre leurs utilisateurs, il est raisonnable de fixer une limite supérieure à la longueur de la relation rel. Cette limite est spécifique au système et est notée $maxDis$, telle que $k \leq maxDis$. Il est utile de remarquer que k correspond à l'indice k des relations isRelated^k et areRelated^k.

14. En anglais, « a persistent, sustained [social-technical] network of individuals who share and develop an overlapping knowledge base, set of beliefs, values, history, and experiences ».



(a) Communauté de groupe



(b) Communauté centrée sur Charlie

FIGURE 1.8. – Exemple de communautés en suivant la Définition 1

1.4.3. Pondération des arêtes pour modéliser la confiance

Dans les systèmes collaboratifs centrés sur la communauté, ces communautés se fondent sur des interactions réelles et authentiques entre les individus. Un concept fondamental dans les relations humaines est la confiance, qui joue un rôle crucial dans la qualité et la profondeur des échanges.

Confiance entre nœuds voisins La confiance en une personne peut être définie comme la croyance que cette personne agit de manière sincère et ne causera aucun tort. Dans les systèmes centrés sur la communauté, la confiance ou la force du lien (terme employé dans SUCH et al. 2015) représente le degré de confiance qu'un utilisateur a envers un autre. Représenter la confiance dans le graphe des relations permet de capturer et d'analyser les relations de manière plus nuancée et réaliste.
En pondérant les arêtes du graphe par le niveau de confiance, on peut visualiser non seulement la structure des interactions, mais aussi la force et la fiabilité des liens entre les membres de la communauté.

Pour certains types de relations, une étiquette supplémentaire sera donc ajoutée

pour déterminer le niveau de confiance associé à cette relation. La nécessité d'étiqueter ces relations et le type de relation à étiqueter dépend des spécificités de chaque système. Par exemple, dans un réseau social, les arêtes représentant les relations de type *amis* seront étiquetées pour refléter le niveau de confiance dans cette amitié [15]. Pour chaque nouvelle amitié, l'utilisateur u donnera une valeur de confiance $t_{(u,u')} \in [0,1]$, signifiant que u a une confiance de $t_{(u,u')}$ en u'. Cette valeur n'est pas symétrique, on peut avoir $t_{(u,u')} \neq t_{(u',u)}$. La Figure 1.9 étend le graphe des relations introduit dans la Figure 1.7 avec les confiances de Alice, Bob et Charlie. Par exemple Alice a une confiance de 0,7 en Bob, donc une confiance plutôt élevée. Pour des raisons de lisibilité, les confiances de Emma, Finn et Gina n'ont pas été explicitées.

Confiance entre deux nœuds distants Dans le but de pouvoir étendre cette confiance aux relations de type $\mathsf{isRelated}^k$, il est nécessaire de pouvoir calculer ou estimer la confiance pour des nœuds distants dans le graphe de relations. Quantifier ce concept entre des utilisateurs qui ne se connaissent pas est difficile et a conduit à la conception d'algorithmes dédiés en utilisant des graphes. L'article JIANG, G. WANG et al. 2017 présente une revue de ces algorithmes. Deux approches sont étudiées dans cet article : l'approche basée sur la simplification des graphes et l'approche basée sur l'analogie des graphes. La première repose sur la simplification d'un graphe en supprimant les chemins les moins pertinents et en aggrégant le reste de l'information pour trouver une estimation de la confiance. La deuxième approche utilise l'analogie entre plusieurs graphes pour prédire la confiance. Dans le cadre de notre travail, nous choisissons une approche basée sur la simplification, car nous disposons d'un unique graphe contenant de l'information sur la confiance entre utilisateurs. Plusieurs algorithmes existent avec ce type d'approche.

TidalTrust (GOLBECK et al. 2005) utilise une approche de filtrage basée sur les chemins pour agréger les valeurs de confiance et les recommandations au sein du réseau. L'algorithme cherche tous les chemins les plus courts entre deux utilisateurs pour calculer le niveau de confiance. Il agrège ensuite les valeurs en appliquant une fonction de confiance à travers ces chemins pour estimer la confiance globale. MoleTrust (AVESANI et al. 2005) propose une approche intégrée qui prend en compte à la fois les relations directes et indirectes pour évaluer la confiance au sein du réseau. L'algorithme calcule les niveaux de confiance à travers le réseau en propageant la confiance de manière itérative à partir des nœuds de confiance directe. Les niveaux de confiance sont ajustés en fonction de la proximité des nœuds dans le réseau, favorisant les liens de proximité immédiate pour améliorer la précision de la propagation de la confiance. L'algorithme SWTrust (JIANG et G. WANG 2011) combine les caractéristiques des réseaux de type "small-world" et des segments de confiance pour la propagation de la confiance. Pour ce faire, il segmente le réseau en clusters et utilise les liens intra-clusters et inter-clusters pour propager la confiance. Il prend en compte les liens

15. Au sein d'un groupe, il est possible que les utilisateurs ne connaissent pas tous les membres de manière directe. Par conséquent, demander à chaque utilisateur d'évaluer son niveau de confiance envers chaque autre membre du groupe serait non seulement peu pertinent, mais aussi extrêmement fastidieux.

forts au sein des clusters et les liens faibles entre les clusters pour une propagation efficace de la confiance. MeTrust (LIU et al. 2013) évalue la confiance en prenant en compte les interactions antérieures, les caractéristiques personnelles et le contexte des relations. Il ajuste les niveaux de confiance en fonction de la similarité et des interactions historiques entre les utilisateurs.

Dans notre modélisation, différents types de relations sont représentés dans le graphe, comme l'appartenance à un groupe, qui est modélisée par des arêtes. Cependant, comme mentionné précédemment, il ne serait pas pertinent de pondérer ces arêtes. Par conséquent, utiliser ces chemins pour estimer la confiance entre deux utilisateurs distants ne serait pas non plus approprié. Les arêtes doivent donc être filtrées en fonction du label de la relation.

Pour la suite de ce mémoire, nous retenons donc l'utilisation de TidalTrust, car cet algorithme peut facilement être étendu avec un filtrage sur les relations à considérer. Dans notre exemple, les relations à considérer sont celles avec l'étiquette « ami ».

La confiance entre deux nœuds du graph est alors définie comme :

Définition 2 (Confiance entre utilisateurs) *Pour calculer la confiance entre deux utilisateurs connectés à n'importe quelle distance, l'algorithme TidalTrust procède en plusieurs étapes. Tout d'abord, il identifie tous les chemins les plus courts entre ces deux utilisateurs, en prenant en compte uniquement les relations autorisées, une extension spécifique que nous avons introduite.*
Ensuite, il combine de manière récursive les niveaux de confiance obtenus sur ces chemins, en filtrant ceux qui offrent une confiance suffisamment élevée pour être pertinente. C'est là qu'intervient la condition max, qui peut être définie en fonction des niveaux de confiance trouvés sur les chemins ou fixée par le système. Cette deuxième étape est formalisée par la formule suivante : $\mathsf{Utrust} : \mathscr{A} \times \mathscr{A} \to [0,1] \subset \mathbb{R}$ *:*

$$\mathsf{Utrust}(u1,u2) = \frac{\sum_{j \in R \mid t_{(u_1,j)} \geq max} t_{(u_1,j)} \times \mathsf{Utrust}(j,u_2)}{\sum_{j \in R \mid t_{(u_1,j)} \geq max} t_{(u_1,j)}}$$

où $\mathscr{A}$ sont les nœuds dans le graphe de relations et $t_{(u,u')}$ représente la confiance entre nœuds voisins, c'est-à-dire entre deux utilisateurs connectés via une relation autorisée (par exemple en tant qu'amis) à une distance de 1 dans le ce même graphe. On rappelle que $t_{(u,u')} \in [0,1]$.

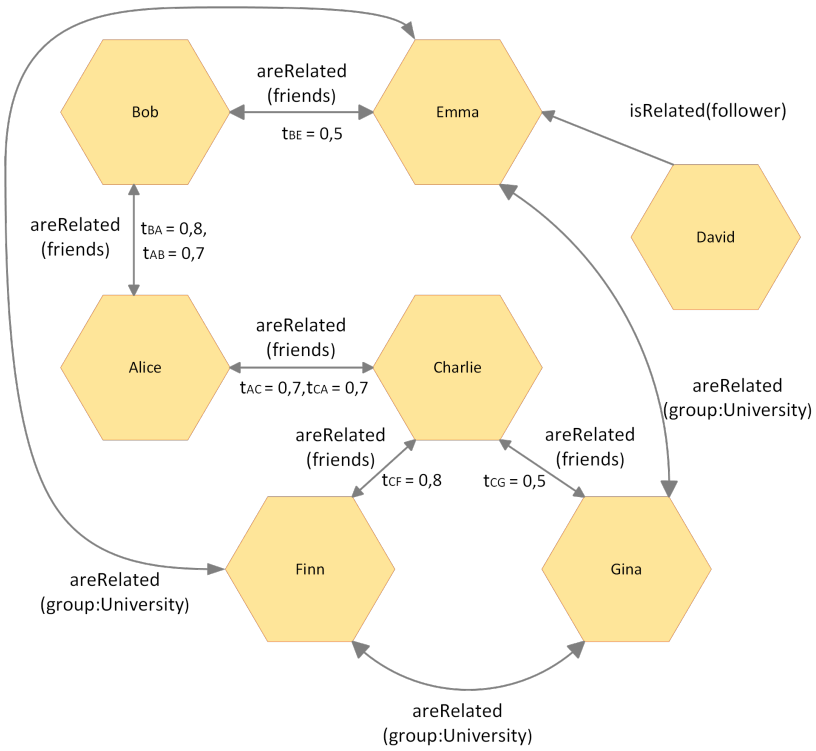


FIGURE 1.9. – Graphe des relations étendu avec la confiance

Voici un exemple montrant comment on peut calculer la confiance en utilisant l'algorithme TidalTrust.

Exemple 1 *Sur la base du scénario de la figure 1.9, dans le tableau 1.2, nous reportons les valeurs de confiance (soulignées) entre un contrôleur et ses amis.*

Les valeurs de confiance restantes sont calculées à l'aide de la fonction Utrust.

Si nous nous concentrons sur Alice, les valeurs de confiance pour David, Emma, Finn et Gina doivent être calculées. Il n'existe qu'un seul chemin le plus court (amis) d'un utilisateur à un autre dans ce scénario. En nous concentrant sur Alice, elle n'est amie qu'avec Bob et Charlie, donc les valeurs pour David, Emma, Finn et Gina doivent être calculées. D'Alice à Emma, il existe un chemin unique qui est

$$\mathsf{areRelated}(Alice, Bob, friend) \wedge \mathsf{areRelated}(Bob, Emma, friend).$$

La confiance d'Alice en Emma est :

$$\mathsf{Utrust}(Alice, Emma) = \frac{t_{(Alice,Bob)} \times \mathsf{Utrust}(Bob, Emma)}{t_{(Alice,Bob)}} = \frac{0.7 \times 0.5}{0.7} = 0.5$$

Ici $\mathsf{Utrust}(Bob, Emma) = t_{(Bob,Emma)}$ *car Bob et Emma sont amis. De même, on obtient pour Finn et Gina,* $\mathsf{Utrust}(Alice, Finn) = 0.8$ *et* $\mathsf{Utrust}(Alice, Gina) = 0.4$. *De même,* $\mathsf{Utrust}(Alice, Finn) = 0.8$ *et* $\mathsf{Utrust}(Bob, Finn) = 0.8$.

Si nous regardons Bob (ou Charlie), le chemin le plus court de lui à n'importe quel utilisateur a une longueur de 3, deux étapes sont alors effectuées pour trouver la valeur de confiance (d'après le calcul précédent, nous avons $\mathsf{Utrust}(A,F) = 0.8$*) :*

Par exemple,

$$\mathsf{Utrust}(Bob,Finn) = \frac{t_{(Bob,Alice)} \times \mathsf{Utrust}(Alice,Finn)}{t_{(Bon,Alice)}} = 0.8$$

D'après le calcul précédent, nous avons $\mathsf{Utrust}(A,F) = 0.8$*.Le tableau 1.2 donne toutes les valeurs* Utrust *calculées dans le scénario présenté dans la Figure 1.9.*

	A	**B**	**C**	**D**	**E**	**F**	**G**
A	x	0.7	0.7	0	0.5	0.8	0.4
B	0.8	x	0.7	0	0.5	0.8	0.4
C	0.7	0.7	x	0	0.5	0.8	0.4

TABLEAU 1.2. – Confiance entre utilisateurs

Malgré la facilité à comprendre le fonctionnement de la provenance et des relations entre utilisateurs, il n'est pas trivial de déduire ces informations des journaux ou des événements. En effet, les logs ou journaux sont centrés sur les différents processus exécutés et récupérer les informations pertinentes des événements du système peut être compliqué.
Cependant de nombreux travaux ont été dédiés à la construction de graphes d'information à partir des journaux du système. On retrouve aussi des travaux sur la capture de la provenance depuis les évnements. Ces différentes propositions montrent la faisabilité de formaliser un système en graphe de provenance et de relations.

1.5. Capture de la provenance et des relations pour la construction du graphe

Capturer la provenance et les relations est essentiel pour construire des graphes précis, permettant d'avoir une modélisation cohérente qui reflète fidèlement les évènements du système et ainsi pouvoir raisonner au-dessus du graphe. Cependant, cette tâche n'est pas toujours triviale : les informations doivent être capturées avec une granularité suffisante pour voir les interactions complexes entre les différents composants du système. De plus, intégrer ces informations dans un graphe nécessite souvent de surmonter des défis techniques, tels que la gestion des grandes quantités de données, le maintien de la cohérence des informations et la gestion des diverses sources de provenance. L'OPM, étant un modèle indépendant de toute implémentation, ne propose pas de solution pour ces problématiques.
Cependant, divers travaux présents dans la littérature permettent de surmonter ces

défis. En particulier, dans le domaine du process mining, des techniques de conversion des logs en graphes ont été développées. Par ailleurs, d'autres recherches se concentrent directement sur la capture de la provenance, en extrayant toutes les informations nécessaires des évènements du système. Dans la Section 1.5.1, nous présentons des travaux qui abordent la conversion des journaux du système en graphe. Puis en Section 1.5.2, nous abordons les travaux ayant été fait pour capturer la provenance.

1.5.1. Conversion de journaux en graphe

Le process mining (ou extraction de processus) est une technique analytique qui permet d'explorer, de visualiser et d'analyser les processus d'un système en utilisant les données collectées dans les systèmes d'information. Un des principaux aspects du process mining est l'extraction des données à partir des journaux des évènements qui se sont produits dans le système. Cette extraction a lieu afin de pouvoir ensuite analyser les données extraites. De nombre travaux explorant différentes stratégies ont été proposés comme le montre l'étude DAKIC et al. 2020. Une des pistes explorées dans ce contexte là est l'utilisation de graphes, pour représenter les données extraites. Par exemple, dans (KING et al. 2005), ils proposent un outil, BackTraker, pour suivre le flux d'informations afin de détecter le point d'intrusion dans une attaque. Leur outil est basé sur un graphe de dépendance construit à partir des logs du système. Pour la construction du graphe, ils proposent un algorithme pour transformer un ensemble de logs d'évènements en un graphe de dépendance. D'autres travaux permettent de passer de logs stockés dans des bases de données à des graphes de connaissance (« knowledge graphs » en anglais) comme XIONG et al. 2023.

1.5.2. Capture de la provenance

Des travaux plus spécifiques à la provenance ont aussi été proposés. Dans ces techniques, les évènements du système sont analysés pour extraire les informations spécifiques à la provenance. Par exemple, le Provenance-Aware System (PASS) (MUNISWAMY-REDDY et al. 2006) est un système de stockage qui collecte et maintient automatiquement la provenance ou la lignée, c'est-à-dire l'historique complet ou l'ascendance d'un élément. Le système PLUS (CHAPMAN et al. 2011) est un magasin de provenance multi-organisationnel qui collecte la provenance à partir de systèmes hétérogènes, permettant de capturer la provenance au niveau de l'application. Comme la provenance est un élément de plus en plus utilisé pour la traçabilité des données, de nombreux travaux étudient de comment capturer la provenance dans le cadre du Resource Description Framework (RDF) (SIKOS et al. 2020).

1.6. Conclusion et discussion

Dans cette section, nous avons défini un modèle permettant de formaliser les systèmes collaboratifs centrés sur la communauté, ou tout autre système informatique manipulant des données personnelles afin de pouvoir répondre aux défis introduits par l'exposition des données privées.

Pour la formalisation, nous proposons d'utiliser la provenance des données ainsi que les relations au sein d'une communauté, si nécessaire. Cette modélisation comporte deux types de graphes, le graphe de provenance G et le graphe des relations S. On peut combiner ces graphes en un seul, nommé *graphe d'évaluation* dans la suite, qui est défini comme $G_E = (\mathcal{N}, \mathcal{E} \cup \mathcal{E}_{social}, L \cup L_{social})$.
Les nœuds peuvent être étendus avec des attributs pour avoir une plus grande précision. Par exemple, le nœud représentant l'instance d'un processus peut contenir un attribut donnant l'action réalisé par ce processus.
Les dépendances attachées aux arêtes peuvent inclure des rôles pour préciser les relations entre les nœuds, ainsi que des horodatages pour établir un ordre total dans la provenance ou maintenir un historique des relations entre utilisateurs. Par la suite, nous utilisons dans certains cas une notation compacte de ces dépendances pour gagner en clarté : pour considérer un rôle spécifique ou ignorer l'étiquette de rôle d'une dépendance (causale ou représentant une relation) R, nous utilisons respectivement $R^r(n, n', T) = R(n, n', r, T)$, où r est l'étiquette de rôle et T est un horodatage (ou une paire d'horodatages), et $R^\circ(n, n', T) = \exists r \in \mathcal{R} : R(n, n', r, T)$ (pour tous les $n, n' \in \mathcal{N}$).
De plus, selon le type de relation, une valeur de confiance peut également être attachée. Des algorithmes existent pour généraliser la confiance entre deux nœuds.
La construction des graphes de provenance et de relations est réalisable, en effet de nombreux travaux sont présent dans la littérature montrant comment passer des évènements d'un système ou des journaux à un graphe.
Notre modélisation offre une flexibilité en permettant d'utiliser uniquement certaines extensions, ce qui allège le modèle lorsque le système ne nécessite pas un niveau de granularité aussi élevé.
À l'aide de cette modélisation, nous proposons des solutions techniques pour répondre aux défis évoqués précédemment, que nous développerons dans la suite de ce mémoire.

2. Contrôle d'accès pour le partage de données multi-propriétaires

Sommaire

2.1. Introduction . . . 56
2.2. Contrôles d'accès dans la littérature . . . 57
2.2.1. Modèles classiques . . . 58
2.2.2. Contrôles d'accès collaboratifs . . . 59
2.3. Défis des contrôles d'accès collaboratifs dans les réseaux sociaux . . . 62
2.3.1. Défis liés aux relations dynamiques . . . 64
2.3.2. Défis liés aux données multi-propriétaires . . . 65
2.3.3. Défis liés au contenu dynamique . . . 66
2.3.4. Discussion . . . 67
2.4. Modèle de contrôle d'accès collaboratif basé sur les attributs et la provenance . . . 69
2.4.1. Spécification des politiques d'accès . . . 70
2.4.1.1. Composants d'une politique . . . 70
2.4.1.2. Définition d'une politique . . . 72
2.4.2. Expression de politiques raffinées . . . 77
2.4.3. Sémantique pour l'évaluation des politiques . . . 81
2.4.4. Évaluation d'accès d'objets composés . . . 85
2.5. Prototype et validation . . . 88
2.5.1. Architecture . . . 88
2.5.2. Générateurs de graphes et de politiques . . . 89
2.5.3. Expérimentations . . . 91
2.5.4. Résultats et discussion . . . 93
2.6. Conclusion et Discussion . . . 98

2.1. Introduction

Le contrôle d'accès est un ensemble de mécanismes conçu pour vérifier et appliquer les droits d'accès aux ressources informatiques. Il utilise des politiques ou des matrices d'accès (LAMPSON 1974) pour déterminer et réguler qui peut voir ou utiliser ces ressources, et comment cela peut être fait. Le but est de garantir que seules les personnes, systèmes ou entités autorisés puissent accéder à des informations spécifiques ou utiliser des fonctionnalités particulières, tout en empêchant les accès non

autorisés. Le contrôle d'accès permet d'assurer deux des propriétés fondamentales de la sécurité : la confidentialité des données et leur intégrité. Les éléments principaux du contrôle d'accès sont l'identification (identifier un utilisateur particulier), l'authentification (vérifier l'identité de cet utilisateur) et l'autorisation (déterminer les ressources et les actions auxquelles l'utilisateur authentifié a le droit d'accéder ou d'exécuter). Des mécanismes de suivis et d'audit sont nécessaires aussi afin de vérifier que tout fonctionne correctement. Dans ce travail, nous abordons le contrôle d'accès comme le mécanisme permettant aux utilisateurs d'exprimer des politiques correspondant à leurs souhaits concernant l'accès à leurs données ou l'impact potentiel sur ces données et l'autorisation des accès vis-à-vis de ces politiques.

Le contrôle d'accès est un domaine d'étude depuis de nombreuses décennies, ayant conduit à de nombreux modèles, correspondant à différents besoins. Cependant, les systèmes collaboratifs centrés sur la communauté, en raison de leur complexité et de leur dynamique, nécessitent de nouvelles spécificités en matière de gestion des accès. Les auteurs de PACI et al. 2018 soulignent les exigences à respecter pour les contrôles d'accès dans ces systèmes. En effet, les mécanismes de contrôle d'accès doivent permettre une administration collaborative, car les données peuvent être détenues par plusieurs propriétaires. Ils doivent également prendre en compte les aspects dynamiques des données et des relations, gérer efficacement les conflits, et être accessibles et compréhensibles par tous les utilisateurs, quel que soit leur milieu ou leur niveau d'expertise.

Notre travail s'inscrit dans ce cadre-là. Nous proposons un modèle de contrôle d'accès pour des systèmes centrés sur la communauté, tels que les réseaux sociaux, permettant de prendre en compte différentes politiques.

Ce chapitre s'appuie sur l'article BERTOLISSI, MARTINEZ ANTON et ZANNONE 2023 et sa version journal en cours de préparation, enrichi de nouvelles explications et compléments pour offrir une vue plus approfondie des concepts abordés, présentant notre modèle de contrôle d'accès. Nous ne traitons pas de l'authentification des utilisateurs, en supposant que ces derniers sont déjà authentifiés. De plus, nous ne considérons pas la sécurisation du modèle contre les attaques telles que le vol d'identité ou autres.

Dans ce chapitre nous présentons tout d'abord les principaux travaux sur le contrôle d'accès, en particulier ceux développés pour les systèmes collaboratifs (Section 2.2), puis nous illustreront en Section 2.3 les principaux défis à relever par les mécanismes des contrôles d'accès dans les réseaux sociaux. Nous présenterons notre modèle répondant à ces défis en Section 2.4 et validons cette approche à l'aide d'un prototype prouvant sa faisabilité (Section 2.5).

2.2. Contrôles d'accès dans la littérature

Comme évoqué précédemment, de nombreux contrôles d'accès existent, répondant à des besoins spécifiques selon le type de système. Les premiers contrôles d'accès logiques, en opposition aux contrôles d'accès physiques, ont vu le jour dans les années

60, majoritairement dans le domaine militaire, pour la protection des données confidentielles et secrètes, mais aussi dans le domaine civil, pour assurer l'intégrité des données. C'est dans ce contexte que des modèles comme Mandatory Access Control (MAC) et Discretionary Access Control (DAC) sont nés. Avec l'enrichissement et la démocratisation des systèmes informatiques dans les entreprises et autres organisations, des modèles basés sur les rôles ont été développés. Malheureusement, ces modèles peuvent être trop rigides et structurés ou manquer de flexibilité et d'expressivité lorsqu'ils sont appliqués aux systèmes collaboratifs nécessitant des contrôles d'accès pour des données multi-propriétaires et utilisables par un grand nombre d'utilisateurs.

Des modèles d'accès collaboratifs ont alors été développés, prenant en compte, par exemple, les relations entre utilisateurs ou les attributs pour augmenter la flexibilité du contrôle d'accès (ASIM et al. 2016). Cependant, des limitations peuvent subsister dans ces modèles comme le montrent les auteurs de PACI et al. 2018.
Dans cette section, nous présenterons les principaux travaux de contrôle d'accès que l'on peut trouver dans la littérature, en particulier ceux développés pour répondre aux problématiques des contrôles d'accès collaboratifs.

2.2.1. Modèles classiques

Certains modèles classiques de contrôle d'accès peuvent fournir un début de solution au contexte collaboratif comme le met en avant TOLONE et al. 2005.

Le contrôle d'accès basé sur les rôles ou Role-Based Access Control (RBAC) (R. SANDHU 1996) attribue des permissions aux utilisateurs en fonction de leurs rôles au sein de l'organisation, facilitant ainsi la gestion des droits d'accès à grande échelle et offrant une plus grande flexibilité par rapport au DAC. Des extensions pour ce modèle ont été proposées, comme ajouter des rôles représentant le contexte (COVINGTON et al. 2001), activés en fonction de l'environnement au moment de la demande.

Le contrôle d'accès basé sur les tâches ou Task-Based Access Control (TaskBAC) (THOMAS et R. S. SANDHU 1998) se concentre sur les tâches spécifiques qu'un utilisateur doit accomplir. Ce modèle permet d'ajuster les permissions en fonction des responsabilités et des actions requises pour chaque tâche, augmentant ainsi la précision du contrôle d'accès dans des environnements dynamiques.

Enfin, le contrôle d'accès basé sur les équipes ou Team-Based Access Control (TeamBAC) (THOMAS 1997) prend en compte la structure organisationnelle en intégrant les équipes ou groupes de travail dans le processus de gestion des permissions, au lieu d'utiliser les individus. Les droits d'accès sont attribués en fonction des équipes auxquelles les utilisateurs appartiennent, facilitant ainsi la collaboration.

Ces modèles offrent des approches variées pour la gestion des accès, chacune étant adaptée à des contextes et des besoins spécifiques, et permettant une meilleure flexibilité et adaptabilité dans la gestion des droits d'accès. Cependant, bien qu'ils présentent certaines qualités pour les systèmes collaboratifs, ils montrent également d'importantes lacunes dans la réponse à certaines problématiques comme le conclut le travail TOLONE et al. 2005. Ces limitations incluent l'incapacité à gérer les change-

ments dynamiques des droits d'accès, les contraintes liées à la prise en compte du contexte global, une collaboration souvent limitée aux tâches spécifiques, ainsi que le manque de possibilités d'administration centralisée ou encore d'utilisation facile et compréhensible par tous.

2.2.2. Contrôles d'accès collaboratifs

Afin de répondre aux besoins des systèmes collaboratifs, de nombreuses approches ont été explorées, comme le montre PACI et al. 2018, où les auteurs comparent les modèles analysés aux exigences qu'ils ont dégagées des besoins spécifiques des systèmes basés sur la communauté. En effet, grâce à leur analyse ils en définissent 7 en lien avec la spécification des politiques, la gouvernance (i.e. la gestion des accès), la facilité d'utilisation et la transparence. Ces exigences peuvent être résumées comme suit.

— **Spécification des politiques :** les modèles doivent être capables de gérer la complexité accrue que la collaboration et les communautés introduisent (*critère 1*), ainsi que de capturer la dynamique et le contexte de ces environnements (*critère 2*).
— **Gouvernance :** les modèles doivent permettre l'administration collaborative des ressources partagées (*critère 3*) et supporter des méthodes de résolution de conflits (*critère 4*).
— **Ergonomie et Facilité d'utilisation :** les systèmes de contrôle d'accès doivent être discrets et ne pas imposer de surcharge supplémentaire aux utilisateurs (*critère 5*), tout en aidant les utilisateurs à inspecter et configurer leurs préférences d'accès (*critère 6*).
— **Transparence :** les systèmes de contrôle d'accès doivent être clairs et permettre aux utilisateurs de comprendre les décisions collaboratives et leurs effets (*critère 7*).

Afin de répondre à ces exigences, des contrôles d'accès basés sur différents éléments comme la confiance et les relations entre les utilisateurs ont été proposés dans la littérature. Ils permettent aussi de prendre en compte la dynamicité des systèmes ou encore diviser les objets en sous-objets afin de garantir un contrôle d'accès précis.

Contrôle d'accès basé sur les relations Le modèle Relationship-Based Access Control (ReBAC) (GATES 2007) permet d'autoriser des requêtes en se fondant sur les relations sociales entre l'utilisateur qui demande l'accès et le propriétaire de la ressource sollicitée. Dans le cadre des réseaux sociaux, le type de relations peut être ami, abonné, etc. Elles sont souvent représentées à travers un graphe de relations où les nœuds représentent les utilisateurs du système et les arêtes les relations qu'ils partagent. Les conditions d'accès sont exprimées sous forme de chemins du graphe de relations. De nombreux modèles se sont basés sur ReBAC pour le généraliser ou le rendre plus flexibles (AKTOUDIANAKIS et al. 2013 ; BRUNS et al. 2012 ; CARMINATI, FERRARI et PEREGO 2009 ; P. W. L. FONG et al. 2009 ; P. W. FONG et al. 2013 ; P. W. FONG 2011 ; SQUICCIARINI et al. 2014 ; CLARK et al. 2022). Cependant, ces modèles ne modélisent que les relations entre les utilisateurs et ne tiennent pas compte des relations

entre les utilisateurs et les données. Ces dernières relations sont cruciales pour retrouver et gérer les données multi-propriétaires. Dans l'article (CRAMPTON et al. 2014), les auteurs proposent de modéliser ce type de relations. Cependant, leur modèle n'utilise qu'un point de vue statique du système et ne permet pas de représenter l'évolution des données.

Contrôle d'accès basé sur la confiance Les modèles de contrôle d'accès basés sur la confiance (Trust-Based Access Control)(AHN et al. 2012; LI, MITCHELL et al. 2002; TRIVELLATO et al. 2014) évaluent la fiabilité des utilisateurs en se fondant sur divers critères tels que l'historique des interactions et des relations. Inspirés par le développement des systèmes de gestion de la confiance et de la réputation, ces modèles adaptent les permissions d'accès en fonction du niveau de confiance attribué, permettant ainsi une gestion dynamique et contextuelle des droits d'accès. Tout comme la plupart des modèles basés sur ReBAC, les relations entre utilisateurs et données ne sont pas modélisées.

Contrôle d'accès basé sur les attributs Le contrôle d'accès basé sur les attributs, ou Attribute-Based Access Control (ABAC) (YUAN et al. 2005), est un modèle de contrôle d'accès offrant plus de granularité, de passage à l'échelle et de flexibilité que d'autres modèles classiques. Il utilise des attributs spécifiques des utilisateurs, des ressources et de l'environnement pour déterminer les permissions, représentées par des politiques d'accès évaluées pour rendre une décision. Ce modèle est particulièrement reconnu grâce à l'architecture eXtensible Access Control Markup Language (XACML), largement utilisée pour implémenter ABAC (OASIS 2013).

Certaines extensions ont été proposées ces dernières années pour répondre aux exigences du contrôle d'accès dans les systèmes collaboratifs. Par exemple, des extensions visent à protéger les données dans les environnements collaboratifs en intégrant des notions de confiance ou de protection de la vie privée (SMARI et al. 2014), et à répondre aux exigences des clouds collaboratifs (MADANI et al. 2023). Pour les réseaux sociaux, SWISSA et al. 2021 introduit des attributs sociaux pour prendre en compte la confiance et les interactions entre les utilisateurs. Dans l'article HSU et al. 2016, les auteurs étendent ABAC pour supporter les attributs de localisation, permettant une gestion plus contextuelle des permissions.

Cependant, la gestion des attributs, surtout lorsqu'ils sont nombreux, peut être complexe, notamment pour des utilisateurs non experts. De plus, l'utilisation des attributs pour modéliser les relations entre utilisateurs n'est pas toujours optimale.

Contrôle d'accès basé sur la provenance Les aspects dynamiques liés à l'évolution des données sont généralement traités par des modèles de contrôle d'accès qui exploitent les informations de provenance pour la prise de décision d'accès. À cet égard, plusieurs travaux ont utilisé lOPM ((PARK et al. 2012; NGUYEN et al. 2013; BERTOLISSI, HARTOG et al. 2019)) dans le contexte du contrôle d'accès. Ils répondent ainsi à des problématiques des systèmes collaboratifs. Cependant, ils ne prennent

pas en compte les relations sociales entre les agents du système, les rendant ainsi pratiques pour les systèmes collaboratifs centrés sur les tâches.

Contrôle d'accès et relation dynamiques Dans les systèmes collaboratifs basés sur les communautés, notamment les réseaux sociaux, les relations entre les utilisateurs peuvent évoluer au fil du temps. Cet aspect dynamique est abordé dans P. W. FONG 2011, où les auteurs introduisent la notion de contexte pour illustrer que les relations interpersonnelles se développent et se maintiennent au sein d'un cadre spécifique. Cette approche est généralisée dans P. W. FONG et al. 2013 avec des opérateurs temporels permettant de spécifier des politiques qui englobent à la fois des éléments basés sur les relations et sur l'historique.

D'autres modèles considèrent les relations transitoires, c'est-à-dire les relations qui peuvent être activées et interrompues au fil du temps (ARORA et al. 2022). Cela permet d'étendre l'applicabilité à des environnements IoT où les relations entre les utilisateurs et les appareils sont transitoires.

Cependant, comme les autres modèles ReBAC, les relations utilisateurs/données ne sont pas modélisées.

Contrôle d'accès et multi-propriétaire Plusieurs modèles de contrôle d'accès récents ont été proposés pour soutenir la spécification et l'évaluation collaboratives des politiques sur les ressources détenues conjointement par plusieurs utilisateurs (BAHRI et al. 2015 ; ILIA, CARMINATI et al. 2017 ; MEHREGAN et al. 2016 ; RAJTMAJER et al. 2016a ; VISHWAMITRA et al. 2017). Ces travaux supposent généralement que tous les utilisateurs liés à un objet ont le même niveau d'autorité sur l'objet. D'autres travaux (H. HU, AHN et JORGENSEN 2013 ; DAMEN, HARTOG et al. 2014 ; MAHMUDLU et al. 2016) proposent un modèle asymétrique où le degré d'autorité d'un utilisateur sur un objet dépend de sa relation avec l'objet. Par exemple, les auteurs de H. HU, AHN et JORGENSEN 2013 identifient quatre principaux contrôleurs pour les données partagées dans les OSN : propriétaire, contributeur, partie prenante et diffuseur. Le travail présenté dans VISHWAMITRA et al. 2017 permet un contrôle d'accès précis pour les photos partagées sur les réseaux sociaux, où les contraintes d'accès sont définies pour les PII associées aux photos partagées plutôt que de se concentrer sur la photo entière, en séparant ainsi la requête d'accès en sous-objets. Cependant, la généralisation à d'autres types de contenu n'est pas triviale.

Cependant, aucun modèle ne répond à toutes les exigences, souvent par manque d'expressivité ou par la difficulté d'étendre leur approche à tout type de contenu (PACI et al. 2018). Dans les réseaux sociaux, il est particulièrement complexe de trouver et considérer tous les utilisateurs impliqués dans un contenu pour évaluer un accès. Cela exige de prendre en compte non seulement les relations entre utilisateurs, mais aussi les divers liens entre les objets et entre les objets et les utilisateurs, comme illustré dans la section suivante.

2.3. Défis des contrôles d'accès collaboratifs dans les réseaux sociaux

Les réseaux sociaux sont un type particulier de système collaboratifs basés sur la communauté dont le but principal est de partager des informations avec son réseau de relations, souvent personnelles rendant les critères mis en avant par les auteurs de PACI et al. 2018 d'autant plus importants. Du point de vue des utilisateurs, lorsqu'ils utilisent un système collaboratif de type réseau social, des préoccupations concernant l'exposition de leur vie privée peuvent émerger.

Dans cette section, nous illustrons, à travers des exemples de scénarios dans un réseau social, ces préoccupations et les lacunes des contrôles d'accès implémentés dans ce type de système par rapport aux exigences présentées précédemment.

Imaginons un réseau social comportant 8 utilisateurs (Alice, Bob, Christopher, Daphné, Edward, Fiona, Finn et George) représenté par le graphe de provenance en Figure 2.1 et dont les utilisateurs partagent les relations présentées en Figure 1.1. Dans ce réseau social, Alice a chargé une photo dans son album $a1$ et a identifié (« taggé ») deux de ses amis, Bob et Christopher. On voit que Bob et Daphné ont partagé cette photo sur leur profil. Daphné a laissé un commentaire sous le post de Bob.

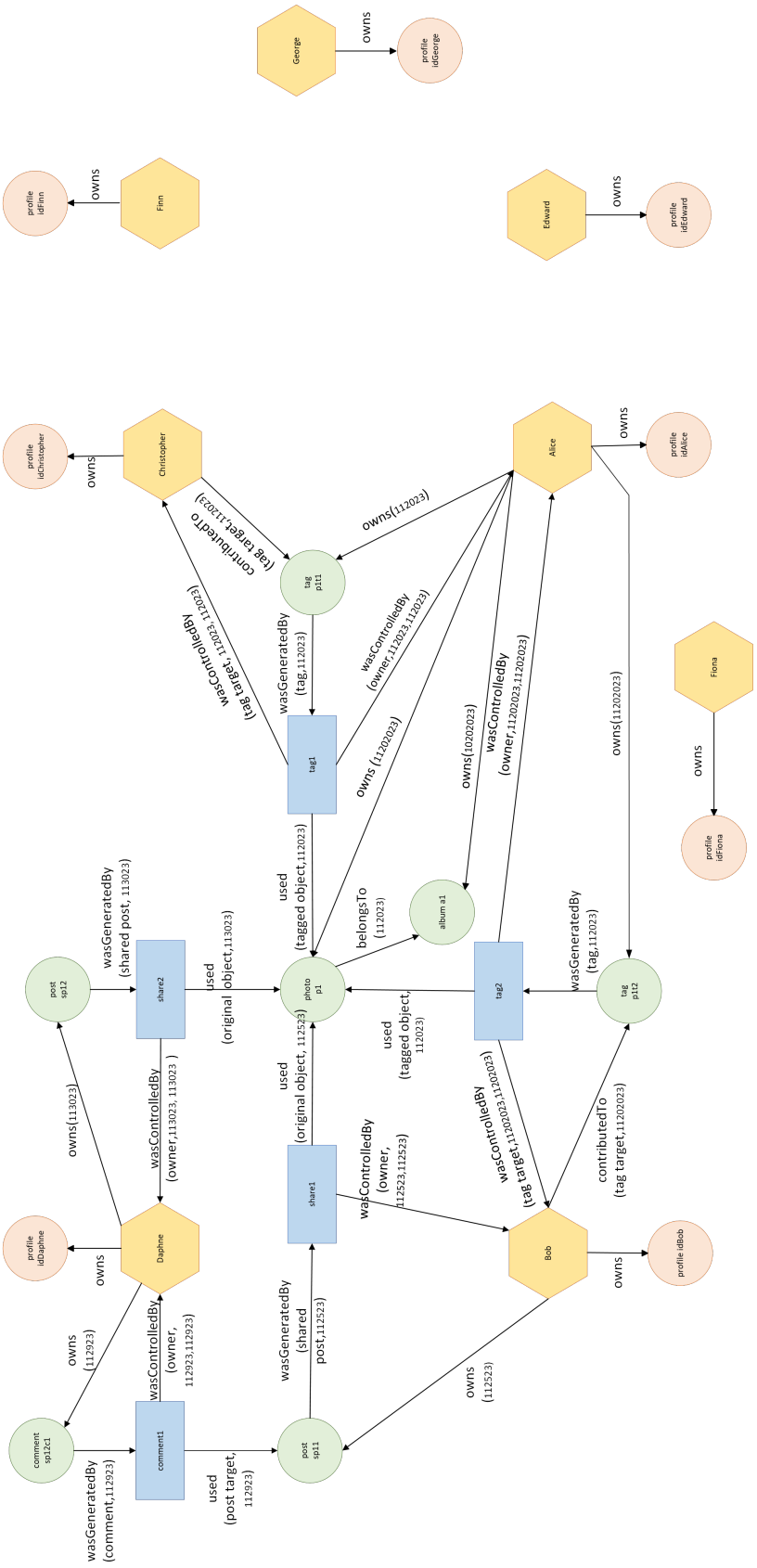


FIGURE 2.1. – Graphe de provenance pour les exemples Sections 2.3 et 2.4

2.3.1. Défis liés aux relations dynamiques

Un premier problème que les utilisateurs peuvent rencontrer est l'absence de possibilité d'utiliser des contraintes de temps dans leurs politiques. Par exemple, les paramètres Facebook ne permettent pas de prendre en compte la date à laquelle une relation a débuté, ni sa durée.

Exemple 2 (Nouvelle amitié) *Alice a récemment rencontré une nouvelle amie, Fiona, lors d'une fête. Elle lui a demandé son amitié sur le réseau social. Comme elles se connaissent depuis peu de temps, Alice ne se sent pas à l'aise de laisser Fiona voir tout le contenu qu'elle a publié sur son profil ou dans ses albums dans le passé. À cette fin, en utilisant les mécanismes de contrôle d'accès actuellement fournis par les systèmes de type réseau social Alice doit exclure manuellement Fiona de chaque contenu publié ou chargé avant d'accepter Fiona dans son groupe d'amis. Il s'agit d'un processus très laborieux et sujet aux erreurs, en particulier lorsque le nombre d'activités avant l'acceptation de la nouvelle amie est important.*

Dans d'autres systèmes comme dans certains services de messagerie instantanée, la visibilité du contenu est accordée à un nouvel utilisateur à partir du moment correspondant à son premier accès à un groupe ou à une conversation. On trouve des mécanismes semblables dans l'édition collaborative où le propriétaire du projet peut choisir la portée de la visibilité à appliquer aux nouveaux utilisateurs (c'est-à-dire depuis la création du projet ou depuis l'entrée de l'utilisateur dans le projet).

Ces limitations dans la gestion des aspects dynamiques des réseaux sociaux peuvent affecter l'intégrité des utilisateurs dans la vraie vie, comme le montre l'exemple suivant.

Exemple 3 (Diffusion d'informations indésirables) *Alice a subi des violences de la part de son ex-mari Edward. Même si Edward a une injonction d'éloignement, Alice craint qu'il puisse obtenir des informations sur sa vie. Ces informations peuvent être divulguées par le contenu qu'elle partage sur ses réseaux sociaux.*

La première étape consiste à bannir Edward de ses amis, mais cela n'est peut-être pas suffisamment restrictif. Les amis qu'ils ont en commun peuvent aussi, involontairement, faire fuiter des informations sensibles à son sujet. Par exemple, un ami d'Edward, qui est aussi ami avec Alice, voit sur le profil d'Alice qu'elle va à un concert samedi. Comme il est également fan du groupe, il poste l'affiche publicitaire du concert sur son profil, mentionnant qu'Alice y assistera également. Edward, en lisant le profil de cet ami, apprend ainsi qu'Alice sera au concert samedi.

Une autre solution serait qu'Alice interdise l'accès à son profil et à son contenu à tout utilisateur qui est ou a été ami avec son ex-mari. Avec les mécanismes de contrôle d'accès actuellement fournis par les réseaux sociaux, Alice ne peut bannir les amis de son ex-mari que manuellement et en modifiant un par un les paramètres d'audience et de visibilité de son profil dans les paramètres de confidentialité. Cependant, cela ne résout pas complètement le problème, car Alice ne connaît peut-être pas tous les anciens amis de son ex-mari, et la charge de travail pour se protéger est immense.

Ces exemples montrent que les mécanismes de contrôle d'accès actuellement fournis par les réseaux sociaux ne sont pas suffisamment expressifs pour prendre en compte l'évolution des relations des utilisateurs au fil du temps. Les exceptions doivent être gérées manuellement, nécessitant des mises à jour régulières et fastidieuses. De plus, les utilisateurs ne pouvant pas consulter les relations passées, il est impossible de satisfaire totalement leurs besoins dans le scénario présenté précédemment. Si l'on regarde par rapport aux exigences présentées précédemment, la spécification des politiques pose donc un problème ainsi que la facilité d'utilisation.

2.3.2. Défis liés aux données multi-propriétaires

Si l'on suppose que la photo $p1$ d'Alice est une photo prise durant une soirée, des préoccupations sur l'exposition de la vie privée peuvent surgir pour les utilisateurs identifiés dessus.

Exemple 4 (Tagging) *Alice règle la visibilité de la photo sur « amis d'amis », c'est-à-dire que les amis d'Alice et les amis de leurs amis peuvent voir la photo. Si Christopher est tagué sur la photo, la visibilité de celle-ci s'étend pour inclure les « amis d'amis » de Christopher (cf. Section 1.2.1.3). Ainsi, la visibilité de la photo devient beaucoup plus large que ce qu'Alice avait initialement prévu.*

Pour restreindre la visibilité de la photo et en reprendre un contrôle partiel, Alice peut modifier ses préférences de confidentialité. Cependant, si elle restreint la visibilité de la photo, par exemple à ses « amis », alors seuls les amis d'Alice et les amis de Christopher pourront voir la photo, ce qui n'inclut pas tous ses amis d'amis. La seule façon pour Alice de restaurer son public initial (c'est-à-dire ses amis d'amis) est de supprimer le tag.

*En général, une solution possible pour Alice afin d'empêcher que les tags modifient la visibilité de ses photos est d'interdire aux utilisateurs de publier sur son profil (*Damen *et* Zannone *2014). Cependant, Alice pourrait trouver cela trop restrictif et cette solution n'est pas intuitive. D'un autre côté, du point de vue de Christopher, il ne peut pas faire grand-chose pour contrôler la visibilité de la photo sur laquelle il est tagué; il peut seulement supprimer le tag pour que la photo n'apparaisse pas sur son profil. Dans ce cas, il perd la possibilité de surveiller facilement ce que devient la photo.*

Cet exemple illustre que les utilisateurs n'ont qu'un contrôle limité sur la visibilité des photos lorsqu'ils utilisent la fonctionnalité de « tag ». Les solutions actuellement disponibles ne sont pas satisfaisantes. En outre, la personne taguée n'a aucun contrôle sur la visibilité de la photo.

Les exigences en matière de gouvernance (comme la possibilité de gérer collectivement les droits) et de spécification des politiques ne sont pas respectées. La transparence est également compromise, car les conséquences de certains choix ne sont pas intuitives. Ainsi, cet exemple suggère qu'il est essentiel de prendre en compte à la fois les préférences de confidentialité associées à la photo et celles liées aux tags lors de l'octroi des autorisations.

2.3.3. Défis liés au contenu dynamique

Une autre fonctionnalité présentant des limitations quant à la protection de la vie privée et à la facilité à comprendre son fonctionnement est l'action *partager*. L'exemple suivant met en avant ces limitations.

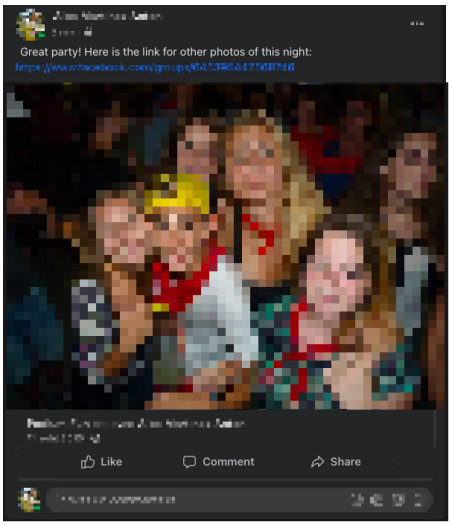
Exemple 5 (Sharing) *Alice change les préférences de sa photo et laisse tout le monde voir la photo (aucune restriction de visibilité). Bob, qui est l'ami d'Alice (cf. Figure 1.1), regarde la photo et décide de la partager sur son profil. En pratique, cela conduit à créer un nouveau post sur son profil avec un lien vers la photo d'Alice, complété par un texte écrit par Bob. Ce texte contient un lien vers un référentiel externe stockant les photos de la fête (par exemple, un lien vers un dossier sur un Cloud partagé). Comme la photo d'Alice a une visibilité publique, l'action* `share` *est disponible dans le post partagé, comme indiqué dans la Figure 2.2a. Daphné, qui n'est pas une amie d'Alice, regarde le post sur le profil de Bob et décide de le re***partager** *sur son profil, ce qui donne lieu à un nouveau post dans son profil. Il convient de noter que la publication sur le profil de Daphné ne contient pas de lien vers la publication de Bob mais vers la photo originale d'Alice (cf. Section 1.2.1.3).*

Supposons maintenant qu'après réflexion, Alice trouve la photo embarrassante pour elle. Elle peut prendre deux mesures possibles pour répondre à ses préoccupations : modifier ses préférences de confidentialité ou supprimer la photo. Les conséquences de ces actions en termes de visibilité des publications partagées de Bob et Daphné sont les suivantes :

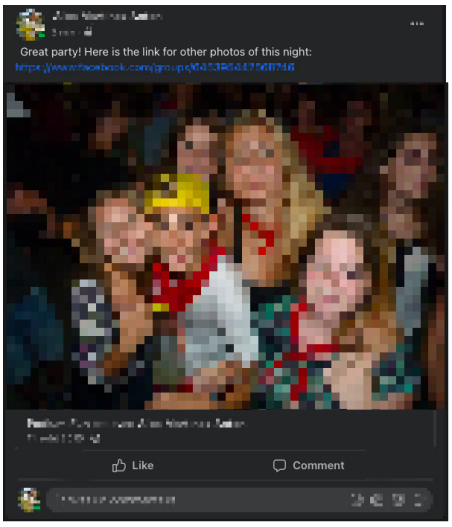
- *Supposons qu'Alice modifie ses préférences de confidentialité pour la photo en* amis. *En raison de ce changement, l'affichage de la photo dans les publications sur les profils de Bob et Daphné ne peut être vu que par les amis d'Alice (Figure 2.2b) ; pour tous les autres utilisateurs, l'affichage de la photo est remplacé par un message d'erreur indiquant que le contenu n'est pas disponible pour eux (Figure 2.2c)* [1]*. Cependant, le texte et les liens fournis avec la photo sont visibles pour tous les utilisateurs qui peuvent voir les posts de Bob et Daphné. Dans les deux cas, l'action de partage n'est plus disponible.*
- *Dans le deuxième scénario, nous supposons qu'Alice décide de supprimer la photo. Dans ce cas, les publications de Bob et Daphné continuent d'exister, mais la photo est remplacée par le même message d'erreur que précédemment. Cependant, contrairement au scénario précédent, l'action de partage est toujours disponible (Figure 2.2d). La publication de Bob est désormais considérée comme une publication « normale », ouvrant la voie à une diffusion plus poussée de l'URL du référentiel et affectant ainsi éventuellement la réputation d'Alice.*

Cet exemple démontre que les modifications des préférences de confidentialité peuvent s'avérer inefficaces pour restreindre l'accès à des informations que l'utilisateur souhaite garder privées. Le texte et les liens accompagnant les photos peuvent

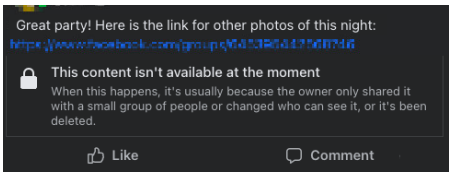
1. Il convient de noter que Daphné ne peut pas non plus `afficher` la publication partagée sur son profil.



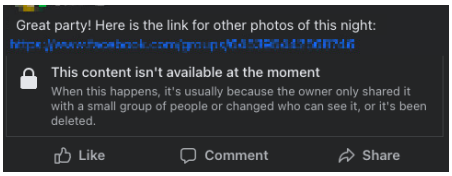
(a) Publication de Bob et Daphné lorsque la visibilité de la photo est définie sur **public**



(b) Publication de Bob et Daphné, telle que vue par les amis d'Alice, lorsque la visibilité de la photo est définie sur les amis d'Alice



(c) Publication de Bob et Daphné, telle que vue par les utilisateurs qui ne sont pas les amis d'Alice, lorsque la visibilité de la photo est définie sur les amis d'Alice



(d) Publication de Bob et Daphné lorsque la photo d'Alice a été supprimée

FIGURE 2.2. – Différents cas pour une publication partagée

contenir des détails sur la photo ou fournir des moyens alternatifs pour la récupérer (par exemple, la photo est également stockée dans un référentiel accessible via le lien fourni par Bob dans cet exemple). Même si l'utilisateur décide de supprimer sa photo, il ou elle perd le contrôle sur les publications partagées.
En termes d'exigences, ce scénario met en évidence des limitations à la fois dans la spécification des politiques de confidentialité et dans la facilité d'utilisation de ces outils.

2.3.4. Discussion

Les réseaux sociaux comme Facebook fournissent à leurs utilisateurs des fonctionnalités pour partager leurs informations avec d'autres utilisateurs et des contrôles pour restreindre la visibilité des informations partagées. Cependant, comme l'illustrent les exemples précédemment introduits, de tels contrôles peuvent être contre-intuitifs

(cf. Exemples 4 et 5) ou inefficaces (cf. Exemples 2 et 3) et entraîner une diffusion incontrôlée d'informations sensibles.

Ces problèmes sont dus aux limitations des mécanismes de contrôle d'accès actuellement utilisés dans les systèmes collaboratifs tels que les réseaux sociaux. Une limitation principale réside dans le fait que le mécanisme de contrôle d'accès sous-jacent repose souvent sur un modèle de gouvernance des données dans lequel les objets appartiennent à un seul utilisateur et néglige la nature complexe des systèmes collaboratifs. Dans ces systèmes collaboratifs, les utilisateurs devraient pouvoir conserver une certaine autorité sur les objets, qui dépend de leur relation avec l'objet (DAMEN, HARTOG et al. 2014). Par exemple, sur Facebook, la visibilité d'un tag dépend du propriétaire de la photo et des préférences de confidentialité associées à la photo ou à l'album contenant la photo. La cible du tag peut supprimer le tag du contenu mais ne peut pas modifier sa visibilité. De plus, ces mécanismes ne peuvent pas gérer les aspects dynamiques du contenu et des relations. Un utilisateur ne peut pas personnaliser les paramètres de visibilité en fonction des contraintes de temps. Par exemple, un utilisateur ne peut pas donner accès à son profil à partir d'un certain moment ou prendre en compte les relations passées pour définir ses préférences de confidentialité. Les exemples montrent que les aspects dynamiques du contenu social peuvent avoir un impact significatif sur leur visibilité et donc sur la confidentialité des utilisateurs : restrictions de visibilité non-appliquées, divulgation d'informations involontaire (Exemple 5). Les Exemples 2 et 3 illustrent comment l'absence de contraintes temporelles dans les préférences de confidentialité des utilisateurs ne permet pas un contrôle précis du partage des données personnelles. Les utilisateurs ne peuvent pas restreindre la visibilité en fonction de leurs relations passées ou des relations passées de leurs amis. De plus, un objet est accessible quelle que soit sa date de publication.

Pour résoudre ces problèmes, les réseaux sociaux doivent fournir un contrôle d'accès précis pour contrôler la visibilité du contenu des utilisateurs, répondant aux exigences mises en avant par PACI et al. 2018. En particulier, le système de contrôle d'accès sous-jacent doit tenir compte de la relation des utilisateurs avec les objets à protéger (par exemple, le propriétaire d'une photo, la cible d'une balise et l'hôte d'une publication) ainsi que des relations entre les objets (par exemple, une photo et ses tags, une publication et ses commentaires). De plus, il doit tenir compte de la nature transitoire des relations entre les utilisateurs.

La section suivante présente notre solution permettant un contrôle d'accès précis, en illustrant son fonctionnement à l'aide des scénarios présentés dans cette section, qui non seulement exploite la relation entre les utilisateurs et les objets, mais prend également en compte l'évolution des objets et la relation entre eux.

2.4. Modèle de contrôle d'accès collaboratif basé sur les attributs et la provenance

Les contrôles d'accès actuellement adoptés dans les systèmes collaboratifs centrés sur les communautés présentent divers problèmes, menant à une mauvaise gestion de la vie privée des utilisateurs, voire à des incidents majeurs comme peuvent être les cas de cyberharcèlement ou d'attaques de clonage ou de désanonymisation ((GAO et al. 2011 ; JAIN et al. 2021)). Le principal problème réside dans l'hypothèse que les objets sont contrôlés par des utilisateurs uniques, alors que plusieurs utilisateurs peuvent voir leur vie privée impactée par un même objet.

Pour répondre aux exigences de ce type de système, nous proposons dans cette section un contrôle d'accès collaboratif permettant de considérer plusieurs contrôleurs, définis de manière similaire à (MAHMUDLU et al. 2016). Cette approche offre une plus grande flexibilité en gardant une trace des interactions entre les utilisateurs et les objets grâce à la provenance des objets (détaillée en Section 1.3) et aux relations entre les utilisateurs (définies en Section 1.4). Pour garantir une plus grande finesse dans les décisions liées à un objet, nous utilisons la notion d'objets composés, où un ensemble de (sous-)objets au sein du système (par exemple, des tags ou des commentaires) n'existe qu'en relation avec un objet principal (par exemple, une photo).

En résumé, le modèle proposé permet d'exprimer des contraintes d'accès basées sur les relations utilisateur-utilisateur, utilisateur-objet et processus-objet. Cela nous permet, par exemple, de prendre en charge le contrôle d'accès basé sur l'historique, où les autorisations qu'un utilisateur peut exercer dépendent des actions précédemment effectuées sur un objet.

Des travaux antérieurs à cette thèse ont utilisé la provenance des objets (appelés artefacts dans l'OPM) pour identifier les autres utilisateurs impliqués et ainsi intégrer leurs préférences de confidentialité dans l'évaluation des politiques. Dans cette section, nous exploitons l'extension de la provenance décrite en Section 1.3, en combinant la notion de propriétaire présentée dans BERTOLISSI, HARTOG et al. 2019 avec les annotations temporelles sur les dépendances causales et relationnelles. Cela nous permet de proposer un contrôle d'accès plus précis, adapté aux défis des systèmes collaboratifs basés sur la communauté, notamment en matière de gestion des contrôles d'accès.

Nous commençons par adapter la spécification des politiques ABAC aux annotations temporelles (Section 2.4.1), en démontrant son expressivité (Section 2.4.2). Ensuite, les Sections 2.4.3 et 2.4.4 présentent la sémantique de l'évaluation des politiques et un algorithme pour réguler l'accès aux objets composés (par exemple, un album et les photos qu'il contient, ou un post et ses commentaires associés). Nous concluons par la présentation de notre prototype et les résultats obtenus, qui valident la faisabilité de ce modèle.

2.4.1. Spécification des politiques d'accès

Nous avons choisi d'utiliser une spécification de politiques proches ABAC et de XACML en raison de leur large adoption dans des implémentations réelles et de la flexibilité qu'ils offrent en tant que modèles.

En suivant le modèle ABAC, nous définissons les politiques et les demandes d'accès en termes de paires d'attribut-valeur. Soit $\mathscr{A}$ un ensemble fini d'attributs, pour chaque attribut $a \in \mathscr{A}$, nous désignons $\mathscr{V}_a$ son domaine, c'est-à-dire que a peut prendre n'importe quelle valeur de $\mathscr{V}_a$.

Une requête q est un ensemble de paires d'attribut-valeur (a_i, v_i) telles que $a_i \in \mathscr{A}$ et $v_i \in \mathscr{V}_{a_i}$, noté $q = \{(a_1, v_1), \ldots, (a_k, v_k)\}$. Nous supposons également un ensemble infini de variables, qui peuvent représenter des nœuds (c'est-à-dire la variable x de domaine $\mathscr{V}_x \subseteq \mathscr{N}$) ou du temps (c'est-à-dire la variable y de domaine $\mathscr{V}_y \subseteq \mathscr{T}$). On note l'ensemble des requêtes $\mathscr{Q}_{\mathscr{A}}$.

Exemple 6 (Requête d'accès) *Dans l'Exemple 5, Bob souhaite* `partager` *la photo* p_1 *d'Alice sur son profil. Cette requête peut être formalisée comme suit :*

$$q = \{(\text{user_id}, \mathit{Bob}), (\text{resource_id}, \mathit{p1}), (\text{destination}, \mathit{profile_idBob}), (\text{action}, \mathit{share}), (\text{current_time}, \mathit{ts}_{\mathit{current}})\}$$

Il convient de noter que l'action `share` *nécessite la spécification de l'utilisateur avec lequel la publication ou la photo est partagée, c'est-à-dire le profil qui recevra le post si l'accès est accordé. La requête comprend un horodatage indiquant l'heure actuelle au moment de la demande. Cela peut être utilisé lors de l'évaluation des politiques pour interpréter les dépendances causales et les contraintes de temps.*

Pour spécifier les politiques, nous adaptons le langage proposé dans Bertolissi, Hartog et al. 2019, qui offre une représentation compacte des politiques XACML (OASIS 2013) avec des conditions sur les chemins, et nous l'étendons pour prendre en charge la représentation temporelle. Les politiques peuvent contenir une décision, des expressions cibles et des conditions de chemins (avec ou sans condition temporelle). Elles peuvent aussi contenir des références à d'autres politiques, représenter des patrons de politiques (« policy template »). Le Tableau 2.1 résume la définition formelle d'une politique $\mathit{pol} \in \mathscr{P}ol_{\mathscr{A}}$.

2.4.1.1. Composants d'une politique

Expression cible ou « target » Une cible (ou « target ») est un élément qui spécifie les conditions sous lesquelles une politique, une règle ou un ensemble de règles s'applique. Il se compose généralement de plusieurs attributs et ensembles d'attributs qui doivent correspondre aux attributs du sujet, de la ressource, de l'action ou de l'environnement pour qu'une politique soit applicable. Ici, nous définissons formellement une expression cible $tgt \in \mathscr{T}arExp_{\mathscr{A}}$ comme :

$$tgt = op(a, v) \mid paths \mid \top \mid \bot \mid \neg tgt \mid tgt \wedge tgt \mid tgt \vee tgt$$

$$
\begin{aligned}
pol &= 1 \mid 0 \mid (tgt, pol) \mid ca(pol_1, \ldots, pol_n) \mid \\
&\quad ca(\{pol_ref(x_1, \ldots, x_k) \mid path\}) \mid ca(\{pol_tem(x_1, \ldots, x_k) \mid path\}) \\
tgt &= op(a, v) \mid paths \mid \top \mid \bot \mid \neg tgt \mid tgt \wedge tgt \mid tgt \vee tgt \\
path &= R^{\circ}(node, node, tstamp) \mid R^{r}(node, node, tstamp) \mid \\
&\quad R^{\circ}(node, node, tstamp, tstamp) \\
&\quad \mid R^{r}(node, node, tstamp, tstamp) \mid \neg path \mid \\
&\quad \mid path \wedge path \mid path \vee path \mid \forall x : path \mid \exists x : path \\
&\quad \mid (path \mid timecons(y_1, ..., y_k)) \\
node &= n \mid x \mid a \mid self \\
tstamp &= ts_i \mid ts_{current} \mid y \mid * \mid \infty \\
timecons &= op(tstamp, tstamp) \mid timecons \wedge timecons \\
&\quad \mid timecons \vee timecons \\
ca &= \mathsf{pov} \mid \mathsf{dov} \mid \mathsf{fa}
\end{aligned}
$$

TABLEAU 2.1. – Définition formelle d'une politique en langage $\mathscr{P}ol_{\mathscr{A}}$

Une expression cible tgt est une composition d'expressions cibles primitives utilisant des opérateurs booléens. Une cible primitive peut être *true* ($\top$), *false* ($\bot$), une expression booléenne $op(a, v)$ où $a \in \mathscr{A}$ est un attribut, $v \in \mathscr{V}_a$ est une valeur dans le domaine de l'attribut a et op est une opération de comparaison ($=, >, <, \neq$), ou une condition de chemin *path*.

Condition de chemin Les conditions de chemins permettent de formaliser des conditions à remplir par rapport au graphe de provenance ou des relations. Formellement une condition de chemin $path \in \mathscr{P}ath_{\mathscr{A}}$ est définie comme :

$$
\begin{aligned}
path \;=\; & R^{\circ}(node, node, tstamp) \mid R^{r}(node, node, tstamp) \\
& \mid R^{\circ}(node, node, tstamp, tstamp) \mid R^{r}(node, node, tstamp, tstamp) \\
& \mid \neg path \mid path \wedge path \mid path \vee path \mid \forall x : path \mid \exists x : path \\
& \mid (path \mid timecons(y_1, ..., y_k))
\end{aligned}
$$

Une condition de chemin *path* peut représenter une dépendance causale dans le graphe de provenance, une dépendance relationnelle dans le graphe des relations, ou une composition de ces dépendances à l'aide d'opérateurs logiques. Ces dépendances peuvent inclure des informations temporelles sous forme d'horodatage ou d'intervalle de temps, selon le type de dépendance, et être soumises à des contraintes temporelles notées $(path \mid timecons(y_1, ..., y_k))$, où les variables de temps $y_1, ..., y_k$ sont considérées comme liées dans *path*. Dans ce cas, il est nécessaire de trouver un chemin satisfaisant la contrainte.

Expression de nœud Une expression de nœud $node \in \mathcal{N}odeExp_{\mathcal{A}}$ est définie comme

$$node = n \mid x \mid a \mid self$$

pouvant être une instance d'un nœud n, une variable x ou un attribut a. Nous utilisons également un attribut spécial *self* pour indiquer l'instance actuelle de l'action demandée.

Lorsqu'un attribut est utilisé comme argument, par exemple d'une condition de chemin, nous supposons qu'il est instancié par la valeur donnée dans la demande d'accès et que cette valeur correspond à un nœud du graphe de provenance. Comme mentionné dans la section 1.3, cela est nécessaire car les attributs concernant les actions utilisées dans les politiques et les demandes d'accès font généralement référence à des « types » d'action plutôt qu'à des instances.

Horodatage Un horodatage $tstamp$ est défini comme

$$tstamp = ts_i \mid ts_{current} \mid y \mid * \mid \infty$$

Cet horodatage peut être un point dans le temps $ts_i \in \mathcal{T}$, le temps courant au moment de la requête $ts_{current}$ fourni par la requête d'accès, une variable y qui sera instanciée durant l'évaluation avec une valeur provenant des graphes, un instant indéfini dans le futur noté ∞, ou une valeur non pertinente notée $*$. Comme pour les nœuds, lorsque on utilise l'attribut comme argument, il sera instancié par la valeur donnée dans la demande d'accès.

Contrainte de temps Les expressions de chemin peuvent contenir des contraintes de temps, $timecons$, pour déterminer la validité d'un chemin. Formellement une contrainte de temps est définie comme :

$$timecons = op(tstamp, tstamp) \mid timecons \wedge timecons \mid timecons \vee timecons$$

Ces contraintes sont des opérations de comparaison booléennes $(=, >, <, \neq)$ ou une composition d'opérations de comparaison booléennes.

2.4.1.2. Définition d'une politique

Une politique est un ensemble de règles qui définissent les conditions sous lesquelles l'accès à une ressource est autorisé ou refusé. Plus formellement, une politique $pol \in \mathcal{P}ol_{\mathcal{A}}$ est définie comme :

$$\begin{aligned} pol \;=\; & 1 \mid 0 \mid (tgt, pol) \mid ca(pol_1, \ldots, pol_n) \mid ca(\{pol_ref(x_1, \ldots, x_k) \mid path\}) \\ & \mid ca(\{pol_tem(x_1, \ldots, x_k) \mid path\}) \end{aligned}$$

Une politique peut être une décision atomique soit *Autoriser* (1) soit *Refuser* (0). Elle peut aussi être conditionnée par une cible (tgt, pol) où la cible tgt définit l'applicabilité de la politique. Par exemple, la politique (resource_type = *photo*, 1) dit qu'elle ne s'applique que si l'objet est de type *photo*, si c'est le cas la politique autorise l'accès.

Une politique peut aussi être une composition de plusieurs politiques combinées à l'aide d'algorithmes de combinaison $ca : ca(pol_1, \ldots, pol_n)$. Ici, nous considérons les algorithmes de combinaison XACML standards : Permit-override (POV) [2] noté pov, Deny-override (DOV) [3] noté dov et First-applicable (FA) noté fa [4] (OASIS 2013).
Une politique peut également être définie en faisant référence à d'autres politiques, auquel cas les politiques référencées seront récupérées. La politique peut aussi être un patron de politique, qui sera instancié par une politique spécifique lors de l'évaluation (voir la suite pour plus de détails).

L'exemple suivant illustre comment, à l'aide de notre formalisme, il est possible de spécifier une politique utilisateur pour un réseau social de type Facebook. La notation $pol_{(x,y)}$ indique que la politique pol est définie par l'entité y sur les objets de type x.

Exemple 7 (Politique utilisateur) *Nous supposons que les utilisateurs peuvent décider de la visibilité de leurs objets (tels que les albums, les photos et les publications) et des tags qui les identifient. Conformément aux paramètres de confidentialité actuels de Facebook, la visibilité est définie en termes de relations interpersonnelles :* **ami**, **ami d'un ami** *et* **public**. *Les préférences de confidentialité peuvent être affinées en ajoutant des exceptions pour des utilisateurs, c'est-à-dire en spécifiant manuellement les utilisateurs qui ne peuvent pas accéder à certains contenus bien qu'ils appartiennent à l'ensemble de relations autorisées. Cette action peut être effectuée si et seulement si les artefacts sont toujours disponibles (ils n'ont pas été supprimés par exemple). Pour simplifier la notation, dans ce qui suit, nous omettons la dépendance* notAvailable(resource_id) *de la définition des politiques.*

Supposons qu'Alice laisse son profil visible à tous les utilisateurs, sauf à Edward (son ex-mari), et autorise uniquement ses amis à y publier, ainsi que ses amis ou les amis de ses amis à partager son contenu. Pour tous les autres utilisateurs, l'action est refusée (représentée par la politique par défaut 0*). La politique d'Alice peut être représentée comme suit :*

$$
\begin{aligned}
&pol_{(profile,Alice)} = (\text{owner} = Alice,\\
&\text{pov}\big(((\text{action} = view \wedge \neg(\text{user_id} = Edward)), 1),\\
&(\text{action} = post \wedge \text{areRelated}(Alice, \text{user_id}, friend, *, \infty), 1),\\
&(\text{action} = share \wedge \text{areRelated}^2(Alice, \text{user_id}, friend, *, \infty), 1), 0\big))
\end{aligned}
$$

Notez que les dépendances areRelated sont décorées avec un intervalle de temps $[*, \infty[$ *pour inclure toutes les amitiés en cours d'Alice. Actuellement, les paramètres de*

2. L'algorithme *permit-overrides* autorise l'accès si au moins une des politiques autorise
3. L'algorithme *deny-overrides* autorise l'accès que si toutes les politiques l'autorisent
4. L'algorithme *first-applicable* ne considère que la première politique applicable.

Facebook ne permettent pas de prendre en compte l'heure de début ou la durée d'une relation.

Alice choisit de rendre ses albums visibles à tous, tandis que ses photos et publications (posts) ne sont visibles que par ses amis. De plus, seuls ses amis peuvent commenter ses photos et publications.

$$
\begin{aligned}
&pol_{(album,Alice)} = (\mathsf{resource_type} = album, \mathsf{owner} = Alice),\\
&\mathsf{pov}\big((\mathsf{action} = view, 1), 0\big)\\
&pol_{(photo,Alice)} = (\mathsf{resource_type} = photo, (\mathsf{owner} = Alice \vee\\
&\mathsf{host} = Alice)), \mathsf{pov}\big((\mathsf{action} = view \vee \mathsf{action} = comment\\
&\wedge \mathsf{areRelated}(Alice, \mathsf{user_id}, friend, *, \infty), 1),\\
&(\mathsf{action} = share, 0), 0\big)\\
&pol_{(post,Alice)} = (\mathsf{resource_type} = post, (\mathsf{owner} = Alice \vee\\
&\mathsf{host} = Alice)), \mathsf{pov}\big((\mathsf{action} = view \vee \mathsf{action} = comment\\
&\wedge \mathsf{areRelated}(Alice, \mathsf{user_id}, friend, *, \infty), 1),\\
&(\mathsf{action} = share, 0), 0\big)
\end{aligned}
$$

En raison des paramètres actuels de Facebook, la fonction de partage est désactivée. En effet, la visibilité d'un objet partagé est décidée par son hôte, ce qui pourrait divulguer des informations à un public initialement non autorisé.

La politique suivante spécifie la visibilité des tags : Alice autorise uniquement ses amis à voir les tags dont elle est la cible (quelle que soit la photo sur laquelle le tag apparaît) :

$$
\begin{aligned}
&pol_{(tag,Alice)} = (\mathsf{resource_type} = tag \wedge \mathsf{resource_target} = Alice),\\
&\mathsf{pov}\big((\mathsf{action} = view \wedge \mathsf{areRelated}(Alice, \mathsf{user_id}, friend, *, \infty), 1), 0\big)
\end{aligned}
$$

Les politiques peuvent être aussi de type référence de politique (*pol_ref*) ou patron de politique (*pol_tem*).

Référence d'une politique Les références de politiques sont utilisées pour récupérer les politiques associées à d'autres objets ou définies par d'autres utilisateurs. Elles sont nécessaires pour déterminer si le demandeur est autorisé à accéder à un objet donné.
Une référence de politique pol_{ref} est une expression qui se résout à l'aide d'une fonction *URI* pointant vers une politique. Nous ne détaillons pas ici le processus de résolution des références de politique, mais nous supposons simplement qu'une référence de politique pol_{ref} génère une fonction $uri : \mathcal{N}^k \rightarrow \mathscr{P}ol_{\mathscr{A}}$, où $\mathcal{N}$ est l'ensemble de nœuds du graphe de provenance. L'Exemple 8 donne une illustration de comment les références de politiques peuvent être utilisées pour trouver des contrôleurs d'un objet. En effet, l'évaluation d'une demande d'accès peut nécessiter l'évaluation des politiques de certains utilisateurs, en plus de son propriétaire. Les politiques pertinentes peuvent être récupérées via des références de politique. Intuitivement, une référence de politique peut être résolue avec une fonction codant un Uniform Resource Identi-

fier (URI) qui identifie les politiques applicables aux nœuds du graphe d'évaluation qui satisfont la requête donnée.

Notez que $uri(x_1,\ldots,x_n)$ est considéré comme liant les variables $x_1,\ldots,x_n$ dans *path*. Étant donné que le nombre de politiques référencées n'est pas connu a priori nous combinons toujours ces politiques à l'aide d'un algorithme de combinaison *ca*.

Exemple 8 (Référence de politique) *Considérons l'Exemple 4 impliquant des photos et des tags. Les préférences de confidentialité de l'utilisateur tagué peuvent être prises en compte à l'aide de la référence de politique suivante, qui récupère les politiques applicables aux tags d'une photo (rappelons que les horodatages $*$ indiquent que les informations temporelles ne sont pas pertinentes) :*

$$\{uri(x,y) \mid \mathsf{wasDerivedFrom}^{\circ}(x,\mathsf{resource_id},*) \wedge \mathsf{contributedTo}(y,x,\mathsf{tag\ target},*)\}$$

Supposons qu'un utilisateur demande à `voir` *la photo p*1 *d'Alice où Christopher et Bob sont tagués. Selon le graphe d'évaluation de la Figure 2.1, la référence de politique se résout en*

$$\{uri(p1t1, Christopher), uri(p1t2, Bob)\}$$

Ces références permettent de trouver les politiques $pol_{(tag,Christopher)}$ et $pol_{(tag,Bob)}$, c'est-à-dire les politiques de Christopher et Bob applicable aux objets de type tag.

Patron d'une politique Les patrons de politiques sont des politiques paramétrées qui sont instanciées au moment de l'évaluation. Un patron de politique *pol_tem* est une expression qui se résout en une politique basée sur une condition de chemin. Comme pour $uri(x_1,\ldots,x_n)$, $pol_tem(x_1,\ldots,x_n)$ sont considérés comme liant les variables $x_1,\ldots,x_n$ dans *path*. Étant donné que le nombre de politiques instanciées n'est pas non plus connu a priori, nous combinons aussi ces politiques à l'aide d'un algorithme de combinaison *ca*. Les patrons de politiques permettent d'exprimer des extraits de politiques générales qui seront instanciées au moment de l'évaluation d'une requête. Les patrons de politiques peuvent être vus comme des extraits de politiques qui permettent d'imposer, dans la politique d'un utilisateur, des contraintes d'accès définies par le système. Ces contraintes ne sont pas basées sur les choix de l'utilisateur, mais concernent d'autres utilisateurs liés par un chemin spécifique au propriétaire de la politique ou à l'objet ciblé par la requête. Ces utilisateurs seront également instanciés lors de l'évaluation de la politique. Par exemple, dans le fonctionnement actuel de Facebook, la visibilité d'une photo est automatiquement élargie lors que des personnes sont identifiées. L'Exemple 9 illustre comment las patrons fonctionne en l'appliquant à l'élargissement de la visibilité.

Exemple 9 (Patron de politique) *Dans un réseau social comme Facebook, la visibilité d'une photo est automatiquement modifiée lorsque des personnes sont identifiées (voir Section 1.2.1.3 ainsi que l'Exemple 4). Le patron de politique nous permet d'écrire cela sous la forme d'une politique avec des variables qui sont instanciées à l'aide d'une*

condition de chemin, comme montré dans l'Exemple 11, où une politique concernant la visibilité des photos est détaillée.

*Considérons à nouveau la photo p*1 *de la Figure 2.1, appartenant à Alice. Le patron de politique pour spécifier une relation d'amitié entre les utilisateurs tagués sur une photo et l'utilisateur demandant l'accès est défini comme suit :*

$$pol_tem_{TT}(y) = \{(\mathsf{areRelated}(y, \mathsf{user_id}, friend, *, \infty), 1) \mid path_target\}$$

où la condition de chemin path_target est définie comme dans l'Exemple 8, path_target = $\mathsf{wasDerivedFrom}^{\circ}(x, \mathsf{resource_id}, *) \wedge \mathsf{contributedTo}(y, x, \mathsf{tag\ target}, *)$ *Cette condition de chemin est satisfaite par un utilisateur y qui est impliqué dans un tag x comme cible du tag et le tag x est associée à la photo demandée* resource_id *suivant une dépendance* wasDerivedFrom *(cf. Tableau 1.1).*
L'amitié est codée à l'aide de la dépendance sociale areRelated *où* user_id *représente l'utilisateur demandant l'accès et y représente tous les utilisateurs tagués dans la photo récupérée via la condition de chemin* $path_target$.

Si l'on applique au scénario de la Figure 2.1 et que Daphné demande à $voir$ *la photo p*1 *d'Alice, où Christopher et Bob sont tagués, le patron de politique est évalué à* 1 *pour* $\mathsf{user_id} = Daphné$ *et* $y = Bob$. *Christopher et Daphné n'étant pas amis, le patron de politique ne sera pas évalué à* 1 *dans cas.*

Dans les exemples suivants, nous montrons comment notre modèle de contrôle d'accès basé sur la provenance peut capturer les politiques actuellement employées par les réseaux sociaux. Nous montrons comment ces politiques peuvent être révisées pour répondre aux problèmes identifiés dans la section 2.3 dans la Section 2.4.2.

Notre formalisation des politiques permet de modéliser les contraintes d'accès implémentées sur des réseaux sociaux existants, comme Facebook. Les deux exemples suivants illustrent en particulier comment formaliser les politiques d'accès pour différentes actions sur les publications et les photos sur Facebook.

Exemple 10 (Contraintes d'accès des publications sur Facebook) *Sur Facebook, la visibilité des publications est déterminée par leur hôte (hébergeur). De plus, un utilisateur est autorisé à* **partager** *une publication si la publication est publique et que l'utilisateur est autorisé à publier sur le profil de l'utilisateur avec lequel la publication est partagée. Dans notre modèle, nous pouvons représenter ces contraintes d'accès, spécifiées par le réseau social, comme suit :*

$$\begin{aligned} &pol_{(post,SN)} = (\mathsf{resource_type} = post),\\ &\mathsf{pov}\Big((\mathsf{action} = view, pol_{(post,\mathsf{host})}),\\ &(\mathsf{action} = share, \mathsf{dov}(pol_{(post,\mathsf{host})}, pol_{(profile,\mathsf{destination})})), 0\Big) \end{aligned}$$

Notez que cette formalisation est déjà plus générale que celle sur Facebook où il est possible de partager un objet seulement si sa visibilité est publique, alors qu'ici on n'impose pas de restrictions a priori sur la visibilité de l'objet. Selon la politique ci-dessus, un utilisateur peut voir un post si la politique de l'hôte ($pol_{(post,\mathsf{host})}$) l'autorise, et il peut partager un post si à la fois la politique de l'hôte et la politique associée au profil de l'utilisateur recevant le partage ($pol_{(profile,\mathsf{destination})}$) l'autorisent.

Exemple 11 (Contraintes d'accès des photos sur Facebook) *La visibilité d'une photo sur Facebook dépend des préférences de l'hôte ainsi que des préférences des personnes taguées sur la photo.*[5] *Comme expliqué dans les Sections 1.2.1.3 et 2.3, lorsqu'un tag est associé à une photo, la politique de confidentialité définie par l'hôte de cette photo s'applique également aux utilisateurs tagués. Dans notre modèle, cela peut être modélisé au niveau des préférences de l'hôte. Considérons la photo p*1 *dans la Figure 2.1 et les préférences d'Alice pour ses photos telles que définies dans l'Exemple 7, puis le modèle actuel utilisé sur Facebook peut être spécifié comme suit :*

$$
\begin{aligned}
&pol_{(photo,Alice)} = (\mathsf{owner} = Alice \vee \mathsf{host} = Alice),\\
&\quad \mathsf{pov}\Big(\big(\mathsf{action} = view, \mathsf{pov}\big(\\
&\qquad (\mathsf{areRelated}(Alice, \mathsf{user_id}, friend, *, \infty), 1),\\
&\qquad \mathsf{pov}(\{pol_tem_{TT}\})\big)\big), 0\Big)
\end{aligned}
$$

Cette politique étend $pol_{(photo,Alice)}$ pour l'autorisation $view$ définie dans Exemple 7 en prenant en compte les utilisateurs tagués, conformément au patron de politique pol_tem_{TT} présenté dans l'Exemple 9 .

*Si nous appliquons cette politique à la photo p*1*, l'évaluation du chemin renvoie deux utilisateurs tagués, Bob et Christopher. On vérifie ensuite la relation entre chacun d'eux et l'utilisateur demandeur, c'est-à-dire si Bob ou Christopher est un* ami *de l'utilisateur demandeur, et on prend l'union de l'ensemble des amis d'Alice, Bob et Christopher (en utilisant l'algorithme de combinaison* pov*). En résumé, en utilisant cette définition, tout utilisateur qui est un* ami *d'Alice, Bob ou Christopher a une visibilité sur la photo, ce qui est la configuration actuellement appliquée sur Facebook.*

2.4.2. Expression de politiques raffinées

Comme montré ci-dessus, nous parvenons à modéliser des contrôles d'accès existants dans des réseaux sociaux. Cependant l'objectif de notre modèle est de pouvoir exprimer des politiques plus fines et précises afin de mieux répondre aux besoins spécifiques des utilisateurs dans ces systèmes. Dans cette section nous montrons comment notre approche permet cette flexibilité dans la spécification des politiques en prenant en compte les préférences de tous les utilisateurs impliqués. Pour commencer, le problème soulevé par l'Exemple 4 en Section 2.3 peut être résolu en partie à l'aide

5. Rappelons que sur Facebook un utilisateur ne peut publier une photo que sur son profil, c'est-à-dire que l'hôte et le propriétaire de la photo coïncident.

de la formalisation donnée un peu plus haut. En reprenant la politique présentée dans l'Exemple 10, nous pouvons modifier la façon de gérer l'action *view*.

Exemple 12 (Contraintes de visibilité raffinées) *On peut affiner la politique du réseau social (SN) proposée dans Exemple 10 pour l'action view sur un post comme suit (l'action* `share` *est traitée dans Exemple 14) :*

$$pol_{(post,SN)} = (\text{resource_type} = post),$$
$$\text{pov}\big((\text{action} = view, \text{dov}(pol_{(post,\text{host})}, pol_{(post,\text{owner})})), 0\big)$$

La politique ci-dessus combine deux politiques, la politique de l'hôte du post (comme dans Exemple 10) et la politique du propriétaire du post, c'est-à-dire de l'utilisateur qui l'a posté. [6] *Ces politiques sont combinées à l'aide de l'opérateur* dov*, ce qui signifie que la demande de visualisation du post n'est accordée que si le propriétaire du post et l'hôte lui en accordent l'accès. Cette politique est donc plus stricte que celle définie dans Exemple 10 en prenant en compte tous les utilisateurs impliqués dans le post.*

Dans l'Exemple 11, les utilisateurs tagués sont pris en compte lors de l'évaluation de la visibilité d'une photo, mais uniquement pour étendre cette visibilité à un ensemble plus large d'utilisateurs. Ceci peut avoir des conséquences indésirables pour le propriétaire de la photo, comme expliqué dans la section 2.3. Notre modèle nous permet de généraliser cette politique et de prendre en compte les préférences de confidentialité des utilisateurs tagués concernant la visibilité de leur tag et les politiques associées à la photo à laquelle appartient le tag. Cela peut être spécifié comme suit :

$$pol_{(photo,SN)} = (\text{resource_type} = photo),$$
$$\text{pov}\Big(\big(\text{action} = view, \text{dov}\big(pol_{(photo,\text{propriétaire})}, pol_{(photo,\text{hôte})},$$
$$\text{dov}\big(\{uri(x,y) \mid \text{wasDerivedFrom}^{\circ}(x, \text{resource_id}, *)$$
$$\wedge \text{contributedTo}(y, x, \text{tag target}, *)\}\big)\big), 0\Big)$$

La politique ci-dessus montre comment la visibilité d'une photo peut être déterminée en combinant, à l'aide de l'opérateur dov*, les préférences du propriétaire et de l'hôte de la photo avec les préférences des utilisateurs (y) apparaissant dans les tags (x) associées à la photo à tout moment dans le passé ($*$). Comme les utilisateurs tagués ne sont pas connus a priori et peuvent changer au fil du temps, la politique récupère leurs politiques au moment de l'évaluation en utilisant des références de politique. Il convient de noter que, dans ce cas, les préférences des utilisateurs tagués n'ont pas besoin d'être codées dans les préférences de l'hôte, mais nous pouvons définir une politique générale à déployer par le réseau social. Cette politique peut être instanciée au moment de l'évaluation en récupérant les préférences des utilisateurs impliqués dans la photo pour laquelle l'accès est demandé, et ces politiques sont évaluées en prenant en compte la relation que chaque utilisateur entretient avec l'objet.*

6. Les deux politiques peuvent contenir des expressions de chemin avec des propriétés temporelles comme indiqué dans Exemple 13.

Afin de traiter l'impossibilité d'utiliser le début d'une relation ou les relations passées comme contrainte d'accès et les problèmes engendrés comme illustré dans les Exemples 2 et 3, nous illustrons dans l'exemple suivant comment notre langage de spécification des politiques permet de remédier à ces lacunes.

Exemple 13 (Politique utilisateur affinée) *Rappelons l'Exemple 3 où Alice ne veut pas divulguer les informations personnelles publiées sur son profil à son ex-mari ou à ses (anciens) amis. Dans notre modèle, les relations passées ou actuelles d'un utilisateur peuvent être capturées à l'aide de la dépendance areRelated, où les heures de début et de fin ne sont pas spécifiées.L'expression de chemin*

$$\mathsf{areRelated}(\mathsf{user_id}, Edward, friend, *, *)$$

est utilisée pour récupérer dans le graphe d'évaluation tous les utilisateurs qui, à tout moment, ont été amis d'Edward.

Pour prendre cela en compte, l'action d'affichage dans la politique de profil de Exemple 7 est étendue comme suit :

$$\begin{aligned} &pol_{(profile,Alice)} = (\mathsf{resource_type} = profile \wedge \mathsf{owner} = Alice),\\ &\mathsf{pov}\Big(\big((\mathsf{action} = view \wedge \neg(\mathsf{user_id} = Edward \wedge\\ &\mathsf{areRelated}(\mathsf{user_id}, Edward, friend, *, *))\big), 1\Big), 0\Big) \end{aligned}$$

Notez que si Alice publie un contenu sur le profil d'un autre utilisateur, les restrictions spécifiées dans la politique ci-dessus ne s'appliquent pas. Pour inclure ces cas, la même condition de chemin doit être intégrée dans la définition des politiques ciblant le contenu correspondant.

Considérons maintenant les préférences de politique d'Alice concernant ses publications. Dans l'Exemple 7, Alice choisit de laisser ses amis voir ses publications. Supposons maintenant qu'elle souhaite restreindre la visibilité aux utilisateurs qui étaient ses amis au moment de la création de la publication. Cela signifie que ses amis n'ont pas accès aux publications créées par Alice avant l'établissement de la relation d'amitié, mais ils ne peuvent voir que les publications créées après cette data. Cela peut être exprimé en utilisant la condition de chemin suivante :

$$\begin{aligned} &\mathsf{areRelated}(Alice, \mathsf{user_id}, friend, ts_b, \infty)\\ &\wedge \mathsf{wasGeneratedBy}(\mathsf{resource_id}, p, post, ts_g) \wedge (ts_b <= ts_g) \end{aligned}$$

où ts_b indique l'heure à laquelle la relation entre amis commence et ts_g indique l'heure de création de la ressource (par exemple un message) dans le système. L'horodatage ∞ indique que la relation entre Alice et user_id *est toujours en cours (sinon, si la relation est terminée, l'expression de chemin ne trouvera aucune correspondance dans le graphe).*

La politique d'Alice pour la publication de Exemple 7 est étendue comme suit :

$$
\begin{aligned}
&pol_{(post,Alice)} = \big(\text{resource_type} = post, (\text{propriétaire} = Alice \vee \\
&\text{hôte} = Alice)\big), \text{pov}\Big(\big((\text{action} = view \vee \text{action} = comment) \\
&\wedge \text{areRelated}(Alice, \text{user_id}, friend, ts_b, \infty) \\
&\wedge \text{wasGeneratedBy}(\text{resource_id}, p, post, ts_g) \wedge (ts_g => ts_b), 1\big), \\
&\big(\text{action} = share, 0\big), 0\Big)
\end{aligned}
$$

De cette façon, seuls les amis actuels, avec lesquels la relation a été établie avant la création de la publication, sont autorisés à voir et à commenter la publication demandée. Cependant, si Alice publie une photo ou un message dans un autre profil utilisateur, elle ne sera pas protégée par cette politique. Dans ce cas, la politique relative aux photos doit être étendue, tout comme la politique de profil.

Avec la possibilité d'inclure des préférences sur les relations passées et présentes, nos politiques peuvent résoudre des scénarios plus génériques, par exemple, en excluant de la visibilité tous les utilisateurs bloqués des amis du propriétaire ou en incluant des conditions sur l'ancienneté ou la nouveauté d'une relation pour accéder à un objet. Cette syntaxe de politiques permet aussi une meilleure précision quant à les gestions des accès liés à l'action *partager* (« share »), comme le montre l'exemple suivant.

Exemple 14 (Partage affiné) *Nous pouvons affiner la politique proposée dans l'Exemple 10 pour l'action* `share` *sur une publication afin de résoudre les problèmes liés au partage mis en évidence dans l'Exemple 5 comme suit :*

$$
\begin{aligned}
&pol_{(post,SN)} = (\text{resource_type} = post \wedge \text{action} = share), \\
&\quad \text{dov}\big(pol_A, pol_B\big) \\
&pol_A = \text{dov}\big(pol_{(post,\text{owner})}, pol_{(post,\text{host})}, pol_{(profile,\text{destination})}\big) \\
&pol_B = \text{dov}\big(pol_{oowner}, pol_{ohost}\big) \\
&pol_{oowner} = \text{dov}\big(\{uri(x,y) \mid \text{wasDerivedFrom}(self, x, \text{original post}, *) \\
&\quad \wedge \text{owns}(y, x, *)\}\big) \\
&pol_{ohost} = \text{dov}\big(\{uri(x,y) \mid \text{wasDerivedFrom}(self, x, \text{original post}, *) \\
&\quad \wedge \text{belongsTo}(x, z, *) \wedge \text{owns}(y, z, *)\}\big)
\end{aligned}
$$

La politique ci-dessus combine deux politiques, pol_A et pol_B, en utilisant l'algorithme de combinaison dov. *De la même manière que la politique de l'Exemple 10, pol_A gère les autorisations directement associées à la publication demandée en combinant (en utilisant à nouveau* dov *comme algorithme de combinaison) les politiques de l'hôte de la publication, de l'utilisateur dont le profil est la destination de l'action* `share`*, plus la politique du propriétaire de la publication. D'autre part, pol_B gère les autorisations associées à la publication dans le cas où la publication est le résultat d'un partage précédent. En particulier, cette politique combine les politiques de l'hôte (pol_{ohost}) et du propriétaire (pol_{oowner}) de la publication d'origine. Les politiques pol_{host} et pol_{owner} sont spécifiées à l'aide d'une référence de politique qui permet de récupérer*

les politiques des publications, définies par l'hôte (respectivement propriétaire) de la publication d'origine. Cette référence est définie par une condition de chemin sur le graphe de provenance prenant comme arguments la publication d'origine x et son hôte (respectivement propriétaire) y. Il convient de noter que l'heure à laquelle la publication a été créée n'est pas pertinente pour cette politique ; par conséquent, les horodatages ont pour valeur $*$.

La politique présentée dans cet exemple répond aux problèmes liés au partage évoqués dans Exemple 5 de la Section 2.3. En effet, contrairement à la configuration actuelle sur Facebook, la politique pol_{oowner} permet de prendre en compte d'éventuelles modifications des préférences du propriétaire du post original dans tout partage ultérieur de ce post. En se référant au scénario de l'Exemple 5, cela signifie que si Alice modifie ses préférences de confidentialité associées à ses photos, cela aura un impact sur la publication de Bob, qui est un partage de la photo d'Alice.

2.4.3. Sémantique pour l'évaluation des politiques

Lorsqu'un utilisateur demande l'accès à un objet, la requête d'accès correspondante est évaluée par rapport aux politiques pertinentes pour déterminer si l'accès doit être accordé ou non. Pour l'évaluation des politiques, nous nous appuyons sur la sémantique présentée dans BERTOLISSI, HARTOG et al. 2019 étendue pour inclure les contraintes temporelles.

L'évaluation des politiques dépend de la requête d'accès entrante. On note cette requête $q \in \mathcal{Q}_{\mathcal{A}}$. Dans cette section nous détaillons la sémantique des différents éléments que peut contenir une politique.

Horodatages et contraintes de temps Rappelons que l'horodatage $ts_{current}$ est une valeur temporelle dans $\mathcal{T}$ instanciée avec l'heure actuelle au moment de l'évaluation de la requête. La variable temporelle y est instanciée avec une valeur temporelle dans $\mathcal{T}$ à partir du graphe de provenance. De plus, dans le but de simplifier les notations, les mêmes symboles sont utilisés pour dénoter un point dans le temps et sa sémantique pour l'évaluation.
Ainsi, le symbole ts_i désigne à la fois le moment précis et son interprétation sémantique. Pour toute contrainte entre deux points dans le temps $ts_i, ts_j \in \mathcal{T}$, la table de vérité de cette contrainte est donnée dans le Tableau 2.2. Le symbole $*$ représente un moment indéterminé et non pertinent. Sa sémantique inclut ainsi n'importe quel point dans l'ensemble $\mathcal{T}$. Aucune restriction n'a besoin d'être appliqué lors de l'utilisation de l'horodatage $*$, il n'est donc pas utilisé dans les contraintes de temps. Le symbole ∞ représente un moment indéterminé dans le futur, c'est à dire après le temps courant $ts_{current}$. Ce symbole peut apparaître dans les contraintes de temps, par exemple pour vérifier qu'une relation existe encore. Pour l'évaluation dans une contrainte de temps, il sera remplacé par $ts_{current}$ et l'évaluation se fera en utilisant la table de vérité : Tableau 2.2.

Comparaison temporelle et **Valeur de vérité**	
$ts_i < ts_j$	Vrai if $i < j$
$ts_i > ts_j$	Vrai if $i > j$
$ts_i = ts_j$	Vrai if $i = j$
$ts_i \neq ts_j$	Vrai if $i \neq j$

TABLEAU 2.2. – Table de vérité des contraintes de temps

Les contraintes de temps sont ainsi évaluées à l'aide de la fonction d'évaluation : $[[\cdot]]_{timecons} : \mathcal{T}imecons \rightarrow \{Vrai, Faux\}$ définie comme suit :

$$[[op(ts_i, ts_j)]]_{timecons}(q) = \begin{cases} Vrai & \text{si op}(ts_i, ts_j) = True \\ Faux & sinon \end{cases}$$

$$[[op(ts_i, \infty)]]_{timecons}(q) = \begin{cases} Vrai & \text{si op}(ts_i, ts_{current}) = True \\ Faux & sinon \end{cases}$$

$$[[op(\infty, ts_i)]]_{timecons}(q) = \begin{cases} Vrai & \text{si op}(ts_{current}, ts_i) = True \\ Faux & sinon \end{cases}$$

$$[[timecons \wedge timecons]]_{timecons}(q) = [[timecons]]_{timecons}(q) \wedge [[timecons]]_{timecons}(q)$$

$$[[timecons \vee timecons]]_{timecons}(q) = [[timecons]]_{timecons}(q) \vee [[timecons]]_{timecons}(q)$$

L'opération $op(ts_i, ts_j)$ est évaluée en suivant la table de vérité du Tableau 2.2.

Expressions de nœuds Étant donnée la requête q, les expressions de nœuds sont évaluées en ensembles de nœuds du graphe d'évaluation à l'aide de la fonction $[[\cdot]]_N : \mathcal{N}odeExp_{\mathcal{A}} \rightarrow \wp(\mathcal{N})$ définie comme suit :

$$[[n]]_N(q) = \{n\}$$
$$[[x]]_N(q) = \emptyset$$
$$[[a]]_N(q) = \{n \in V_a \mid (a, n) \in q\}$$
$$[[self]]_N(q) = \{p_{self}\}$$

Un nœud n est évalué en l'ensemble singleton qui le contient. Comme pour les horodatages, une variable de nœud libre ne peut correspondre à aucun nœud dans le graphe. Dans ce cas, le noeud est évalué à l'ensemble vide. Lorsqu'une variable x est présente dans une politique, elle doit donc être liée et instancié à l'aide des chemins dans les graphes. Les attributs sont instanciés avec leur valeur correspondante dans la requête q. L'expression de nœud *self* fait référence au nœud de processus ajouté si la requête est validée.

Condition de chemin Les conditions de chemin sont évaluées en booléens à l'aide de la fonction d'évaluation $[[\cdot]]_C : \mathscr{P}ath_{\mathscr{A}} \rightarrow \{Vrai, Faux\}$ définie comme suit :

$$[[R^{\circ}(node, node', ts)]]_C(q) = \begin{cases} Vrai & \text{si } \exists n \in [[node]]_N(q), \\ & n' \in [[node']]_N(q), \\ & \mathbf{R}^{\circ}(n, n', ts) \\ Faux & \text{autrement} \end{cases}$$

$$[[R^{r}(node, node', ts)]]_C(q) = \begin{cases} Vrai & \text{si } \exists n \in [[node]]_N(q), \\ & n' \in [[node']]_N(q), \\ & \mathbf{R}^{r}(n, n', ts) \\ Faux & \text{autrement} \end{cases}$$

$$[[\neg path]]_C(q) = \neg [[path]]_C(q)$$

$$[[path \wedge path']]_C(q) = [[path]]_C(q) \wedge [[path']]_C(q)$$

$$[[path \vee path']]_C(q) = [[path]]_C(q) \vee [[path']]_C(q)$$

$$[[\exists x : path]]_C(q) = \exists n \in V_x : [[path[n/x]]]_C(q)$$

$$[[\forall x : path]]_C(q) = \forall n \in V_x : [[path[n/x]]]_C(q)$$

$$[[path \mid timecons(y_1, ..., y_k)]]_C(q) = \exists ts_1, ..., ts_n \in \mathscr{T} : [[path[ts_1/y_1, ..., ts_k/y_k]]]_C(q) \wedge [[timecons(ts_1, ...ts_k)]]_{timecons}$$

Dans les deux premiers cas, l'évaluation dépend d'un horodatage. Si $ts = *$, $\mathbf{R}^{\circ}(n, n', ts)$ ou $\mathbf{R}^{r}(n, n', ts)$ s'évalueront à vrai s'il existe un $ts_i \in \mathscr{T}$ tel que la dépendance $\mathbf{R}^{\circ}(n, n', ts_i)$ (respectivement $\mathbf{R}^{\circ}(n, n', ts_i)$) existe dans les graphes. Si $ts = \infty$, alors la dépendance doit exister avec l'horodatage ∞. Pour les variables y, il est nécessaire que la variable soit liée, c'est-à-dire que la dépendance soit associée à une contrainte temporelle pour être évaluée.

Lorsque la dépendance contient deux horodatages, la sémantique d'évaluation de

$$[[R^{r}(node, node', ts, ts')]]_C(q) \text{ et } [[R^{\circ}(node, node', ts, ts')]]_C(q))$$

est similaire à

$$[[R^{r}(node, node', ts)]]_C(q) (\text{resp.} [[R^{\circ}(node, node', ts)]]_C(q)).$$

La notation $[n/x]$ désigne la substitution de n aux occurrences libres de x. Cela signifie que $[[\exists x : path]]_C(q)$ et $[[\forall x : path]]_C(q)$ sont évalués à vrai s'il existe une instanciation (respectivement pour toute instanciation) de x dans les graphes où le chemin est évalué à vrai. De même, $[ts_i/y]$ désigne la substitution de la variable y par une valeur temporelle $ts_i \in \mathscr{T}$. En d'autres termes, $[[path \mid timecons(y_1, ..., y_k)]]_C(q)$ est évalué à vrai s'il existe un chemin instanciant les variables $y_1, ..., y_k$ à $ts_1, ...ts_k$ de

manière à ce que la contrainte temporelle $timecons(ts_1, ...ts_k)$ soit évaluée à vrai.

Expression de cible Les cibles sont évaluées en booléens à l'aide de la fonction d'évaluation $[[\cdot]]_T : \mathcal{T}arExp_{\mathscr{A}} \rightarrow \{Vrai, Faux\}$ définie comme suit :

$$
\begin{aligned}
[[\top]]_T(q) &= Vrai \\
[[\bot]]_T(q) &= Faux \\
[[op(a,v)]]_T(q) &= \begin{cases} Vrai & \text{si} \exists v' \in V_a : (a,v') \in q \wedge op(v',v) \\ Faux & \text{autrement} \end{cases} \\
[[\neg tgt]]_T(q) &= \neg [[tgt]]_T(q) \\
[[tgt \wedge tgt']]_T(q) &= [[tgt]]_T(q) \wedge [[tgt']]_T(q) \\
[[tgt \vee tgt']]_T(q) &= [[tgt]]_T(q) \vee [[tgt']]_T(q)
\end{aligned}
$$

L'évaluation d'une cible est vraie si la cible est toujours vraie ($\top$). À l'inverse, si la cible vaut faux ($\bot$), elle est évaluée à faux. Lorsque la cible comporte des opérations sur un attribut, elle est évaluée à vrai si et seulement si la requête contient cet attribut dont la valeur vérifie l'opération avec celle de la cible. L'évaluation des combinaisons logiques de cibles, comme pour les autres composants des politiques, est égale à la combinaison des évaluations de chaque cible primaire séparément.

Politiques Les politiques sont évaluées pour obtenir une décision afin de répondre à la requête d'accès. Si la politique vaut 1 (respectivement 0), la décision sera *Autoriser* (respectivement *Refuser*) pour la requête. Lorsque la politique est conditionnée par une cible, elle sera évaluée à vrai si la politique est applicable (c'est-à-dire que la cible est évaluée à vrai), sinon la réponse sera *NA*, signifiant que cette politique n'est pas applicable à l'évaluation de la requête q. Lors de la combinaison de politiques, la décision résulte de la combinaison des décisions des politiques individuelles.

Les politiques sont évaluées pour obtenir une décision dans $\mathscr{D}$ comme suit :

$$
\begin{aligned}
[[1]]_P(q) &= Autoriser \\
[[0]]_P(q) &= Refuser \\
[[(tgt, pol)]]_P(q) &= \begin{cases} [[pol]]_P(q) & \text{si } [[tgt]]_T(q) = Vrai \\ NA & \text{sinon} \end{cases} \\
[[ca(pol_1, \ldots, pol_m)]]_P(q) &= \mathbf{ca}([[pol_1]]_P(q), \ldots, [[pol_m]]_P(q)) \\
[[ca(\{pol_ref(x_1, \ldots, x_k) \mid path\})]]_P(q) &= \\
&\quad \mathbf{ca}([[pol_1]]_P(q), \ldots, [[pol_m]]_P(q))
\end{aligned}
$$

où $pol_1 \ldots, pol_m$ sont les politiques résultant de la résolution de la référence de politique :

$$
\begin{aligned}
&\{pol_1 \ldots, pol_m\} = \{uri(n_1, \ldots, n_k) \mid n_1 \in \mathcal{V}_{x_1}, \ldots, n_k \in \mathcal{V}_{x_k}, [[path[n_1/x_1, \ldots, n_k/x_k]]]_C(q) = Vrai\} \\
&[[ca(\{pol_tem(x_1, \ldots, x_k) \mid path\})]]_P(q) = \\
&\qquad \mathbf{ca}([[pol_1]]_P(q), \ldots, [[pol_m]]_P(q))
\end{aligned}
$$

où $pol_1 \ldots, pol_m$ sont les politiques résultant de la résolution du patron de politique :

$$\{pol_1, \ldots, pol_m\} = \{pol_tem(n_1/x_1, \ldots, n_k/x_k) \mid n_1 \in \mathcal{V}_{x_1}, \ldots, n_k \in \mathcal{V}_{x_k}, \\ [[path[n_1/x_1, \ldots, n_k/x_k]]]_C(q) = Vrai\}$$

Lors de la combinaison de références de politiques et de patrons de politiques, il est nécessaire de retrouver ces politiques en instanciant les variables avant de les évaluer. Ensuite, la combinaison s'effectue de la même manière que pour des politiques classiques.

Cette sémantique permet d'évaluer les politiques écrites en suivant le langage présenté en Section 2.4.1, et ainsi d'évaluer l'accès à des objets multi-propriétaires. Cependant, cette sémantique n'est pas suffisante pour évaluer l'accès à des objets comme un ensemble de sous-objets.

2.4.4. Évaluation d'accès d'objets composés

La sémantique présentée précédemment ne prend pas en compte l'évaluation des objets comme un ensemble composé d'un objet principal et de sous objets comme évoqué en Section 2.3. La modélisation explicite des objets composés est essentielle en raison de la nature des données dans les systèmes collaboratifs comme les réseaux sociaux : un commentaire fait sur une photo n'a aucun sens sans la photo, ou un objet de publication ne peut pas exister sans le profil associé. Par exemple, si un utilisateur demande à `afficher` un profil, nous devrions pouvoir déterminer les publications et les albums que l'utilisateur est autorisé à consulter sur le profil. Nous supposons que les objets composés contiennent un objet principal et une liste de sous-objets. Ces sous-objets sont les artefacts ayant une dépendance belongTo (définie en Section 1.3) avec l'objet principal. En conséquence, un post peut être considéré comme un objet composé où le post lui-même est l'objet principal et la liste des commentaires associés au post sont ses sous-objets. Par exemple, une photo peut être associée à une liste de commentaires et de tags associée à la photo.

Pour déterminer si un utilisateur peut *voir* (action `view`) un objet composé, nous devons déterminer non seulement si l'accès à l'objet principal est autorisé, mais également à ses sous-objets. À cette fin, si l'accès à l'objet principal est accordé (et que l'action est `view`), le mécanisme de contrôle d'accès doit non seulement récupérer la politique de l'objet principal, mais aussi récupérer de manière récursive ses sous-objets et créer une requête d'accès pour chacun d'eux afin de déterminer quels sous-objets peuvent être visualisés par l'utilisateur. Nous présentons ici un algorithme permettant d'évaluer les requêtes d'accès aux objets composés, dans un réseau social.

L'Algorithme 1 prend en entrée une requête (`query`), qui comprend les attributs caractérisant l'utilisateur demandant l'accès, l'objet pour lequel l'accès est demandé, l'action à effectuer et si l'action demandée est `share` la destination, indiquant l'utilisateur avec lequel le message doit être partagé, et l'ensemble des politiques (`policySet`)

Algorithm 1 Visibilité des objets composés

Input: `query` for an object `object`, made by a user `user` for an action `action`, with `destination` if existing; `policySet` representing the set of policies employed by the social network

Output: `visibility_list` containing the object and the associated visible sub-objects.

```
1: visibility_list ← ∅
2: candidate_list ← ∅
3: decision ← evaluation(query,policySet)
4: if decision=permit then
5:     append object to visibility_list
6:     if action=view then
7:         for all o such that belongTo(o, object) do
8:             append o to candidate_list
9:         end for
10:        while candidate_list≠ ∅ do
11:            extract o from candidate_list
12:            squery ← replace(query,o)
13:            decision ← evaluation(squery,policySet)
14:            if decision=permit then
15:                append o to visibility_list
16:                for all o_i such that belongTo(o_i, o) do
17:                    append o_i to candidate_list
18:                end for
19:            end if
20:        end whiles
21:    end if
22: end if
23: return visibility_list
```

employées par le réseau social pour réguler l'accès aux contenus. L'algorithme renvoie la liste des objets pour lesquels l'autorisation demandée est accordée (`visibility_list`). L'algorithme maintient également une liste (`candidate_list`) pour stocker les sous-objets pour lesquels l'accès doit être évalué.

L'algorithme commence par initialiser les listes `visibility_list` et `candidate_list` à l'ensemble vide (lignes 1-2). Ensuite, il évalue la requête donnée par rapport aux politiques employées par le réseau social à l'aide de la fonction `evaluation`. Cette fonction implémente la sémantique du langage de politique présenté dans la section 2.4.1 (ligne 3). Si l'évaluation renvoie une décision `permit` (1), l'algorithme ajoute l'objet à la liste des objets visibles (lignes 4-6).

Dans le cas où l'accès à l'objet est autorisée et si l'action demandée est `view`, l'algo-

rithme évalue la visibilité des sous-objets de l'objet demandé (lignes 7-22). En particulier, l'algorithme récupère l'identifiant de tous les sous-objets de l'objet demandé en exploitant la dépendance belongsTo et les stocke dans la liste `candidate_list` (lignes 6-9). Tant qu'il reste des objets dans `candidate_list`, l'algorithme extrait un objet de cette liste et génère une nouvelle requête pour cet objet en utilisant la fonction `replace` (ligne 12). Intuitivement, cette fonction remplace les attributs concernant l'objet demandé par ceux concernant le sous-objet, tout en gardant inchangés les attributs concernant l'action et le demandeur. [7] La nouvelle requête est évaluée par rapport aux politiques employées par le réseau social (ligne 13), et si la décision est `permit`, le sous-objet est ajouté à `visibility_list` et ses sous-objets sont ajoutés à `candidate_list` (lignes 14-19), de manière similaire à ce qui est fait pour l'objet demandé. Il est utile de noter que les lignes 16 - 18 permettent de retrouver les sous-objet d'un sous-objet. Selon le système ou le niveau de granularité de l'affiche de l'objet principale, il peut être utile de ne pas exécuter ces lignes ou d'ajouter une condition sur la profondeur des sous-objets à récupérer ou le type.

L'algorithme continue jusqu'à ce que la visibilité de tous les sous-objets ait été calculée (c'est-à-dire que candidate_list soit vide) et se termine en renvoyant la liste des objets pour lesquels l'accès est accordé (`visibility_list`) (ligne 23). Il est à noter que cette liste contient, au plus l'objet demandé pour toute action autre que `view` car les autres actions n'affectent pas les sous-objets.

L'Exemple 15 montre comment fonctionne cet algorithme en se basant sur le scénario présenté en Figure 2.1.

Exemple 15 (Évaluation d'un objet composé) *Considérons que Christopher fait une demande pour voir le post de Bob* `sp11` *(Figure 2.1). La fonction* `evaluation` *de l'Algorithme 1 implémente la politique Facebook présentée dans l'Exemple 10.*

Selon la politique Facebook, la `décision` *d'accès est calculée en récupérant les préférences de confidentialité de l'hôte de la publication, c'est-à-dire Bob. Supposons que la politique de Bob autorise Christopher à voir sa publication, ce qui signifie que la fonction* `évaluation` *renvoie* `permit` *comme* `décision`*.*

Ensuite, l'algorithme récupère tous les commentaires liés à la publication demandée par la dépendance belongsTo *car l'action de la requête est* `voir` *et les stocke dans la liste* `candidate_list`*.*

Dans notre scénario, `candidate_list` *= [commentsp12c1]. Une nouvelle requête* `squery` *est effectuée pour évaluer la visibilité de chaque commentaire dans la liste, et la fonction* `évaluation` *évalue les préférences du propriétaire du commentaire, c'est-à-dire celles de Daphné.*

Si Christopher est autorisé à voir le commentaire par Daphné, alors la sortie finale de l'algorithme, `visibility_list`*, contiendra la publication ainsi que le commentaire de Daphné. Sinon, seule la publication de Bob sera visible pour Christopher et renvoyée dans la* `visibility_list`*."*

7. Notez qu'il n'y a pas d'attributs concernant la `destination` car ces attributs ne sont présents que dans les demandes d'action `share`.

2.5. Prototype et validation

Pour évaluer notre modèle et montrer la faisabilité de notre approche, nous avons développé un prototype [8], implémentant un contrôle d'accès pour un système de type réseau social. En particulier, il implémente l'algorithme d'évaluation des politiques présenté dans la section précédente. Son fonctionnement général est le suivant. Pour chaque requête entrante, il va l'évaluer en fonction de l'ensemble des politiques qu'il possède et renvoyer une réponse, comprenant la décision et le ou les objets visibles par le demandeur (y compris les sous-objets de l'objet demandé si l'action est `voir`).

Le but des tests menés est d'étudier l'évolution des performances lors de l'agrandissement du graphe, car il s'agit d'un contrôle d'accès conçu pour être appliqué à des réseaux sociaux, où le volume de données et d'utilisateurs est particulièrement important. L'objectif est d'identifier les paramètres qui influencent le plus les performances du système lorsque la taille du graphe augmente. En particulier, nous analysons l'impact des facteurs tels que le nombre de nœuds, la densité des relations ou la quantité de sous-objets par photo ou post, afin de mieux comprendre comment ces éléments affectent l'efficacité globale du calcul d'accès.

Afin d'obtenir des résultats exploitables et représentatifs, nous avons développé un générateurs de graphes et un générateur de politiques utilisateurs dans le but de pouvoir tester notre approche sur une multitude de scénarios différents.

Dans cette section, nous présentons d'abord l'architecture et les choix d'implémentation du prototype, puis nous discutons de l'approche choisie pour les générateurs. Enfin nous discutons des paramètres et des résultats de notre évaluation.

2.5.1. Architecture

Les principaux composants de notre prototype sont représentés dans la Figure 2.3. Le composant principal de notre prototype est le *module de contrôle d'accès* (« Access Control Module ») qui intercepte les demandes d'accès des utilisateurs et calcule la décision d'accès en fonction de l'algorithme décrit dans la Section 2.4.4.

Lors de la réception d'une demande d'accès, le *module de contrôle d'accès* récupère les politiques pertinentes dans les fichiers *Policy files*, qui contiennent les politiques employées par le réseau social pour les différents types de contenu ainsi que les préférences d'accès des utilisateurs (par exemple, les politiques de l'Exemple 12). Pendant l'évaluation des politiques, le module interagit avec le *module de raisonnement* (« Reasoning Module ») lorsque l'évaluation d'une politique nécessite l'évaluation d'une condition de chemin.

Le module de raisonnement est implémenté sous forme de programme Python utilisant la bibliothèque pyDatalog [9]. Cela nous permet de tirer parti d'algorithmes

8. Une version préliminaire du prototype est présentée dans BERTOLISSI, MARTINEZ ANTON, TESTUD et al. 2022.

9. https://pypi.org/project/pyDatalog/

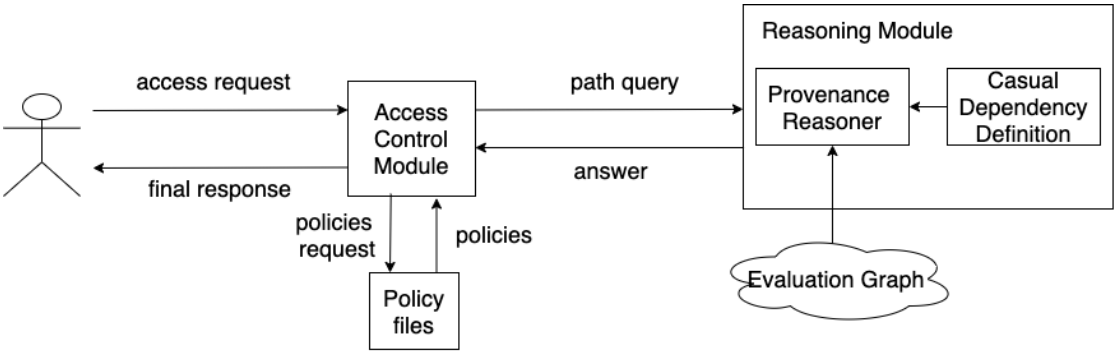


FIGURE 2.3. – Architecture du prototype

efficaces développés pour la résolution de requêtes dans les solveurs Prolog, en particulier la restriction Datalog. Nous avons implémenté des règles à la fois pour la construction du graphe d'évaluation et pour la résolution des requêtes de chemin basées sur le graphe obtenu et la définition des dépendances causales et de relations.

Le *graphe d'évaluation* est construit à partir des informations de provenance récupérées dans les journaux du système ainsi que des informations sur les relations entre utilisateurs, qui sont mises à jour au moment de la demande de contrôle d'accès. Nous considérons dans ce prototype que l'on obtient directement le fichier pyDatalog contenant les informations sous formes de règles Datalog. Le graphe d'évaluation est construit en fusionnant le graphe de provenance et des relations comme défini dans la Section 1.6. Les requêtes de chemin sont traduites en requêtes Datalog et le solveur Datalog sous-jacent choisit automatiquement les faits et les règles nécessaires pour les résoudre (par exemple, en utilisant pour la déduction les règles correspondant aux dépendances occasionnelles de Table 1.1). Lorsqu'une politique contient des contraintes de temps, le module de contrôle d'accès vérifie s'il existe un ensemble d'horodatages récupérés par le solveur respectant les contraintes de temps. Lorsque la requête logique correspond à un chemin contenant une référence de politique, les variables non bornées qui peuvent apparaître dans cette requête sont instanciées par le solveur avec toutes les valeurs possibles (correspondant aux nœuds du graphe d'évaluation) qui la satisfont. Les réponses sont ensuite utilisées pour récupérer dynamiquement des politiques pertinentes supplémentaires (utilisateur) à partir des fichiers de politiques. Ces politiques utilisateur sont ensuite évaluées par le module de contrôle d'accès pour déterminer si l'accès doit être autorisé.

2.5.2. Générateurs de graphes et de politiques

Dans le but de générer une grande quantité de scénarios, nous avons développés deux générateurs : un générateur de graphes et un générateur de politiques, simulant des graphes et des politiques qui auraient pu être extraits d'un réseau social simple comme décrit en Section 1.2.1.

Générateur de graphes Le générateur de graphes prend en entrée un fichier contenant la définition des dépendances, et des entiers fixant le nombre d'utilisateurs, post, photos, commentaires et tags par publication du système.

D'un côté il génère les relations entre les utilisateurs. Certaines relations sont terminées, donc avec un horodatage de fin différent de ∞ et d'autres toujours actives. Afin de se rapprocher des interactions de vrais utilisateurs, nous utilisons le modèle « friend to friend » expliqué dans LEVENS et al. 2022. Le modèle « friend-to-friend » fonctionne en construisant progressivement un graphe de relations d'amitié entre utilisateurs. On commence par créer un petit graphe initial de trois utilisateurs, où chacun est ami avec les deux autres. Ensuite, pour chaque nouvel utilisateur u_n ajouté, on le lie à un utilisateur existant u_e choisi aléatoirement avec une probabilité p. Ensuite, parmi les amis de cet utilisateur u_e, un autre utilisateur u_f est sélectionné au hasard, et avec une probabilité q, une relation d'amitié est également créée entre lui u_f et le nouvel utilisateur u_n. Les relations d'amitié sont symétriques avec ce modèle. La valeur des probabilités pour les relations toujours actives ou terminées est de $p = 1$ et $q = 0.75$. On applique ce modèle deux fois : une fois pour les relations terminées et une fois pour les relations toujours existantes.

La création des dépendances du graphe modélisant la création de post, de photos et d'albums sont créés de façon aléatoire. L'ajout de tags est réalisé en prenant parmi les amis du propriétaire de la photo. S'il n'a pas d'amis ou pas assez d'amis, des utilisateurs aléatoires sont pris parmi les utilisateurs existants. Pour les commentaires, le propriétaire est aléatoire parmi l'ensemble des utilisateurs.

Les horodatages sont créés aléatoirement dans un intervalle prédéfini. Une vérification est réalisée pour s'assurer de la cohérence de ces horodatages (une relation commence avant de terminer, un commentaire est généré après une photo,etc.).

Le générateur produit un fichier Python contenant toutes les dépendances et relations, écrite en utilisant la librairie *pyDatalog*.

Générateur de politiques Afin de simuler les préférences d'accès des utilisateurs, le générateur de politiques crée un fichier dédié pour chaque utilisateur, contenant sa politique d'accès. Chaque politique définit des règles d'accès spécifiques pour chaque objet du réseau et chaque action possible. Ces règles peuvent être celles par défaut, signifiant que l'accès est accepté, ou elles peuvent inclure des conditions sur les relations que les utilisateurs entretiennent et les dépendances dans le graphe. Le générateur génère des politiques avec et sans l'extension de temps pour comparer les performances. Dans le cas des politiques non-temporelles les relations peuvent être uniquement amis, amis d'amis ou la négation de ces deux relations. Lors de la génération avec des horodatages, les politiques sont plus fines, on peut utiliser les relations des politiques non-temporelles mais aussi les conditions vues dans les exemples de la Section 2.4. Ainsi les conditions d'accès peuvent dépendre de relations dans le passé, de la date de début de la relation, etc. Plus formellement, les conditions de chemins peuvent être composées d'une composition aléatoire d'au plus quatre dépendances de areRelated (utilisée pour modéliser la relation ami), areRelated^2 (utilisée pour modéliser la relation ami d'ami) et owns. Les politiques non-temporelles n'utilisent que

les trois dépendances précédentes. Comme l'extension temporelle permet plus de flexibilité dans les politiques, les négations de areRelated et areRelated^2 peuvent aussi être choisies. Une dépendance supplémentaire, wasGeneratedBy, est utilisée dans le cas d'une politique exprimant une contrainte de temps entre le début d'une relation utilisateur et la création d'un artefact (comme dans l'Exemple 13). Les valeurs d'horodatage et les contraintes de temps sont choisies de manière aléatoire et ajoutées aux dépendances. Il est utile de préciser que la politique système n'est pas générée à chaque scénario et est incluse dans le module du contrôle d'accès.

Les fichiers générés sont des fichiers textes, structurés comme montré en Annexe A.

2.5.3. Expérimentations

Nous avons réalisé plusieurs expériences pour évaluer la faisabilité pratique de notre approche dans le contexte des réseaux sociaux. Dans un premier temps, nous avons étudié le modèle sans observations temporelles, comme présenté dans BERTOLISSI, MARTINEZ ANTON et ZANNONE 2023. Ensuite, nous avons étudié l'évolution des performances du modèle par rapport aux performances du modèle sans l'extension d'observation de temps.

Cette section présente notre évaluation expérimentale à l'aide du prototype : nous décrivons d'abord les paramètres de l'expérience puis discutons des résultats. Déterminer si un utilisateur peut accéder à un objet donné nécessite l'utilisation du graphe d'évaluation (composé du graphe de provenance et du graphe des relations), représentant l'évolution des objets au sein du système et les relations sociales entre les utilisateurs, et les politiques de contrôle d'accès définies par le réseau social et par les utilisateurs pour réguler l'accès aux objets téléchargés sur la plateforme du réseau social (cf. Algorithme 1).

Pour réaliser nos expériences, nous avons généré aléatoirement des graphes d'évaluation. Dans sa configuration par défaut, un graphe d'évaluation comprend 400 utilisateurs, 200 publications, 200 photos. Chaque publication/photo comporte au maximum 100 commentaires. De plus, une photo comporte au maximum 50 tags et est partagée au maximum 20 fois. Ces paramètres ressemblent à la configuration typique des réseaux sociaux tels que Facebook, qui limite à 50 le nombre de tags pouvant être associés à une photo [10]. En ce qui concerne la politique de contrôle d'accès, nous avons utilisé celle présentée dans l'Exemple 12 comme politique de base et avons généré de manière aléatoire les politiques utilisateur. En particulier, nous avons utilisé le générateur de politiques pour chaque utilisateur, en créant une politique pour chaque type d'objet (par exemple, photos, publications) et d'action (par exemple, publication, affichage, partage).

Comme l'action `view` peut nécessiter plus d'interactions avec les modules *reasoning* et *access control*, en raison de l'évaluation des requêtes liées aux sous-objets (cf. Section 2.4.4), nos tests se limitent aux requêtes pour cette action. Cela nous permet

10. https://www.facebook.com/help/267689476916031, consulté en juillet 2024

d'évaluer notre approche lorsque nous traitons des objets composés.

Les performances de l'évaluation des politiques dépendent de la taille du graphe d'évaluation, du nombre de politiques et du nombre de sous-objets d'un objet composé. Pour étudier l'évolution des performances entre l'évaluation des politiques temporelles et celles non-temporelles, nous avons réalisé 3 expériences, chacune dans les modèles non-temporel et temporel.

Le premier ensemble d'expériences a pour but de mesurer les performances globales des modèles avec et sans extension temporelle, en faisant varier la taille du graphe. Dans une première expérience, nous faisons varier le nombre de photos et posts dans le système et regardons l'impact de ces modification sur les temps d'exécution. Il convient de noter que l'augmentation du nombre d'artefacts nécessite également l'introduction de nœuds représentant les instances de processus utilisées pour leur génération et des dépendances occasionnelles reliant les instances de processus aux artefacts d'entrée et de sortie ainsi qu'aux utilisateurs exécutant ces processus. Les deuxième et troisième expériences sont similaires à la première. La deuxième se concentre sur la variation du nombre d'utilisateurs, afin d'augmenter le nombre de politiques utilisateurs (respectivement la variation du nombre de tags et de commentaires par photo et post pour augmenter le nombre de sous-objets). Grâce aux résultats de performance obtenus pour le modèle avec et sans contraintes temporelles, nous avons pu calculer le coefficient de performance, qui représente le facteur de temps supplémentaire requis par le modèle avec contraintes temporelles par rapport à celui sans contraintes présenté dans BERTOLISSI, MARTINEZ ANTON et ZANNONE 2023.

Pour mieux comprendre les variations des performances du prototype, nous avons également mesuré le temps requis pour récupérer et évaluer une politique dans le cadre du modèle avec l'extension temporelle. De plus, nous avons étudié la relation entre le temps d'évaluation d'une requête et le nombre de contraintes temporelles présentes dans les politiques nécessaires à l'évaluation de cette requête.

Les réseaux sociaux n'affichent généralement pas tous les commentaires associés aux publications et aux photos. Par exemple, Facebook affiche par défaut deux commentaires associés aux publications et affiche des commentaires supplémentaires par lots de 20 à la demande d'un utilisateur. Nous avons pris en compte ce comportement dans nos expériences et étudié dans quelle mesure le nombre de sous-objets affichés influence le temps d'évaluation. Plus précisément, nous avons répété chaque test avec un nombre différent d'éléments associés (2, 10, 50, 100 ou 1000).

Une analyse préliminaire a montré qu'il existe une différence de temps significative entre les évaluations des requêtes autorisées (jusqu'à 1 minute 30 secondes) et celles des requêtes refusées (moins de 10 millisecondes) dans les deux modèles. La différence de performances entre le modèle non-temporel et celui utilisant l'extension temporelle est limitée par un coefficient de performance de 2. Le nombre de contraintes temporelles dans une évaluation a un impact plus important sur les performances que la complexité des contraintes temporelles. Nous avons observé que le temps d'évaluation d'une politique est quasi instantané, même dans le cas de politiques

temporelles. En revanche, pour la récupérer, il est beaucoup plus long (jusqu'à 100 fois plus long).

Pour cela, pour chaque configuration de test, nous avons généré aléatoirement 8 graphes d'évaluation (et les politiques utilisateur associées). Pour chaque graphe, nous avons généré 25 requêtes différentes. Dans les résultats, nous reportons le temps moyen pour chaque configuration.

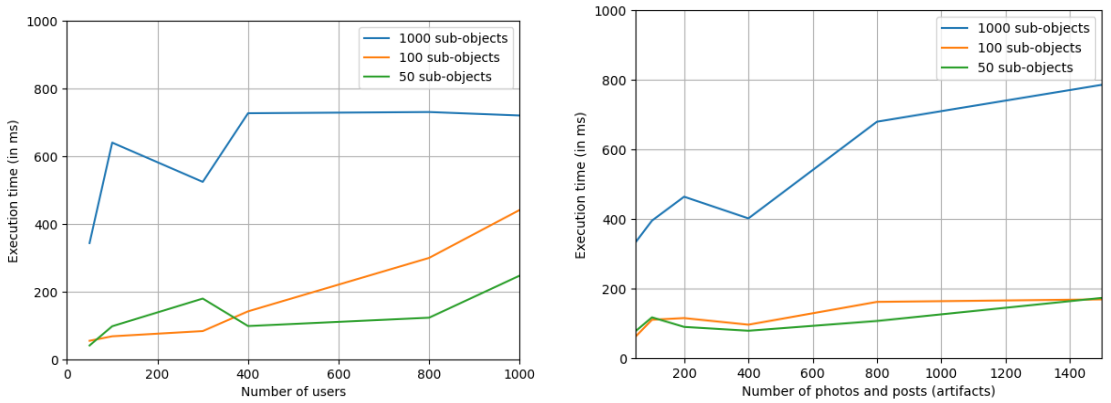
Les expériences ont été réalisées sur un MacBook Pro équipé d'une puce Apple M2 et de 16 Go de RAM.

2.5.4. Résultats et discussion

Les résultats de nos tests sont montrés dans les Figures 2.4, 2.5, 2.6, 2.7, et 2.10, comportant les résultats pour la version temporelle et non temporelle de notre modèle. Les graphiques représentent le temps moyen d'évaluation d'une requête. La Figure 2.6 représente le temps moyen d'évaluation, en ne tenant compte que des requêtes menant à un refus. La Figure 2.8 présente la comparaison entre la version temporelle et la version non temporelle du modèle. La Figure 2.9 montre le ratio entre le temps de récupération et d'évaluation d'une politique utilisateur.

Les résultats de nos expériences montrent un grand écart entre le temps d'évaluation d'une requête autorisée et d'une requête rejetée. L'augmentation des photos et des posts et l'augmentation du nombre d'utilisateur, que ce soit dans le modèle sans extension temporelle (cf. Figure 2.4) ou avec (cf. Figure 2.5) ont un faible impact sur les performances. Cependant, la quantité de sous-objets dans les objets composés a un impact élevé sur les performances, en particulier sur les requêtes recevant une réponse positive (Figure 2.7), prenant plus d'une seconde pour renvoyer le résultat. Cependant, dans les réseaux sociaux existants le nombre de sous-objets est limité. L'impact de l'utilisation d'horodatages et conditions temporelles est faible, comme montré dans la Figure 2.8.

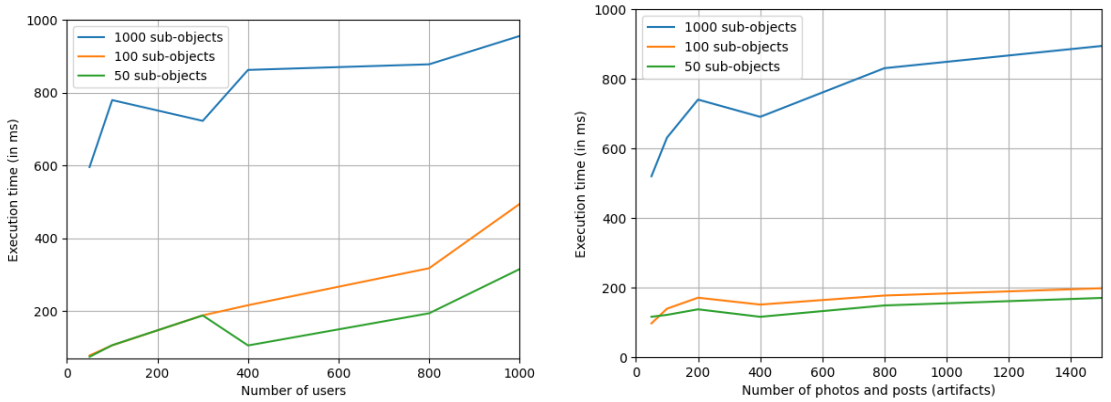
Enfin, la Figure 2.9 montre que les temps nécessaire à la récupération des politiques est plus important rapport à celui nécessaire à leur évaluation.



(a) Évolution des performances lors de la variation du nombre d'utilisateurs

(b) Évolution des performances lors de la variation du nombre de photos et posts

FIGURE 2.4. – Temps d'exécution moyen pour résoudre une requête dans le modèle sans extension temporelle



(a) Évolution des performances lors de la variation du nombre d'utilisateurs

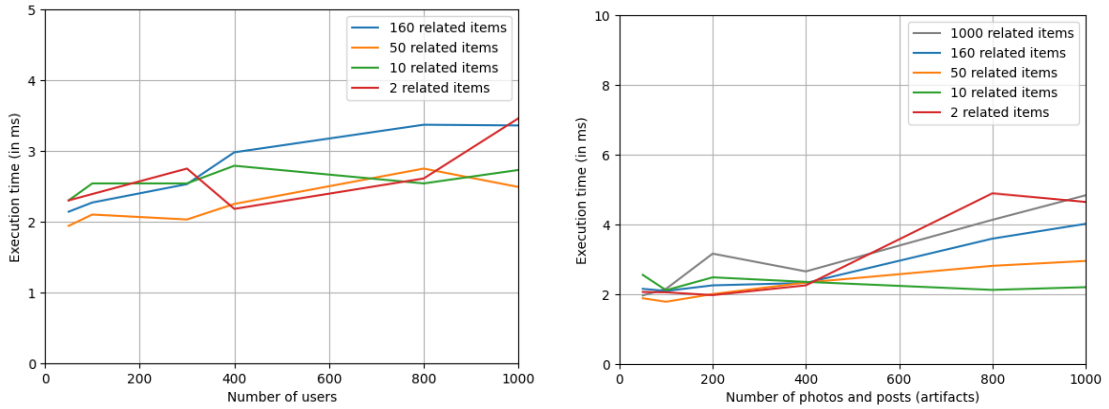
(b) Évolution des performances lors de la variation du nombre de photos et posts

FIGURE 2.5. – Temps d'exécution moyen pour résoudre une requête dans le modèle avec extension temporelle

La suite de cette section détaille plus en profondeur les observations tirés des résultats mentionnés ci-dessus.

Accès autorisé et accès refusé On observe qu'il y a un très grand écart entre le temps moyen de réponse à une requête (Figures 2.5 et 2.4) et le temps nécessaire dans le cas où l'accès est refusé (Figure 2.6). Dans le cas d'une requête refusée, la réponse est quasi instantanée (moins de 6 millisecondes), quelles que soient les valeurs des paramètres (Figure 2.6), alors qu'en moyenne une requête peut prendre jusqu'à presque 1 seconde dans le cas où il y a plus de 1 000 sous-objets. Ceci s'explique

car lorsqu'un accès est refusé, il n'est pas nécessaire de retrouver les sous-objets et les politiques associées.



(a) Temps d'exécution moyen pour un nombre variable d'utilisateurs

(b) Temps d'exécution moyen pour un nombre variable de photos et de publications

FIGURE 2.6. – Résultats de performance pour les requêtes refusées avec des politiques temporelles

Impact des sous-objets La Figure 2.7 montre dans les deux cas, avec ou sans extension temporelle, le temps de réponse croit fortement lors de l'augmentation du nombre de sous-objets autorisées. Ce temps peut dépasser la seconde en cas de grand nombre. On peut observer ceci dans les courbes « 1000 *related items* » des Figures 2.4 et 2.5 donnant le temps d'exécution moyen lorsqu'on autorise 1000 sous-objets par post/photo, qui montrent un grand écart par rapport aux courbes représentant les requêtes avec au plus 50 et 100 sous-objets. Cependant, en pratique, le nombre de sous-objets est souvent limité dans OSN, les réseaux sociaux restreignent souvent le nombre de sous-objets, comme évoqué précédemment avec l'exemple de Facebook. Même si certains posts peuvent devenir viraux, avec des milliers de commentaires, ces cas extrêmes se produisent rarement. En effet, des études ont montré qu'un message contient en moyenne moins de cinq commentaires (BARBERIO et al. 2017). De plus, les réseaux sociaux limitent généralement le nombre de commentaires affichés aux utilisateurs. Par exemple, Facebook n'affiche initialement que deux commentaires par défaut, les commentaires supplémentaires étant affichés par lots de 20.

Nous avons aussi observé que la récupération d'une politique est une opération très lourde. Plus il y a des sous-objets, plus le nombre de politiques à considérer est important. Améliorer les performances de récupération de politiques est une piste à suivre pour améliorer les performances globales de l'outil.

Comparaison entre version temporelle et non temporelle Si l'on regarde les résultats d'évaluation de requêtes mis en avant par les Figures 2.4 et 2.5, on observe qu'évaluer une requête dans le modèle considérant les horodatages (Figure

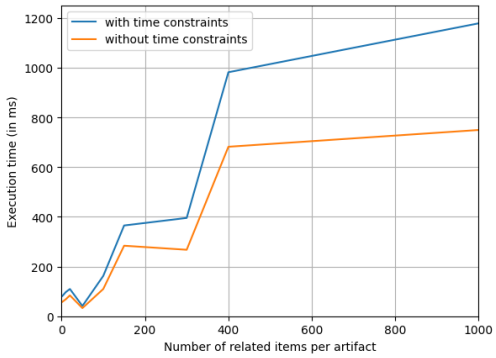
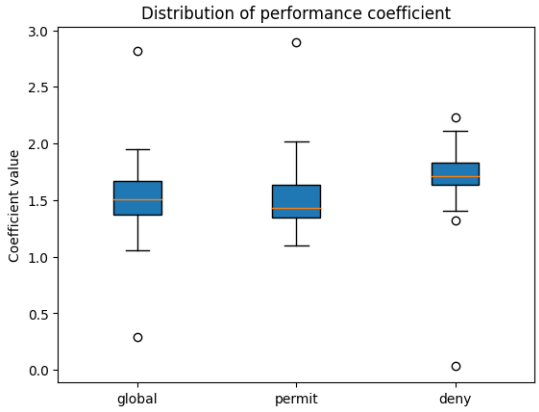


FIGURE 2.7. – Moyenne du temps d'évaluation des requêtes autorisées pour un nombre variable d'éléments associés par artefact

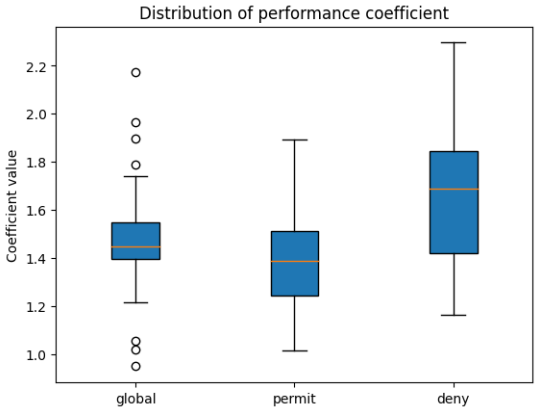
2.5) consomme plus de temps que dans le cas non-temporel (Figure 2.4). Cependant, comme le montre le graphique de la Figure 2.8c, cette différence reste linéaire. À partir des résultats obtenus dans les expériences, nous avons calculés les coefficients de performance entre l'évaluation d'une requête dans le modèle non-temporel et dans le modèle temporel. Ce coefficient est compris entre 1,2 et 2, comme le montrent les Figures 2.8a et 2.8b. Le temps supplémentaire nécessaire pour évaluer les requêtes dans le modèle de contrôle d'accès temporel est tout à fait raisonnable.

Impact des politiques Afin de mieux comprendre l'augmentation du temps requis pour évaluer une requête en fonction du nombre croissant de sous-objets attachés, nous avons entrepris d'étudier les performances liées aux politiques d'accès. Le temps nécessaire pour retrouver une politique peut être considéré comme une constante, dépendant du nombre de politiques, ce qui, dans notre cas, est directement lié au nombre d'utilisateurs. En général, nous pouvons imaginer que récupérer un fichier de politique utilisateur est plus rapide que l'évaluer, car pendant cette dernière, il est nécessaire de rechercher des chemins dans le graphe pour instancier certaines variables ou vérifier certaines conditions. Cependant, il est apparu que le processus de recherche d'une politique est plus coûteux en temps que celui de l'évaluer. La Figure 2.9 illustre le ratio entre le temps requis pour récupérer une politique et celui pour l'évaluer. On y observe clairement que la recherche de la politique est plus longue et que ce temps augmente proportionnellement avec le nombre d'utilisateurs.

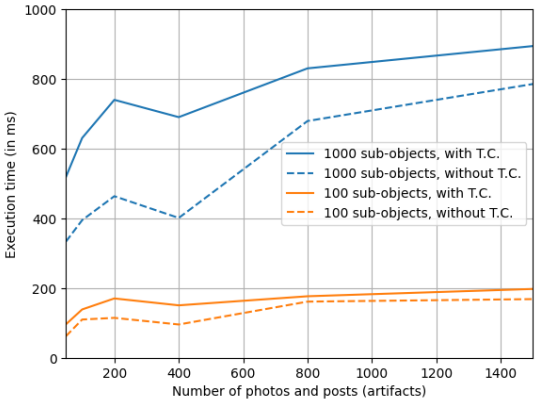
En effet, les résultats ont montré que trouver et ouvrir un fichier de politique peut prendre jusqu'à 20000 fois plus de temps (la valeur tombe à 112 en considérant les valeurs entre le premier et le dernier quartile) que de le résoudre. Le temps médian de récupération est de $0,117ms$ lorsque le temps médian d'évaluation est de $0,002$. Selon la médiane, il faut 20 fois plus de temps pour trouver une politique que pour l'évaluer. Nous conjecturons que ce temps très important nécessaire à la recherche d'une politique est dû au choix de stockage de celles-ci sous forme de fichier texte. Une modification de leur stockage, par exemple dans une base de données permettrait probablement d'améliorer les performances. En effet, il n'y aurait plus besoin de



(a) Distribution pour un nombre variable d'utilisateurs



(b) Distribution pour un nombre variable d'artefacts



(c) Comparaison des performances des modèles avec et sans contraintes de temps (T.C.)

FIGURE 2.8. – Distribution des coefficients de performance

manipuler des fichiers texte et de nombreux efforts ont été déployés pour optimiser les requêtes dans les bases de données.

Un autre point d'intérêt liés aux politiques peut être le lien entre la complexité de celles-ci et la performance. La figure 2.10 montre que le nombre de conditions temporelles utilisées par requête a un impact significatif sur le temps d'évaluation. Ce résultat confirme l'observation du temps d'évaluation lorsque le nombre de sous-objets, et donc le nombre de politiques utilisateur, augmente.

Globalement, nos expériences montrent que les performances de l'évaluation des politiques dépendent principalement du nombre d'éléments liés associés aux posts et aux photos et du nombre d'éléments récupérés. Si l'évaluation s'avère lourde en calcul lorsque les posts et les photos sont associés à un grand nombre de commentaires et de tags ou lorsque les permissions de plusieurs sous-objets doivent être évaluées, en réalité, elles sont souvent limitées dans les réseaux sociaux. A cet égard, les résultats montrent que notre approche est réalisable pour les configurations réellement adop-

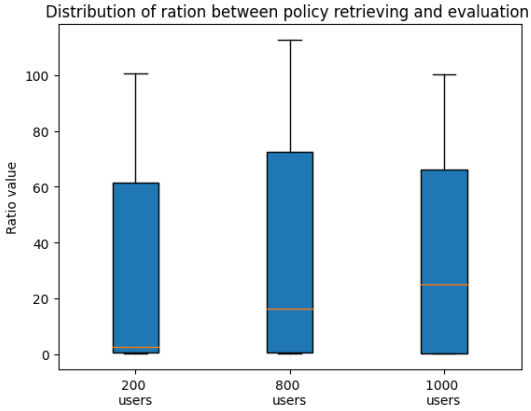


FIGURE 2.9. – Distribution du ratio entre le temps de récupération et d'évaluation d'une politique

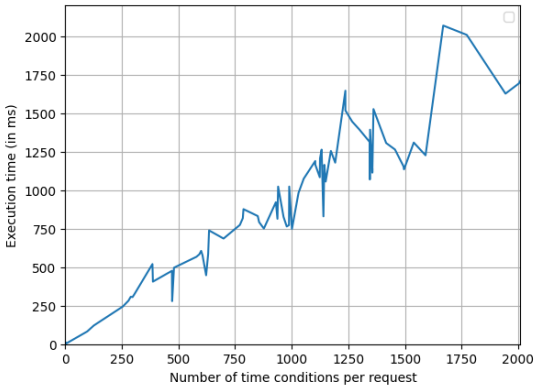


FIGURE 2.10. – Évolution du temps d'évaluation pour un nombre variable de conditions temporelles

tées par les réseaux sociaux. Il convient de noter que notre implémentation prototype n'a pas fait l'objet d'une phase d'optimisation. Les résultats de nos expériences suggèrent que changer le format de stockage des politiques des utilisateurs peut conduire à une amélioration des performances globales. Nos résultats montrent également que la surcharge de calcul du modèle temporel n'est pas significativement importante, en comparaison au gain de flexibilité.

2.6. Conclusion et Discussion

Dans ce chapitre nous avons présenté une approche de contrôle d'accès pour des réseaux sociaux qui exploite les données de provenance horodatées, complétées par des informations sociales, pour avoir un mécanisme précis permettant de faire respecter les préférences de confidentialité multipartites. Notre modèle répond aux critères de spécification des politiques et partiellement à ceux de gouvernance spécifiés dans

PACI et al. 2018. Nous avons fourni une sémantique d'évaluation adaptée, basée sur celle présentée dans BERTOLISSI, HARTOG et al. 2019 et proposons un algorithme d'évaluation pour les objets composés. Un prototype de preuve de concept, ainsi qu'une évaluation expérimentale, sont fournis pour démontrer la faisabilité de notre approche. L'étape suivante de la validation, laissée pour de futurs travaux, consistera en une étude impliquant des utilisateurs afin de tester la facilité et la pertinence d'utilisation du modèle. Cependant, cela nécessitera au préalable le développement d'un prototype plus abouti, intégrant une interface graphique, permettant aussi de répondre aux critères d'ergonomie et transparence.

Nous avons étudié l'extension à d'autres systèmes collaboratifs centrés ou non sur la communauté, tel que les chaînes d'approvisionnement et les clouds proposant des outils collaboratifs (édition de documents en ligne, ...). Notre modèle permet de s'adapter à de tels systèmes car il supporte les principaux besoins tel que la délégation (ou remplacement) qui peut être modélisé à l'aide des relations entre utilisateurs. Un contrôle d'accès basé sur la provenance peut prendre en compte les séparations dynamiques ou non des responsabilités (« separation of duty » ou « dynamic separation of duty » en anglais) comme le montre le travail NGUYEN et al. 2013. La notion de sous-objets peut être étendue à d'autres types de contenu, tels que des sections d'un document ou des parties d'un projet.

Grâce aux horodatages, notre modèle peut calculer des accès basés sur des relations transitoires, ce qui permettrait de l'étendre de manière assez naturelle aux systèmes IoT. Cependant, un travail important devra être mené en matière de performances, car l'une des principales problématiques du contrôle d'accès dans l'IoT est sa limitation en termes de capacités de calcul et de ressources. Un choix adapté de stockage pour les politiques, ainsi qu'une implémentation utilisant des technologies appropriées, permettrait d'améliorer les performances du modèle.

L'intégration de notre travail dans XACML est laissée pour de futures améliorations. L'idée est de spécifier les conditions de chemin avec l'élément `<PolicyIdReference>` ou de les coder sous forme de fonctions définies par l'utilisateur invoquant un service externe en utilisant par exemple un framework comme SAFAX (KALUVURI et al. 2015). Dans notre contexte, chaque dépendance causale peut être implémentée sous forme de fonction définie par l'utilisateur qui invoque le module Provenance pour sa résolution.

Notre modèle se base sur des algorithmes de combinaison classiques, comme DOV ou POV, et par conséquent il ne permet pas une gestion fine des conflits pouvant exister entre les différentes préférences des utilisateurs. Dans le chapitre suivant, nous abordons cette problématique nous présentons une stratégie de résolution de conflits, pouvant être employée dans ce contrôle d'accès.

3. Gestion de conflits dans les systèmes multi-utilisateurs

Sommaire

3.1. Introduction . . . 101
3.2. Stratégies de résolution de conflits dans la littérature . . . 101
3.2.1. Algorithmes de combinaison . . . 102
3.2.2. Stratégies de majorité et de consensus . . . 102
3.2.3. Résolution basée sur la théorie des jeux . . . 103
3.2.4. Résolution de conflits dans les photos . . . 103
3.2.5. Résolution par seuil . . . 104
3.2.6. Facteurs de décision . . . 105
3.3. Défis de la gestion des conflits . . . 106
3.4. Modèle de résolution de conflits basé sur un seuil . . . 109
3.4.1. Sensibilité . . . 110
3.4.1.1. Confiance de l'utilisateur dans une communauté . . . 111
3.4.1.2. Sensibilité d'un objet . . . 112
3.4.2. Intérêt à partager . . . 113
3.4.2.1. Exactitude d'un objet . . . 114
3.4.2.2. Propagation communautaire . . . 115
3.4.2.3. Intérêt à partager . . . 116
3.4.3. Résolution du conflit . . . 117
3.4.3.1. Coefficients de crédibilité . . . 118
3.4.3.2. Résolution du conflit . . . 119
3.5. Validation du modèle . . . 120
3.5.1. Variations des facteurs . . . 120
3.5.2. Analyse comparative par rapport aux modèles basés sur le seuil 121
3.5.3. Prototype . . . 123
3.5.3.1. Architecture . . . 123
3.5.3.2. Analyse comparative par rapport aux stratégies basiques 124
3.5.3.3. Étude comparative détaillée des approches fines . . . 125
3.5.3.4. Performances . . . 125
3.6. Conclusion et Discussion . . . 126

3.1. Introduction

Avec la démocratisation des systèmes collaboratifs centrées sur la communauté et les contrôles d'accès collaboratifs, le risque de conflits lors de l'évaluation d'un accès a fortement augmenté. En effet, par nature, ces systèmes gèrent des données conjointement détenues par plusieurs utilisateurs (les contrôleurs comme définis en Section 1.2). Plusieurs politiques peuvent être nécessaires lors de l'évaluation d'une requête, pouvant conduire à des décisions divergentes. Cela génère un conflit qui doit être résolu en trouvant une décision d'accès collective. Il est important de noter que les relations dans ce type de système imitent les relations dans le monde physique. De plus dans les systèmes de type réseaux sociaux, des bénéfices existent au partage des données. Par exemple, le partage peut permettre de garder contact avec ses proches (KRASNOVA et al. 2010). Il est donc crucial de gérer les conflits de manière proche du comportement humain.

La gestion des conflits dans le contrôle d'accès fait donc référence au processus de résolution des situations où des politiques d'accès avec des décisions contradictoires sont appliquées. Elle vise donc à garantir que les droits d'accès sont appliqués de manière cohérente et sécurisée en utilisant des mécanismes de résolution appropriés.

Afin d'obtenir une résolution de conflits plus précise et adaptée au problème soulevé par les réseaux sociaux, nous proposons une stratégie de résolution de conflits qui peut être intégrée à un système de contrôle d'accès pour résoudre ce problème. Ce chapitre est basé sur l'article MARTINEZ ANTON et al. 2024 dont le contenu est repris ici avec des explications et des ajouts supplémentaires. Cette stratégie est compatible avec le modèle de contrôle d'accès proposé dans le chapitre précédent.

Dans ce chapitre nous présentons notre travail consistant en une solution de gestion de conflits qui utilise la provenance des données et les relations entre les utilisateurs, notamment leur niveau de confiance, pour décider d'une décision unique. Nous présentons en Section 3.2 un survol des différentes techniques proposées dans la littérature, ainsi qu'un focus sur les principales notions utilisées dans la gestion de conflit. Dans la Section 3.3, nous illustrons les principaux défis liés à la gestion des conflits à l'aide d'un exemple. Nous présentons notre modèle de gestion de conflits utilisant la provenance en Section 3.4 répondant aux défis mis-en-avant et validons notre approche à l'aide d'un prototype et de différentes expériences (Section 3.5).

3.2. Stratégies de résolution de conflits dans la littérature

La gestion des conflits dans les systèmes collaboratifs centrés sur la communauté a été largement étudiée ces dernières années. Trouver un algorithme optimal de résolution des conflits, en particulier pour les réseaux sociaux, n'est pas une tâche triviale. Plusieurs solutions ont été proposées, utilisant différentes stratégies basées sur diverses théories et adaptées à divers contextes (PACI et al. 2018). Certaines sont triviales, comme les algorithmes de combinaison de XACML (OASIS 2013), d'autres

sont basées sur des théories mathématiques solides ou d'autres éléments tentant de mimer le comportement humain. Dans cette section, nous présentons d'abord un aperçu de la littérature sur ce sujet, puis nous détaillons les principaux facteurs pris en compte pour la résolution des conflits dans les réseaux sociaux.

3.2.1. Algorithmes de combinaison

Les premiers mécanismes de contrôle d'accès avec gestion de conflits pour les systèmes collaboratifs reposaient sur des stratégies prédéfinies. Par exemple, SHEN et al. 1992 définit un ensemble de règles à appliquer en cas de conflit, consistant à hiérarchiser les différentes décisions d'accès. Cette approche de priorisation est largement adoptée, en particulier en raison de la popularité des architectures XACML (OASIS 2013), qui mettent en œuvre des algorithmes de combinaison pour résoudre les conflits. Ces algorithmes sont bien connus. L'idée de ces algorithmes est de prendre en compte les différentes décisions des contrôleurs vis-à-vis d'une requête et de prioriser l'une des décisions selon l'algorithme choisi. Par exemple, l'algorithme Permit-override (POV) accorde l'accès si l'une des politiques considérées est évaluée à « Permit ». À l'inverse, l'algorithme Deny-override (DOV) accorde l'accès uniquement s'il n'y a aucune sous-politique évaluée à « Deny ».
On trouve également la stratégie Owner-override (OOV) où seule la préférence du propriétaire de la donnée est prise en compte, employé dans la plupart des contrôles d'accès traditionnels. Un autre algorithme très utilisé, employé par des réseaux sociaux très répandus comme Facebook, est le Host-override (HOV). Avec ce dernier, seule la politique de la personne hébergeant les données dans son espace personnel est considérée.

Le problème avec ces types de solutions est qu'à mesure que le nombre d'utilisateurs impliqués dans le conflit augmente, le résultat devient trop strict. D'autres stratégies sont explorées impliquant les utilisateurs afin d'avoir des solutions plus adaptées.

3.2.2. Stratégies de majorité et de consensus

Certains travaux s'appuient sur des stratégies basées sur la majorité ou le consensus. L'algorithme de consensus conduit, après plusieurs rounds de négociation, à une réponse uniforme de toutes les sous-politiques (consensus fort) ou au moins de certaines d'entre elles (consensus faible).

La majorité faible accorde l'accès s'il y a plus de « Permits » que de « Denies » dans les sous-politiques, tandis que l'algorithme de majorité forte accorde l'accès si plus de la moitié des sous-politiques permettent l'accès. Par exemple, LI, Q. WANG et al. 2009 propose un langage de combinaison des politiques pour étendre les stratégies fournies par XACML, avec des algorithmes de majorité et de consensus. Dans ALSHAREEF et al. 2020, ils utilisent la majorité faible pour résoudre les conflits restants après avoir appliqué le *most-specific-takes-precedence principle* (DE CAPITANI DI VIMERCATI et al. 2007). Ce principe stipule que, lorsqu'il y a des conflits entre différentes politiques de contrôle d'accès, la politique la plus spécifique doit prévaloir. Ainsi, si plusieurs règles

sont applicables à une même demande d'accès, celle qui est la plus précisément définie pour le contexte particulier de la demande sera utilisée pour prendre la décision finale. Le travail présenté dans CARMINATI et FERRARI 2011 introduit des politiques de sécurité collaboratives et utilise l'algorithme de combinaison de majorité (ou DOV) pour résoudre les conflits possibles soulevés par la politique collaborative. H. HU, AHN et JORGENSEN 2013 propose de calculer la décision d'un accès en utilisant un vote entre les différents contrôleurs. Ce vote se base sur les différentes préférences des contrôleurs d'un objet et les niveaux de confiance qu'ils partagent, ainsi que le poids de chaque contrôleur dans la décision. Ils ont aussi présenté une version de majorité et de consensus utilisant cette valeur plus granulaire.

Cependant, l'utilisation de consensus, de votes ou de protocoles de négociation présente des limitations lorsqu'elle est appliquée aux réseaux sociaux, car elle ajoute une couche de complexité pour les utilisateurs. Afin de réduire la complexité, certains travaux ont appliqué la théorie des jeux à ces techniques pour palier au problème.

3.2.3. Résolution basée sur la théorie des jeux

D'autres travaux ont appliqué certaines notions de théorie des jeux aux protocoles de négociation pour la résolution des conflits, comme par exemple l'équilibre de Nash. Dans H. HU, AHN, ZHAO et al. 2014, le protocole est basé sur la recherche de l'équilibre de Nash en amenant les contrôleurs à ajuster leurs préoccupations en matière de confidentialité pour maximiser leur fonction d'utilité. Pour ce faire, ils divisent les utilisateurs en audiences (une audience étant un ensemble d'utilisateurs ayant les mêmes décisions pour chaque contrôleur) et trouvent le meilleur ensemble d'audiences autorisées pour obtenir le meilleur compromis entre le risque pour la confidentialité et la perte de partage. La solution présentée dans RAJTMAJER et al. 2016b résout le consensus entre les différentes préférences des contrôleurs par le biais d'un jeu.

Cependant, ces approches ne reflètent pas correctement le comportement des utilisateurs, car ils considèrent le comportement humain comme totalement rationnel, comme mis en avant dans H. HU, AHN, ZHAO et al. 2014.

3.2.4. Résolution de conflits dans les photos

Lorsque le type de contenu à protéger est une image, des efforts considérables ont été consacrés au contrôle d'accès et à la gestion des conflits pour ce type de contenu. L'idée commune est de flouter les visages ou les parties des images faisant référence aux contrôleurs qui refusent l'accès, plutôt que de rejeter la requête d'accès.

Dans ILIA, POLAKIS et al. 2015, le mécanisme de contrôle d'accès floute les visages des contrôleurs qui ne doivent pas être visibles selon leurs préférences. Une approche plus fine est présentée dans VISHWAMITRA et al. 2017, où les auteurs proposent d'utiliser les Informations Personnelles Identifiables (PII ou données personnelles) pour déterminer quels utilisateurs apparaissent sur une image. En cas de conflit d'accès, les PII des contrôleurs dont l'accès n'a pas été accordé seront floutées sur l'image.

De cette manière, l'utilisateur n'aura accès qu'aux parties de l'image concernant les contrôleurs ayant eu l'accès ou ne permettant pas d'identifier les contrôleurs refusant l'accès.
L'automatisation de ces solutions est un défi majeur en particulier pour reconnaître les éléments à supprimer afin de réduire la charge pour l'utilisateur ou d'éviter les comportements malhonnêtes de l'utilisateur (ZHONG et al. 2018).

Le problème avec ce type de solution reste la difficulté de l'étendre à d'autres types de contenu que des images. Il existe d'autres stratégies qui permettent d'utiliser des propriétés intrinsèques aux objets et aux relations pour gérer les conflits sur tout type de contenu, ce sont les stratégies de résolution par seuil.

3.2.5. Résolution par seuil

D'autres stratégies sont basées sur des seuils pour résoudre un conflit dans certains travaux plus récents, en particulier pour résoudre les conflits de contrôle d'accès sur les types de systèmes de réseaux sociaux.

Ces stratégies résolvent les conflits de manière plus fine que les précédentes et utilisent un ensemble de valeurs, calculées autour de l'objet et du niveau de confiance des utilisateurs pour évaluer les accès. Par exemple, dans H. HU, AHN et JORGENSEN 2011, ils opposent, comme dans H. HU, AHN, ZHAO et al. 2014, le risque de confidentialité de l'accès à la perte de partage d'un élément pour une audience afin de résoudre les conflits de préférence des contrôleurs. Cependant, au lieu d'employer un protocole de négociation, les auteurs proposent un algorithme basé sur un seuil. Dans H. HU, AHN et JORGENSEN 2013, en plus du vote de décision, ils proposent un vote dépendant de la valeur de la sensibilité, calculé sur les différents niveaux de sensibilité de l'objet de la requête. Ces niveaux sont choisis par les contrôleurs, et utilisent ces valeurs comme seuil pour autoriser ou non l'accès. Dans SUCH et al. 2015, les auteurs proposent de calculer le niveau de sensibilité d'un élément, ainsi que l'importance du conflit pour les différents contrôleurs. Ces valeurs estimeront la volonté de changement de décision des contrôleurs, qui sera utilisée comme seuil pour modifier la décision des contrôleurs.

Ces méthodes fines induisent plus ou moins de surcharge pour les utilisateurs en raison de la manière dont les valeurs sont calculées. En effet, pour calculer ces valeurs, l'utilisateur doit généralement en définir manuellement d'autres, et trouver la bonne valeur n'est pas une chose triviale.

Une autre méthode, décrite dans PAIVA MELO MARIN 2015, repose sur le concept d'équité, défini comme l'équilibre entre le nombre de fois où les préférences d'un utilisateur ont été violées ou appliquées. Ainsi, les conflits ne sont pris en compte que parmi les personnes pour lesquelles l'équité n'est pas respectée. Les conflits restants sont ensuite résolus en utilisant les notions de risque et de sensibilité des objets. Cependant, un déséquilibre peut survenir entre le respect de la vie privée et l'équité lorsque les utilisateurs présentant un risque élevé pour leur vie privée ne sont pas pris en compte dans la résolution des conflits en raison du filtrage axé sur l'équité.

L'assistant ELVIRA (MOSCA et al. 2022) réutilise la notion d'audience, mais pour

résoudre un conflit, il calcule une fonction d'utilité qui déterminera quels publics pourraient accéder à l'élément. La fonction d'utilité est basée sur les valeurs fondamentales de Schwartz (S. H. SCHWARTZ 2012). Le but de ce travail est de trouver la meilleure audience pour tous les contrôleurs, par négociation sans surcharge pour les utilisateurs, du fait de la représentation de chaque utilisateur par un agent ELVIRA. Cependant, l'idée d'utiliser des ensembles d'utilisateurs au lieu d'utilisateurs entraîne une perte de granularité masquant les différents types de relations que les utilisateurs peuvent entretenir avec les contrôleurs.

Étant donné que les modèles basés sur un seuil sont ceux qui modélisent le mieux les interactions humaines, notre travail vise à maintenir cette souplesse tout en réduisant drastiquement la surcharge pour l'utilisateur. Notre approche permet d'alléger l'expérience utilisateur sans sacrifier la précision des interactions modélisées.

3.2.6. Facteurs de décision

En se concentrant sur les stratégies fines, en particulier celles basées sur des seuils, des notions similaires peuvent être trouvées dans différents articles. Les conflits sur des requêtes d'accès sont causés par des désaccords entre les contrôleurs sur certains demandeurs, en raison des différences de préférences de confidentialité ou de relations interpersonnelles. La première raison provient de la sensibilité des données, qui peut être considérée comme trop élevée pour certains contrôleurs par rapport aux demandeurs. La deuxième raison est plus liée à la confiance, car les contrôleurs peuvent ne pas avoir la même confiance envers les mêmes utilisateurs.

L'idée de la résolution des conflits en utilisant des mécanismes fins est de trouver un équilibre entre les préférences de confidentialité et le gain de partage dont les utilisateurs peuvent bénéficier. Pour quantifier les conflits, certaines notions fines ont été définies, certaines liées aux objets, d'autres aux utilisateurs.

Confiance Comme expliqué en Section 1.4.3, la confiance en quelqu'un peut être définie comme la croyance que cette personne agit sincèrement et ne causera aucun tort. Dans la gestion de conflit, cela représente le degré de confiance qu'un utilisateur a envers un autre (SUCH et al. 2015). Bien qu'il s'agisse d'un concept central dans les interactions humaines, il est difficile à quantifier automatiquement entre des inconnus.

Préférences de respect de la vie privée Les utilisateurs peuvent avoir différents points de vue sur la protection générale de leur vie privée. Les préférences de confidentialité donnent une estimation du niveau d'importance que la vie privée a pour un utilisateur en général. Elles sont souvent déduites des politiques par défaut de l'utilisateur (H. HU, AHN et JORGENSEN 2011).

Sensibilité d'un objet La sensibilité d'un objet est liée au concept de « privacy ». La sensibilité d'un objet pour un contrôleur donne une indication des dommages qui pourraient lui être causés si l'objet tombait entre de mauvaises mains. Plus un objet est sensible pour un utilisateur, plus les dommages en termes de vie privée peuvent être graves. La sensibilité mesure le degré de sensibilité d'un objet pour un utilisateur spécifique. Cette valeur peut être arbitraire, dépendant uniquement du point de vue personnel du contrôleur (H. HU, AHN et JORGENSEN 2011), ou elle peut être calculée comme dans SUCH et al. 2015 pour être plus rationnelle.

Importance du conflict L'importance d'un conflit mesure l'importance pour un utilisateur que le conflit soit résolu selon ses propres préférences. Plus le conflit est important pour un utilisateur, plus il souhaite que ses préférences soient suivies. Cette notion est utilisée avec la sensibilité de l'objet dans SUCH et al. 2015 pour calculer la volonté d'un utilisateur de changer sa décision. Si l'on compare aux modèles utilisant le risque de confidentialité par rapport à la perte de partage, l'importance du conflit sera élevée si le risque de confidentialité est trop important ou si la perte de partage est trop élevée.

Exposition de la vie privée vs gains à partager Le risque de confidentialité estime le risque d'être lésé si l'objet est accédé par un certain ensemble de demandeurs. La valeur est calculée en utilisant la sensibilité des objets, les préférences de confidentialité ainsi que la confiance envers le(s) demandeur(s). Elle est généralement utilisée pour équilibrer le bénéfice du partage comme dans H. HU, AHN et JORGENSEN 2011. Le bénéfice du partage mesure les avantages sociaux tels que le maintien des relations ou l'empathie mutuelle définie par KRASNOVA et al. 2010 qu'un utilisateur peut obtenir en donnant accès à l'objet demandé. Il est calculé en combinant la confiance envers le(s) demandeur(s) avec la sensibilité de l'objet et les préférences de confidentialité.

Cette notion de bénéfice a été prise en compte sous différents noms : certains travaux ne s'intéressent pas aux bénéfices de l'accès accordé, mais aux pertes si l'accès est refusé (H. HU, AHN et JORGENSEN 2011). D'autres, comme dans MOSCA et al. 2022, mesurent l'utilité de partager une donnée.

3.3. Défis de la gestion des conflits

Comme mentionné précédemment, proposer une gestion des conflits adaptée est une tâche complexe qui nécessite de répondre à différentes problématiques.

Un scénario simple illustrant ces principales problématiques est le suivant. Imaginons un petit réseau social qui fonctionne comme présenté en Section 1.2.1. Sur cette plateforme en ligne, Alice, Bob, Charlie, David, Emma, Finn et Gina partagent différentes relations, représentées en Figure 1.9.

Alice, utilisatrice du réseau social, charge une photo d'une fête sur son profil. Elle identifie sur la photo Bob et Charlie, deux de ses amis. Bob et Charlie apparaissant sur la photo peuvent voir leur vie privée impactée, leurs préférences doivent donc être

prises en compte. Ils font donc aussi partie des contrôleurs avec Alice. Alice souhaite partager sa photo avec ses amis et les amis de ses amis, mais Bob et Charlie ont des préférences différentes. Bob aimerait uniquement que ses amis aient accès à la photo alors que Charlie n'a pas de préférences, excepté pour Gina, avec qui il ne souhaite pas partager la photo. La politique par défaut, appliquée lorsque aucune autre politique n'est applicable à la requête, est de refuser l'accès.

Exemple 16 (Préférences des contrôleurs) *On formalise les préférences en politiques utilisateurs (Section 2.4.1) comme suit :*

$$
\begin{aligned}
pol_{(photo,Alice)} = \quad & (\text{resource_type} = photo \wedge \text{controller}(Alice, \text{resource_id})), \\
& \text{pov}\big((\text{areRelated}^2(Alice, \text{user_id}), 1), 0\big) \\
pol_{(photo,Bob)} = \quad & (\text{resource_type} = photo \wedge \text{controller}(Bob, \text{resource_id})), \\
& \text{pov}\big((\text{areRelated}(Alice, \text{user_id} \vee user_id = Charlie), 1), 0\big) \\
pol_{(photo,Charlie)} = \quad & (\text{resource_type} = photo \wedge \text{controller}(Charlie, \text{resource_id})), \\
& \text{pov}\big((\neg(\text{user_id} = Gina)), 1), 0\big)
\end{aligned}
$$

On écrira $Pol_c(u_r, o)$ *le résultat de l'évaluation de la politique* $pol_{(\text{resource_type},c)}$ *lorsque* u_r *est l'utilisateur effectuant la requête (le demandeur) et* resource_type *est le type de l'objet* o*. Pour tous les contrôleurs, si le demandeur ne satisfait pas la politique, l'accès sera refusé. Cela est illustré par l'algorithme de combinaison POV, qui combine la politique spécifiée par l'utilisateur avec la politique par défaut (ici* 0 *pour indiquer un refus).*

Certains utilisateurs satisferont toutes les préférences. Dans cet exemple, Emma satisfait toutes les préférences, donc une stratégie de résolution de conflits n'est pas nécessaire. Cependant, certains utilisateurs ne satisferont que certaines préférences mais pas d'autres, ce qui peut engendrer un conflit dans ce type de situation. Si l'on reprend notre exemple, Finn satisfait les politiques d'Alice et Charlie, mais pas celle de Bob. Gina ne satisfait que celle d'Alice. Plusieurs stratégies peuvent être prises en compte. On pourrait envisager de combiner ces préférences pour obtenir une réponse unique en appliquant des algorithmes courants tels que DOV. Cependant, ces solutions basiques ne reflètent pas le comportement humain. En effet, dans la vie réelle, l'un des contrôleurs peut changer d'avis, soit parce que l'utilisateur demandeur est digne de confiance, soit parce que ce contenu particulier ne causera pas de tort s'il est vu par cet utilisateur.

Une autre solution consiste à laisser les contrôleurs négocier pour parvenir à un accord sur une décision. Cependant, cela imposerait une charge supplémentaire, car ils devraient passer du temps à chaque fois qu'un nouvel utilisateur souhaite accéder à une donnée qu'ils contrôlent. Cela conduirait probablement l'utilisateur à abandonner la protection de sa vie privée ou à cesser de partager du contenu. De plus, en raison de la nature des systèmes de réseaux sociaux, les utilisateurs demandant un accès nécessitent une réponse immédiate, rendant cette solution difficile à mettre en

œuvre.

Il ressort de cet exemple que, pour développer une stratégie de gestion des conflits applicable à des systèmes collaboratifs de type réseau social, il est crucial de proposer une solution compréhensible par tous les utilisateurs, indépendamment de leurs parcours de vie. Cette solution doit être intuitive, alignée sur les comportements humains, réactive et demander peu d'efforts de la part des utilisateurs. Cette nécessité est formalisée dans les exigences présentées dans PACI et al. 2018 (cf. Section 2.2.2). En particulier, il est mis en avant qu'il est nécessaire, dans la gestion des accès des systèmes collaboratifs, de disposer d'une gestion des conflits adaptée sans ajouter de charge supplémentaire pour les utilisateurs.

Dans ce travail, nous mettons l'accent sur la proximité de la résolution avec le comportement humain, l'instantanéité de la résolution après l'apparition d'un conflit, et une charge limitée pour les utilisateurs. En effet, la problématique liée à une utilisation intuitive peut être traitée par des choix adaptés pour l'interface utilisateur, en la rendant compréhensible par des explications claires.

Afin d'illustrer le fonctionnement de notre méthode, cet exemple sera utilisé tout au long de la section. Le graphe de provenance représentant ce réseau social est dépeint en Figure 3.1,où l'on voit qu'Alice a chargé une photo et a identifié Bob et Charlie dessus, On observe qu'un premier accès a été accordé à Emma pour voir la photo.

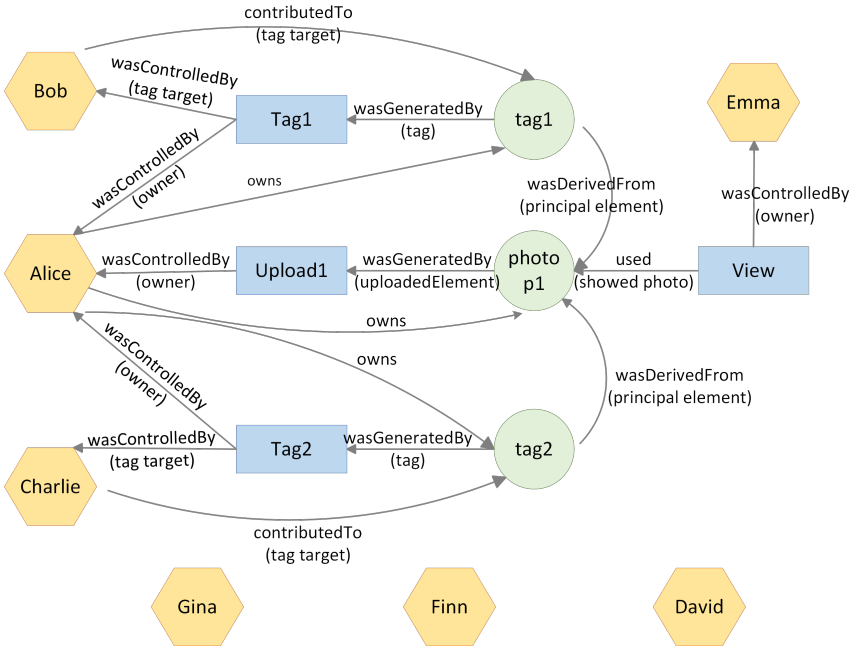


FIGURE 3.1. – Graphe de provenance utilisé dans le scénario illustrant les défis et notre modèle de gestion des conflits

3.4. Modèle de résolution de conflits basé sur un seuil

Les stratégies de contrôle d'accès les plus proches du comportement humain sont celles basées sur un seuil (cf. Section 3.2), calculé à l'aide des propriétés intrinsèques des objets. Elles imitent les interactions des utilisateurs dans la vie réelle. Cependant, elles entraînent souvent une charge supplémentaire pour les contrôleurs des objets, en raison de la saisie manuelle de certaines valeurs.

Notre stratégie présentée ici repose sur des notions similaires à celles d'autres articles, comme la sensibilité d'un objet et l'intérêt à le partager, mais utilise la modélisation du système via des graphes pour réduire la sollicitation des contrôleurs.

Le but principal des réseaux sociaux est de partager de l'information avec d'autres utilisateurs, ce qui peut avoir un impact positif sur les interactions sociales (KRASNOVA et al. 2010). Cependant de par la nature des données présentes dans ces systèmes, cela peut présenter des risques pour la vie privée des utilisateurs. Pour résoudre un conflit, il est donc nécessaire de faire le rapport entre les bénéfices du partage et les risques (Section 3.4.3).

Pour évaluer l'utilité et les avantages de ce partage, nous introduisons la notion d'**intérêt à partager un objet** (Section 3.4.2), qui se base à la fois sur la qualité de l'objet (*exactitude*) et son degré de diffusion. Pour quantifier le risque lié à la divulgation d'aspects privés à des utilisateurs de fiabilité variable, nous définissons la **sensibilité de l'objet** afin d'estimer le risque pour les contrôleurs (Section 3.4.1). Cette sensibilité dépend de la confiance accordée par chaque contrôleur aux personnes ayant accès à l'objet (*confiance dans sa communauté*) ainsi que de la sensibilité résiduelle des objets ayant contribué à celui de la requête (*sensibilité héritée*).

Dans cette section, nous reprenons ces deux concepts et les redéfinissons de manière plus objective et moins contraignante pour les utilisateurs, afin de proposer une méthode de résolution de conflits raffinée. Pour ce faire, nous utilisons le graphe de provenance et le graphe des relations enrichi avec la confiance entre les utilisateurs. Seules les relations actives lors de la résolution de conflits sont nécessaires et l'ordre total des actions du système n'est pas requis nous permettant d'omettre les horodatages pour simplifier le modèle.

Nous commençons par introduire les facteurs auxiliaires nécessaires au calcul de la sensibilité de l'objet (Section 3.4.1) et de l'intérêt à le partager (Section 3.4.2, puis nous présenterons la formule permettant de résoudre un conflit entre plusieurs contrôleurs pour cet objet et un demandeur (utilisateur effectuant la requête) en Section 3.4.3. Le Tableau 3.1 récapitule les différents facteurs utilisés.

Sensibilité de l'objet		
$\mathsf{Ctrust}(path, k, c, o) =$	$\underset{u \in Com_c(path,k)}{min}(\mathsf{Utrust}(c,u) \times Pol_c(u,o))$	Confiance de c en la communauté formée par les utilisateurs ayant accès à l'objet o
$\mathsf{HSens}(o) =$	$\underset{\substack{o' \neq o \mid \mathsf{wasDerivedFrom}^+(o,o') \\ c \in P \mid \mathsf{controller}(c,o'),}}{average} \mathsf{Ctrust}(path, k, o', c)$	Sensibilité héritée des objets ayant été nécessaires à la création de o
$\mathsf{Sens}(o) =$	$\frac{\mathsf{HSens}(o)}{\mathsf{HSens}^+(o)} \times \underset{c \in P \mid \mathsf{controller}(c,o)}{average} \mathsf{Ctrust}(path, k, c, o)$	Sensibilité de l'object o
Intérêt à partager l'objet avec un utilisateur		
$\mathsf{Acc}(o) =$	$\underset{c \in \mathscr{A} \mid \mathsf{controller}(c,o)}{average} \mathsf{CoAcc}(o,c)$	Exactitude de l'objet o
$\mathsf{Spread}(o,a)$	$\begin{cases} 1 \; if \; \underset{com \in Co_{u_r}}{max} \frac{ln(e+access(com,o))}{\lambda} < 1 \\ \underset{com \in Co_{u_r}}{max} \frac{ln(e+access(com,o))}{\lambda} \; otherwise \end{cases}$	Diffusion de l'objet o dans l'entourage de l'utilisateur a
$\mathsf{SInt}(o, u_r) =$	$\mathsf{Acc}(o) \times \frac{1}{\mathsf{Spread}(o,u_r)}$	Intérêt à partager un objet o sur la requête effectuée par l'utilisateur u_r
Résolution du conflit		
$\mathsf{R}(\mathsf{u_r}, \mathsf{o}) =$	$\frac{\alpha \times \mathsf{Sens}(o)}{\beta \times \mathsf{SInt}(o,u_r)}$	Résolution de conflits pour la requête effectuée par l'utilisateur u_r sur l'objet o

TABLEAU 3.1. – Récapitulatif des formules pour la résolution de conflits

3.4.1. Sensibilité

La sensibilité est déterminée par la confiance que les contrôleurs ont dans les utilisateurs qui peuvent accéder à l'objet. L'intuition derrière cette notion est la suivante. Si les contrôleurs n'autorisent dans leurs préférences que des utilisateurs en qui ils ont une confiance forte, cela signifie que les contrôleurs considèrent que l'information en question peut leur porter préjudice si elle tombe entre de mauvaises mains. La sensibilité de cet objet sera donc élevée. À l'inverse, si un utilisateur ayant une faible confiance de la part des contrôleurs est autorisé par certaines des politiques, la sensibilité sera plus faible. Ainsi, la sensibilité d'un objet est liée aux relations des contrôleurs : cette sensibilité peut diminuer lorsque les contrôleurs créent de nouvelles relations avec d'autres utilisateurs. Cependant, cette valeur est indépendante de la demande, car elle ne dépend pas du demandeur.

Pour quantifier la sensibilité, nous utilisons la confiance communautaire ainsi que la sensibilité résiduelle des objets précédents, appelée sensibilité héritée. Par exemple, pour déterminer la sensibilité d'une photo dans un réseau social, on considère la confiance communautaire des différents contrôleurs de la photo, mais aussi la confiance communautaire des contrôleurs de l'album contenant la photo (c'est-à-dire la sensibilité héritée). On formalise ces notions et leur calcul dans la suite de cette section.

3.4.1.1. Confiance de l'utilisateur dans une communauté

La confiance communautaire ou confiance dans sa communauté pour un objet permet d'estimer le niveau de risque associé à l'accès à cet objet s'il venait à tomber entre de mauvaises mains pour un contrôleur en question. Pour la quantifier, on utilise la confiance entre ce contrôleur et les utilisateurs qui sont autorisés par ses préférences. Le niveau de confiance entre utilisateurs est donné par l'algorithme TidalTrust (GOLBECK et al. 2005) (voir Section 1.4.3). Afin d'obtenir une estimation de la confiance pertinente (l'estimation de la confiance via une tierce personne n'est pas significative si la confiance entre les deux premiers utilisateurs est faible), nous fixons le paramètre max de l'algorithme TIdalTrust à 0,1.

En utilisant la confiance entre deux utilisateurs quelconques dans le système, nous définissons la confiance d'un contrôleur dans sa communauté centrée sur le contrôleur pour un objet, la confiance communautaire, inspirée de SUCH et al. 2015.

Définition 3 (Confiance communautaire) *La confiance d'un contrôleur c dans sa communauté centrée sur lui-même de rayon k pour un objet o est définie comme*

$$\mathsf{Ctrust} : \mathscr{R}el \times \mathbb{N} \times \mathscr{A} \times \mathscr{O} \rightarrow [0,1] \subset \mathbb{R}$$

, lorsque $\mathscr{R}el$ est l'ensemble de relations de la communauté. La formule pour calculer cette valeur est :

$$\mathsf{Ctrust}(path, k, c, o) = \underset{u \in Com_c(path,k)}{min} (\mathsf{Utrust}(c,u) \times Pol_c(u,o))$$

Les variables *path* et k permettent de réduire la complexité du calcul. En effet, on ne regarde qu'un sous ensemble d'utilisateurs. La relation *path* se réfère à la relation qui construit la communauté et elle est spécifique au système.

Comme cette valeur est utilisée pour calculer la sensibilité, prendre en compte la confiance de l'utilisateur le moins fiable reflète le niveau de confiance nécessaire pour accéder à l'objet. Dans le scénario présenté dans les exemples, nous considérons la relation *amis* comme chemin et fixons le rayon de la communauté centrée sur l'utilisateur à $k = 1$.

L'exemple ci-dessous illustre le calcul de la confiance communautaire pour les contrôleurs Alice, Bob et Charlie.

Exemple 17 *Considérons la photo $p1$ dans la Figure 3.1 , ses trois contrôleurs Alice, Bob et Charlie, ainsi que leurs politiques utilisateur présentées en Exemple 16.*

Le Tableau 3.2 récapitule les valeurs des confiances entre les différents utilisateurs de la Figure 1.9. Nous considérons ces valeurs pour calculer la confiance communautaire d'Alice, Bob et Charlie pour la photo $p1$.

Au sein d'une communauté, seule la confiance d'un contrôleur c envers les utilisateurs auxquels c permet d'accéder à l'objet est prise en compte. Par exemple, en considérant $path = \mathsf{areRelated}(c, u, friends)$ et $k = 1$, la confiance communautaire de Charlie n'utilisera pas la confiance que Charlie a en Gina, mais seulement celle qu'il a en Alice et Finn.

	A	**B**	**C**	**D**	**E**	**F**	**G**
A	x	0.7	0.7	0	0.5	0.8	0.4
B	0.8	x	0.7	0	0.5	0.8	0.4
C	0.7	0.7	x	0	0.5	0.8	0.4

TABLEAU 3.2. – Confiance entre les utilisateurs de la Figure 1.1

La confiance communautaire d'Alice dépend de la confiance qu'elle a en Bob et Charlie. Celle de Bob dépend de Alice et Emma. Selon le scénario de la Figure 1.9, nous obtenons

$$\mathsf{Ctrust}(\mathit{friends}, 1, \mathit{Alice}, p1) = \mathsf{Ctrust}(\mathit{friends}, 1, \mathit{Charlie}, p1) = 0,7$$

pour Alice et Charlie, et pour Bob :

$$\mathsf{Ctrust}(\mathit{friends}, 1, \mathit{Bob}, p1) = 0,5$$

3.4.1.2. Sensibilité d'un objet

La sensibilité d'un objet dépend de ces contrôleurs, mais non de l'utilisateur effectuant la demande d'accès. Intuitivement, si un contrôleur a une confiance faible en un utilisateur autorisé par ses préférences, la sensibilité ne doit pas être très élevée.

Pour l'estimer, nous utilisons la confiance communautaire pour chaque contrôleur de l'objet o en les combinant ainsi que les confiances communautaires des objets desquels dérive o. Ces dernières confiances sont combinées de façon à obtenir la sensibilité héritée. Par exemple on considère les confiances communautaires des contrôleurs d'un album pour calculer la sensibilité d'une photo.

Sensibilité héritée Pour garder une trace et pouvoir prendre en compte la sensibilité résiduelle d'un objet, nous calculons la sensibilité héritée. On la formalise de la façon suivante :

Définition 4 (Sensibilité héritée) *La sensibilité héritée d'un objet o, notée* $\mathsf{HSens} : \mathcal{O} \rightarrow [0,1] \subset \mathbb{R}$*, est définie comme une combinaison des confiances de la communauté centrée sur le contrôleur par rapport aux objets à partir desquels l'objet demandé o est dérivé.*

$$\mathsf{HSens}(o) = \underset{\substack{o' \neq o | \mathsf{wasDerivedFrom}^+(o,o') \\ c \in P | \mathsf{controller}(c,o'),}}{average} \mathsf{Ctrust}(\mathit{path}, k, o', c)$$

Nous définissons également une variation de la fonction sensibilité héritée, notée HSens^+ où o est inclus dans l'ensemble des objets dérivés. C'est à dire que la condition $o' \neq o$ n'est pas présente dans la définition.

Sensibilité globale La sensibilité d'un objet o est donnée en combinant les valeurs de la confiance dans la communauté des différents contrôleurs de o, ainsi que la sensibilité héritée de l'objet. L'objectif est de balancer la sensibilité de l'objet (obtenue en combinant les confiances de la communauté centrée sur le contrôleur) en utilisant le ratio de la sensibilité héritée avant et après l'existence de l'objet. Plus précisément, on obtient la sensibilité d'un objet o en pondérant la confiance communautaire combinée de tous les contrôleurs avec le ratio de la sensibilité héritée. Une sensibilité héritée très importante par rapport à celle de l'objet o augmentera cette dernière. Au contraire si elle est très basse, elle impactera à la baisse celle de o.

Définition 5 (Sensibilité) *La fonction sensibilité* $\mathsf{Sens} : \mathcal{O} \rightarrow [0,1] \subset \mathbb{R}$ *est définie comme*

$$\mathsf{Sens}(o) = \frac{\mathsf{HSens}(o)}{\mathsf{HSens}^{+}(o)} \times \underset{c \in P | \mathsf{controller}(c,o)}{average} \mathsf{Ctrust}(path, k, c, o)$$

Le ratio représente le résidu de sensibilité et sert de poids pour la confiance communautaire. Il est égal à 1 *lorsque l'objet o n'est dérivé d'aucun autre objet du système.*

L'exemple suivant donne les valeurs de la sensibilité pour la photo $p1$ dans notre scénario.

Exemple 18 *Calculons la sensibilité pour la photo p1 par rapport aux contrôleurs Alice, Bob et Charlie, dont la confiance dans leurs communautés est calculée dans l'Exemple 17.*

La photo $p1$ *n'est dérivée d'aucun objet. Le ratio* $\frac{\mathsf{HSens}(p1)}{\mathsf{HSens}^{+}(p1)}$ *est égal à 1. La sensibilité dépend alors uniquement des confiances communautaires des contrôleurs.*

$$\begin{aligned} \mathsf{Sens}(p1) &= average\,(\mathsf{Ctrust}(friends, 1, Alice, p1), \mathsf{Ctrust}(friends, 1, Bob, p1), \\ &\quad \mathsf{Ctrust}(friends, 1, Charlie, p1)) \\ &= average\,(0.7, 0.5, 0.7) \\ &= 0.63 \end{aligned}$$

Puisque la valeur de sensibilité se situe dans $[0,1]$*, la sensibilité de p1 est moyennement élevée. Intuitivement, cela signifie que laisser l'accès à la photo peut être préjudiciable si elle tombe entre de mauvaises mains.*

3.4.2. Intérêt à partager

L'objectif principal des utilisateurs de réseaux sociaux en ligne est le partage d'informations au sein de leur réseau de relations. En s'inspirant des concepts présentés dans les articles H. Hu, Ahn et Jorgensen 2011 and Mosca et al. 2022 nous définissons l'intérêt à partager, comme une fonction permettant de quantifier les avantages qu'un contrôleur a s'il partage une information avec un autre utilisateur.

Dans notre formalisation, cette fonction dépend de l'utilisateur demandant l'accès à l'information (c'est-à-dire le demandeur) et de l'information elle-même (c'est-à-dire

l'objet demandé). Nous supposons que le bénéfice augmente lorsque l'information est fiable et digne de confiance (*exactitude*) et lorsque l'information n'a pas été largement diffusée (*propagation communautaire*). Ces deux notions sont calculées du point de vue des contrôleurs.

Nous définissons formellement ci-dessous l'exactitude, le propagation communautaire ainsi que l'intérêt à partager.

3.4.2.1. Exactitude d'un objet

L'exactitude d'un objet donne une estimation de la qualité de l'information qu'il possède. Elle est calculée à partir de la confiance mutuelle des contrôleurs de l'objet. Intuitivement, plus les contrôleurs se font mutuellement confiance, plus la fiabilité de l'objet (du point de vue des contrôleurs) est élevée.

La confiance dans un utilisateur peut être liée à la fiabilité de l'information qu'il partage. Autrement dit, suivant l'idée que plus un contrôleur est digne de confiance, plus l'information qu'il partage est fiable.

Nous définissons ainsi l'exactitude d'un objet o pour un contrôleur c comme la combinaison des confiances mutuelles entre c et les contrôleurs de o ainsi que les objets à partir desquels o est dérivé.

Définition 6 (Exactitude pour un contrôleur) *L'exactitude d'un objet o pour un contrôleur c, notée* $\mathsf{Acc}_c : \mathscr{O} \times \mathscr{A} \rightarrow [0,1] \subset \mathbb{R}$*, est définie comme la confiance minimale de c dans les contrôleurs des objets à partir desquels o est dérivé :*

$$\mathsf{CoAcc}(o,c) = \underset{\substack{o' | \mathsf{wasDerivedFrom}^+(o,o') \\ c' \in \mathscr{A} | \mathsf{controller}(c',o')}}{min} \mathsf{Utrust}(c,c')$$

En d'autres mots, l'exactitude pour un contrôleur c est la confiance minimale qu'il a dans les contrôleurs des objets à partir desquels o est dérivé. Cette confiance représente le contrôleur le plus susceptible de partager des informations de mauvaise qualité selon c.

L'exactitude d'un objet est obtenue en combinant les valeurs individuelles d'exactitude de chaque contrôleur.

Définition 7 (Exactitude globale) *L'exactitude globale d'un objet, notée* $\mathsf{Acc} : \mathscr{O} \rightarrow [0,1] \subset \mathbb{R}$*, est définie comme :*

$$\mathsf{Acc}(o) = \underset{c \in \mathscr{A} | \mathsf{controller}(c,o)}{average} \mathsf{CoAcc}(o,c)$$

L'exemple suivant illustre le calcul de l'exactitude dans notre scénario.

Exemple 19 *Nous souhaitons déterminer l'exactitude de la photo p1, comme illustré dans la Figure 3.1. La dépendance* $\mathsf{wasDerivedFrom}^+(p1,o')$ *ne retourne que p1. En effet, l'objet p1 n'a pas d'objets dont il est dérivé.*

Les contrôleurs à considérer sont Alice, Bob et Charlie. En utilisant les valeurs de confiance de l'Exemple 1, l'exactitude de p1 pour les contrôleurs Alice, Bob, Charlie, est

$$\mathsf{CoAcc}(p1, Alice) = \min(0.7, 0.7) = 0.7$$
$$\mathsf{CoAcc}(p1, Bon) = \min(0.8, 0.7) = 0.7$$
$$\mathsf{CoAcc}(p1, Charlie) = \min(0.7, 0.7) = 0.7$$

On obtient alors l'exactitude pour p1 est

$$\mathsf{Acc}(p1) = average(0.7, 0.7, 0.7) = 0.7$$

3.4.2.2. Propagation communautaire

Les accès précédents à un objet peuvent influencer l'intérêt à accorder davantage d'accès à celui-ci. Par exemple, si les membres d'un groupe ont déjà largement accédé à une information, il peut être moins significatif, en termes de partage de l'information, de fournir un accès à un nouvel utilisateur appartenant au même groupe.
Cela est dû au fait que l'information est déjà largement diffusée au sein de la communauté, en raison des accès précédents par les membres du groupe. On pourrait argumenter que diminuer l'intérêt à accorder l'accès à mesure que l'information se répand dans la communauté est une approche contre-intuitive. Cependant, du point de vue du contrôleur, si une information a déjà été largement accédée dans cette communauté, il devient moins crucial de la partager davantage avec les membres de la communauté.

Pour quantifier la notion de diffusion de l'information au sein d'une communauté, nous récupérons les accès *par communauté* qui ont été réalisés sur l'objet demandé dans un intervalle de temps (spécifique à chaque système). Ces accès peuvent être récupérés directement en inspectant le graphe de provenance. Cependant, il est plus efficace, en termes de complexité de calcul, de consulter directement les journaux système correspondant à la fenêtre temporelle prédéfinie.

Pour chaque demande d'accès, nous identifions la communauté contenant le demandeur qui a enregistré le plus grand nombre d'accès. Pour déterminer ces communautés, nous examinons toutes les relations dans le graphe social à une distance maximale $maxDis$ du demandeur. Autrement dit, nous considérons toutes les relations à une distance proche pour maintenir la notion de communauté. Les accès dans l'intervalle de temps prédéfini sont regroupés par communauté et le plus grand nombre d'accès est choisi.

Il est important de noter que la notion de propagation communautaire ne dépend pas du contrôleur de l'objet, mais de l'objet lui-même et du demandeur.

Définition 8 (Propagation communautaire de l'objet) *La propagation communautaire,* $\mathsf{Spread} : \mathscr{O} \times \mathscr{A} \to \mathbb{R}$*, est définie par*

$$\mathsf{Spread}(o, u_r) = \begin{cases} 1 \; si \; \max\limits_{com \in Co_{u_r}} \frac{ln(e+access(com,o))}{\lambda} < 1 \\ \max\limits_{com \in Co_{u_r}} \frac{ln(e+access(com,o))}{\lambda} \; autrement \end{cases}$$

Nous notons u_r l'utilisateur faisant la demande et par $Co_{u_r} = \{com \mid u_r \in com\}$ l'ensemble des communautés auxquelles le demandeur appartient. Le terme $accesscomo$ désigne le nombre d'accès (accordés) effectués par les membres de la communauté com à l'objet o.
Le coefficient λ est une constante positive (qui peut dépendre du système) introduite pour contrôler l'impact de propagation communautaire dans le calcul de l'intérêt à partager (voir Définition 9). La propagation communautaire est calculée en prenant le logarithme du nombre d'accès. Le nombre d'Euler e permet d'obtenir $\mathsf{Spread}(o, u_r) > 1$ *lorsque* $\lambda = 1$.

Selon la définition ci-dessus, plus un objet a été accédé par une communauté à laquelle appartient le demandeur, plus sa valeur de Spread est élevée. L'utilisation d'un facteur logarithmique ainsi que du facteur λ atténue l'augmentation de la valeur de diffusion, ce qui est essentiel pour calculer un intérêt à partager raisonnable.

Exemple 20 (Propagation communautaire) *Considérons une requête d'accès provenant de Finn qui demande l'accès à la photo $p1$ et $maxDis = 1$. Les communautés auxquelles Finn appartient, selon le graphe des relations en Figure 1.9, sont*

$$Co_{Finn} = \{Com_{Finn}(\mathsf{isRelated}(Finn, x, friends), 1), \\ Com(\mathsf{isRelated}(Finn, y, \mathtt{group:University}), 1)\}$$

Récemment, selon la Figure 3.1, seule l'utilisatrice Emma, dans la communauté formée par le groupe « University » *a eu accès à $p1$, on a alors $accesscomp1 = 1$. Nous choisissons donc cette communauté, car c'est celle ayant le plus grand nombre d'accès. Avec la constante λ fixée à $\lambda = 1.7$ (voir Section 3.5.1 pour les détails), nous obtenons alors* $\mathsf{Spread}(p1, Finn) = 1$, *puisque* $\frac{\ln(e+1)}{1.7} < 1$. *Étant donné que le nombre d'accès est très faible, la propagation de la photo $p1$ n'affectera pas la valeur de l'intérêt à partager (voir Exemple 21).*

On remarque que Gina, l'autre utilisateur où la demande d'accès génère un conflit entre les préférences des utilisateurs, appartient aussi au groupe « University ». *La valeur de l'intérêt à partager est la même que pour Finn.*

3.4.2.3. Intérêt à partager

En utilisant les notions définies précédemment, nous pouvons formaliser l'intérêt des contrôleurs à partager un objet avec un utilisateur.

Définition 9 (Intérêt à partager) *La fonction calculant l'intérêt à partager,* $\mathsf{SInt} : \mathscr{O} \times \mathscr{A} \to \mathbb{R}$, *est définie par :*

$$\mathsf{SInt}(o, u_r) = \mathsf{Acc}(o) \times \frac{1}{\mathsf{Spread}(o, acc)}$$

L'intérêt à partager dépend de la qualité d'un objet et du niveau de propagation dans le voisinage de l'utilisateur effectuant la requête d'accès. Selon la définition, un

demandeur qui rejoint une communauté où une information a été largement diffusée, a une valeur d'intérêt à partager plus basse par rapport à celui qu'il avait en tant qu'externe à cette communauté.

En suivant notre scénario, les valeurs de l'intérêt à partager pour Finn et Gina sont calculées dans l'exemple ci-dessous.

Exemple 21 (Intérêt à partager) *L'intérêt à partager la photo p1 avec l'utilisateur Finn est égale à*

$$\mathsf{SInt}(p1, F) = \mathsf{Acc}(p1) \times \frac{1}{\mathsf{Spread}(p1, Finn)} = 0.7 \times \frac{1}{1} = 0.7$$

en utilisant les valeurs calculées dans les Exemples 19 et 20.
Si Gina émet la même requête que Finn, le calcul de l'intérêt à partager donne le même résultat.

La valeur de l'intérêt à partager des contrôleurs avec Finn ou Gina est élevée, ce qui signifie que les contrôleurs bénéficieront du partage de la photo avec eux.

3.4.3. Résolution du conflit

Pour résoudre un conflit, il est nécessaire de comparer les valeurs obtenues pour l'objet et le demandeur (sensibilité et intérêt de partager) et de déterminer quelle notion a le plus de poids dans cette demande. Pour calculer cette décision d'accès collective lorsque les préférences des contrôleurs diffèrent, nous proposons une stratégie de résolution de conflits qui utilise les facteurs de prise de décision définis précédemment. L'objectif de la sensibilité d'un objet et de l'intérêt de partager est de quantifier un conflit parmi les contrôleurs pour un demandeur et de choisir la « meilleure » résolution du conflit en termes d'indicateurs. Intuitivement, si la sensibilité de l'objet est élevée et l'intérêt de partager est faible, la résolution du conflit devrait aboutir à un refus d'accès. Dans le cas contraire, l'accès à l'objet devrait être accordé.

Nous les comparons en faisant un ratio entre les deux valeurs. Si le résultat est supérieur à 1, la sensibilité est trop importante par rapport à l'intérêt de partager. Lorsque le ratio est inférieur à 1, l'intérêt de partager est plus important, l'accès doit donc être permis.
Le problème du ratio est qu'il ne prend pas en compte la confiance que les contrôleurs accordent au demandeur, ce qui pourrait fournir une indication supplémentaire sur le risque ou le bénéfice associé à l'accès. Par exemple, si nous ne prenons pas en compte ce dernier aspect, Finn et Gina auront tous deux accès à la photo, indépendamment de la confiance que les contrôleurs ont en eux.

Pour résoudre ce problème, nous équilibrons la sensibilité et l'intérêt à partager avec deux coefficients, notés α et β, respectivement.

Ci-dessous, nous définissons d'abord formellement ces deux coefficients de crédibilité, puis donnons la formule permettant de résoudre les conflits.

3.4.3.1. Coefficients de crédibilité

Les coefficients de crédibilité pondèrent les notions précédemment définies, en explicitant les caractéristiques de la relation entre les contrôleurs et le demandeur. Le coefficient α concerne les contrôleurs qui refusent l'accès, tandis que β concerne ceux qui l'accordent. Ces coefficients permettent une évaluation et une gestion plus précises des permissions d'accès, améliorant ainsi la robustesse du système.

Dans notre modèle, α et β sont définis en fonction des niveaux de confiance des contrôleurs envers le demandeur. Cependant, ces coefficients peuvent également intégrer d'autres caractéristiques, comme l'importance du conflit, tel que défini dans SUCH et al. 2015.

Coefficient α Le coefficient α représente la confiance que les contrôleurs ont dans le demandeur, dans le contexte où ils lui refusent l'accès à l'objet demandé selon leurs politiques utilisateurs, c'est-à-dire $Pol_c(u_r, o) = 0$. L'idée sous-jacente à ce coefficient est que plus les contrôleurs se méfient d'un utilisateur, plus le risque de préjudice par cet utilisateur est élevé. Par conséquent, dans ce cas, il est nécessaire d'augmenter la valeur de sensibilité. À l'inverse, si les contrôleurs ont une grande confiance envers le demandeur, le risque de préjudice reste faible, même lorsque l'objet est sensible. En suivant cette idée, le coefficient α minimise la sensibilité de l'objet.

Nous définissons donc α comme suit :

$$\alpha = 2 - \underset{c \in P | \mathsf{controller}(c,o)}{min} (\mathsf{Utrust}(c, u_r)) \times (1 - Pol_c(u_r, o)))$$

Coefficient β Le coefficient β est utilisé pour équilibrer l'intérêt à partager et représente la confiance que les contrôleurs accordent au demandeur, en supposant qu'ils lui accordent l'accès à l'objet conformément à leurs politiques utilisateur. Les contrôleurs donnant accès voudront encore plus que leur choix soit respecté lorsque leur confiance envers le demandeur est élevée. Si la confiance est faible, ils sont moins disposés à appuyer leur choix. Par conséquent, le coefficient β est défini comme :

$$\beta = 1 + \underset{c \in P | \mathsf{controller}(c,o)}{max} (\mathsf{Utrust}(c, u_r) \times Pol_c(u_r, o))$$

En raison de la définition de la fonction Utrust, les valeurs de α et β se situent entre 1 et 2.

Autres facteurs de décisions Dans ce modèle, nous avons choisi de baser les coefficients de crédibilité sur la confiance des utilisateurs, car cette notion est essentielle dans la gestion des relations entre les utilisateurs : α représente la confiance des utilisateurs qui refusent l'accès, tandis que β représente celle des utilisateurs dont les préférences accordent l'accès.
Ces coefficients pourraient être remplacés par la notion d'importance du conflit

présentée dans l'article SUCH et al. 2015. Dans ce cas, α serait défini comme une combinaison de l'importance du conflit pour les contrôleurs refusant l'accès, et β comme une combinaison de l'importance du conflit pour ceux accordant l'accès.
Ces coefficients peuvent également être liés au nombre de contrôleurs accordant ou refusant l'accès, permettant ainsi d'implémenter une atténuation basée sur une stratégie de majorité.

3.4.3.2. Résolution du conflit

Comme évoqué précédemment, la résolution du conflit est obtenue en faisant le ratio entre la sensibilité et l'intérêt à partager, pondérés respectivement par α et β. La formule est donnée dans la définition suivante.

Définition 10 (Résolution du conflit) *La formule finale de résolution de conflits est donnée par :*

$$\mathsf{R}(\mathsf{u_r}, \mathsf{o}) = \frac{\alpha \times \mathsf{Sens}(\mathsf{o})}{\beta \times \mathsf{SInt}(\mathsf{o}, \mathsf{u_r})}$$

avec α et β les coefficients de crédibilité définis au dessus.

Lorsque $\mathsf{R}(\mathsf{u_r}, \mathsf{o}) \geq 1$, la sensibilité de l'objet est élevée par rapport à l'intérêt à partager pour autoriser l'accès, donc la décision d'accès sera *refusée*. Au contraire, lorsque $\mathsf{R}(\mathsf{u_r}, \mathsf{o}) < 1$, l'intérêt à partager est plus importante que la sensibilité, donc la demande d'accès sera *autorisée*.

Afin d'illustrer son utilisation, revenons au scénario présenté tout au long des exemples de la Section 3.4

Exemple 22 *Considérons les requêtes d'accès réalisées par Finn et Gina sur la photo* p1. *La confiance que les contrôleurs accordent à Finn est de* 0,8 *(Exemple 1). En suivant les préférences des contrôleurs présentées en Exemple 16, Bob est le seul contrôleur qui n'autorise pas Finn dans sa politique. Les coefficients α et β ont pour valeur :*

$\alpha = 2 - 0.8 = 1.2$, *and* $\beta = 1 + 0.8 = 1.8$. *La résolution de conflits accorde l'accès à Finn, car*

$$\mathsf{R}(\mathsf{Finn}, \mathsf{p1}) = \frac{1.2 \times 0.63}{1.8 \times 0.7} = \frac{0.756}{1.26} = 0.6 < 1$$

Cela correspond à l'intuition, puisque la SInt *est plus élevée que la valeur de* Sens *et que tous les contrôleurs ont une confiance élevée en Finn. En raison de cette confiance élevée de tous les contrôleurs en Finn, la différence entre la sensibilité pondérée et l'intérêt à partager a augmenté, renforçant l'idée d'autoriser l'accès.*

Considérons maintenant que Gina a réalisé la requête, les valeurs SInt *et* Sens *sont les même que pour Finn. La différence entre Finn et Gina dans la résolution de conflits est la confiance que les contrôleurs déposent en eux. Seuls les coefficients α et β peuvent*

modifier l'issue du conflit. La confiance que les contrôleurs accordent à Gina est moyennement basse, $0,4$. *Le calcul de la résolution du conflit est le suivant :*

$$\mathsf{R}(\mathsf{Gina},\mathsf{p1}) = \frac{1.6 \times 0.63}{1.4 \times 0.7} = \frac{1.008}{0.98} = 1.03 > 1$$

Puisque $\mathsf{R}(\mathsf{Gina},\mathsf{p1}) > 1$, *l'accès de Gina à la photo* $p1$ *sera refusé. Ce résultat s'explique par une confiance moyennement basse envers le demandeur et de la proximité entre* SInt *et* Sens. *Ainsi, le poids du coefficient* α *conduit à une réponse négative pour Gina. Si les deux valeurs avaient une différence significative, alors* α *et* β *n'affecteraient pas l'issue du conflit.*

On peut observer que les résultats sont conformes à l'intuition. Lorsque le niveau de confiance dans le demandeur est élevé parmi tous les contrôleurs, l'accès lui est accordé.

Lorsque les valeurs de sensibilité et intérêt à partager présentent une grande différence, l'impact des coefficients α et β n'est pas suffisamment important pour modifier l'issue.

3.5. Validation du modèle

Pour valider notre méthode de résolution de conflits, nous avons étudié les variations des différents facteurs afin de vérifier que notre approche restait proche du comportement humain en matière de résolution de conflits. Nous avons également corroboré ces résultats en comparant notre modèle à d'autres modèles basés sur des seuils, montrant qu'il peut les englober sans surcharger l'utilisateur. Enfin, nous avons développé un prototype générant des réseaux sociaux et des requêtes, ce qui nous a permis d'appliquer notre modèle et de le comparer à des stratégies de combinaison classiques et des stratégies plus fines sur de nombreux exemples.

3.5.1. Variations des facteurs

Dans notre approche, les facteurs considérés reposent sur la confiance entre les utilisateurs ainsi que sur leur relation avec l'objet demandé. Il est raisonnable de supposer que plus nous faisons confiance à quelqu'un, moins le risque de subir des dommages de sa part est élevé. De plus, un individu est plus enclin à soutenir et à insister pour permettre à quelqu'un d'accéder à une information si la confiance en cette personne est élevée. Ce sont ces hypothèses que nous vérifions pour valider notre définition des facteurs.

Lorsque la sensibilité (c'est-à-dire le potentiel de préjudice) augmente, les chances que l'accès à l'objet soit accordé diminuent : soit la sensibilité devient supérieure à l'intérêt de partage, soit la différence entre les deux diminue. La Figure 3.2a, basée sur les exemples de la section précédente, montre aussi que la sensibilité héritée par

un objet impacte son degré de sensibilité. Si l'on pondère la sensibilité avec α, il est d'autant plus probable que l'on obtienne une valeur supérieure à 1 lors de l'application de la formule de résolution.

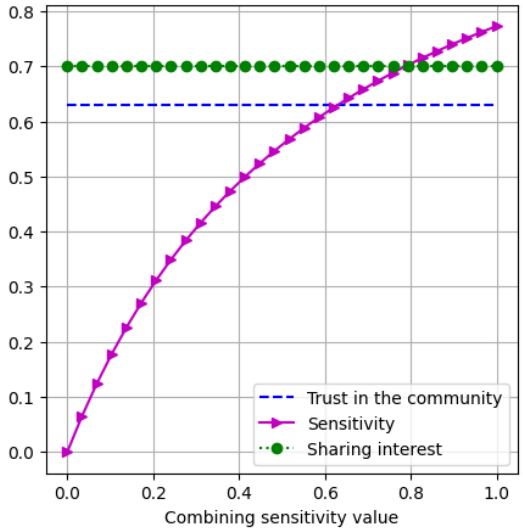
À l'inverse, lorsque l'intérêt de partager (c'est-à-dire l'utilité de permettre l'accès) augmente, le résultat de la résolution de conflits est plus susceptible d'être positif (car l'intérêt à partager devient plus grand que la sensibilité, avec ou sans β). L'intérêt à partager, dépendant de l'exactitude et de la propagation communautaire, augmente si la première valeur augmente et diminue lorsque la deuxième augmente (Figures 3.2b et 3.2c).

Il est important de se rappeler que, bien que cela puisse sembler contre-intuitif (cf. Figure 3.2c), plus un objet a été diffusé, moins il est important pour les contrôleurs de donner accès à cet objet : l'information a déjà été diffusée, donc l'intérêt à partager cet objet est réduit. La fonction de diffusion au sein de la communauté est régulée par le coefficient λ, qui a été fixé dans nos scénarios à 1.7. Avec cette valeur, la diffusion au sein de la communauté a un impact sur l'intérêt de partage sans annuler totalement la valeur de l'exactitude. Comme le montre la Figure 3.2d, la grande confiance des contrôleurs dans un demandeur a deux conséquences : d'une part, le risque de préjudice étant réduit, le coefficient α diminue, la valeur de la sensibilité pondérée par α est faiblement impactée, augmentant légèrement. D'autre part, la valeur de l'intérêt de partage, lorsqu'elle est pondérée par le coefficient β, est élevée.

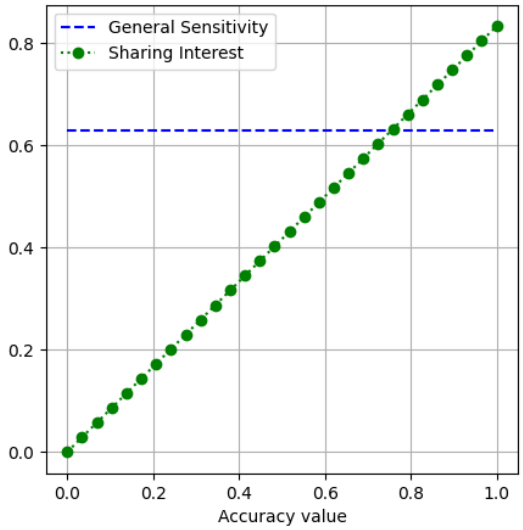
Les variations montrées dans les graphiques de la Figure 3.2 valident nos hypothèses.

3.5.2. Analyse comparative par rapport aux modèles basés sur le seuil

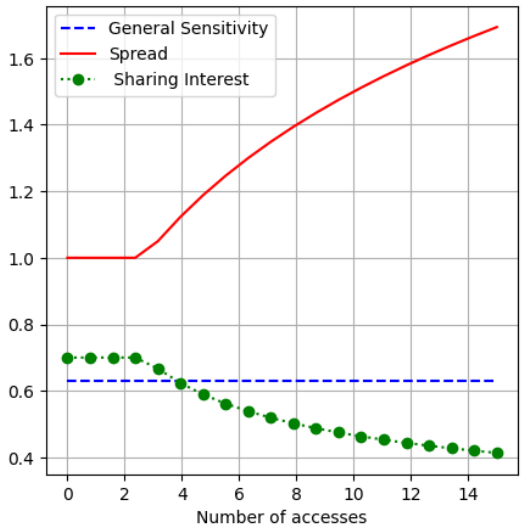
Comme présenté dans la Section 3.2, l'idée la plus courante dans les modèles basés sur des seuils est de mettre en équilibre les risques pour la vie privée avec les avantages du partage comme abordé par H. Hu, Ahn et Jorgensen 2011 ou encore Mosca et al. 2022. En général, les risques pour la vie privée sont calculés en utilisant la sensibilité d'un objet, la confiance dans le demandeur et les préférences en matière de confidentialité des contrôleurs. Les deux premiers facteurs sont pris en compte dans notre formule par la sensibilité et le coefficient α. Les préférences en matière de vie privée sont utilisées pour équilibrer la valeur de la sensibilité dans H. Hu, Ahn et Jorgensen 2011, car dans les modèles où cette valeur est directement fournie par les contrôleurs, elle peut être exagérée pour induire une décision en leur faveur. Dans notre cas, la valeur est calculée à partir de la plateforme de réseau social, donc aucune atténuation n'est nécessaire. L'avantage du partage est aussi mesuré dans ce travail-là en calculant un dual du risque pour la vie privée. Dans notre cas, nous utilisons différentes valeurs pour fournir une quantification, comme la notion de qualité de l'objet ou sa propagation. Le modèle présenté dans Mosca et al. 2022 repose également sur cet équilibre, tout comme notre approche. Cependant, leur calcul d'utilité se base sur des audiences, c'est-à-dire des groupes d'utilisateurs partageant les mêmes droits sur un objet donné, ce qui rend une comparaison avec



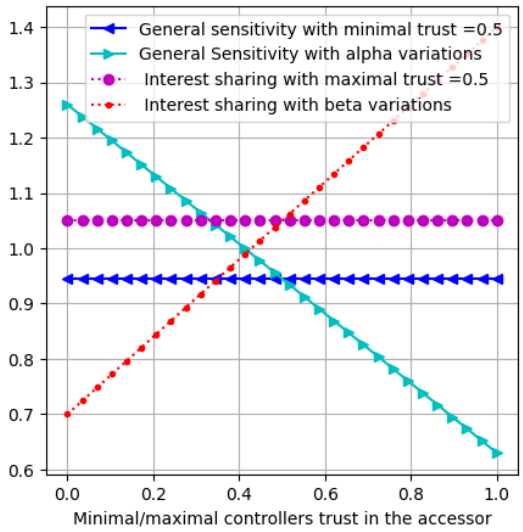
(a) Impact de la sensibilité héritée sur la sensibilité



(b) Impact de l'exactitude sur l'intérêt à partager



(c) Impact de la propagation communautaire sur l'intérêt à partager



(d) Influence des coefficients α et β

FIGURE 3.2. – Variations de la sensibilité et de lintérêt à partager en fonction de leur composantes

notre modèle compliqué.

Dans SUCH et al. 2015, les auteurs basent leur stratégie sur la volonté de changement. La sensibilité de l'objet est calculée de façon similaire à notre facteur. Ce modèle mesure l'importance d'un conflit, soit en fonction de la confiance dans la communauté, soit en fonction de la sensibilité de l'objet combinée à la confiance dans le demandeur. Plus la différence entre l'une des deux premières valeurs et la dernière est grande, plus

l'importance du conflit est élevée. Notre approche est différente, mais notre modèle tend à coïncider avec le résultat [1]. Par exemple, en suivant leur scénario, notre modèle conduit aux mêmes décisions d'accès. En effet, dans leur exemple basé sur un réseau social de 6 utilisateurs, Alice et Bob doivent se mettre d'accord sur les utilisateurs pouvant voir une photo où ils apparaissent tous les deux. Un conflit surgit pour deux des utilisateurs : Eve et Franck sont uniquement autorisés par les préférences de Bob. La confiance d'Alice en Eve et Franck est très basse, mais Bob a une confiance très élevée en Eve. Avec notre calcul les valeurs de sensibilité et d'intérêt à partager sont moyennes, donnant une forte importance aux coefficients α et β. Ceci signifie que Eve obtient une réponse positive (dû au β très élevé) et que l'accès de Franck est refusé (dû à un α très important et un β moyen), même pour des petites valeurs de propagation communautaire. En faisant varier le paramètre λ, nous pouvons amener le modèle à obtenir le même résultat pour des différentes valeurs de propagation communautaire. En outre, en modifiant la définition des coefficients de crédibilité, nous pouvons adapter α et β pour tenir compte de l'importance du conflit et ainsi établir une équivalence entre les deux modèles. En effet, plus le conflit est important pour les contrôleurs, plus α ou β (selon la décision liée aux préférences des contrôleurs) viendront majorer la sensibilité ou l'intérêt à partager, ce qui est conforme à leur approche.

3.5.3. Prototype

Dans le but de valider notre modèle vis-à-vis d'autres stratégies utilisées pour résoudre la gestion de conflits, nous avons crée un prototype implémentant notre solution, ainsi que d'autres solutions largement répandues ou basées sur des notions proches. Pour faire cela, nous avons adapté le générateur de graphes et de politiques utilisateur utilisés dans la validation du contrôle d'accès collaboratif (Section 2.5), et avons utiliser Python pour implémenter les fonctions de résolution de conflits. De plus, dû à la nécessité de répondre aux requêtes de façon instantanée dans le contexte des systèmes collaboratifs centrés sur la communauté, nous utilisons ce prototype pour vérifier la faisabilité de notre solution en termes de performances.

Dans cette section, nous présentons d'abord l'architecture globale de notre prototype, puis nous discutons des résultats obtenus en comparaison à d'autres algorithmes de combinaison utilisés dans les réseaux sociaux. Nous donnons une analyse comparative entre notre modèle et celui présenté dans SUCH et al. 2015. Finalement, nous présentons les résultats liés à la performance.

3.5.3.1. Architecture

Pour valider notre approche, nous avons adapté les générateurs de graphes simulant un réseau social ainsi que celui de politiques utilisateur, comme présenté en 2.5. Tout d'abord, nous avons mis de côté l'extension temporelle, car dans notre gestion de conflit les horodatages ne sont pas utilisés.

1. Une analyse plus approfondie est donnée en Section 3.5.3.3

Le générateur de graphes a été étendu avec des prédicats représentant la confiance. Pour chaque relation areRelated entre deux utilisateurs, deux prédicats $trust$ sont ajoutés. Le niveau de confiance n'étant pas réciproque, une valeur de confiance pour chaque composante isRelated est nécessaire. Cette valeur est définie aléatoirement dans l'intervalle $]0,1]$. Les relations utilisateurs continuent à être générées en suivant le modèle « friend to friend » (LEVENS et al. 2022). Les probabilités utilisées sont $p = 1$ (probabilité avec les nœuds déjà existants) et $q = \frac{1}{2}$ (probabilité avec un voisin du nœud déjà existant). Le contenu social de chaque utilisateur (photos, publications) est intégré au graphe de provenance de la même manière que décrit précédemment

Concernant les politiques utilisateurs, en fonction des options Facebook, le générateur de politiques crée pour chaque utilisateur un fichier contenant ses préférences pour les actions voir, poster, partager, commenter pour les différents types d'objets (photo, post, album, profil). Les préférences peuvent être de personne à tout le monde via n'importe quelle combinaison des arêtes du graphique social.

Le programme principal génère des requêtes basées sur les graphiques générés précédemment et exécute notre solution de résolution de conflits, ainsi que DOV, POV et HOV (méthode actuelle de Facebook) si un conflit apparaît. Il implémente aussi la résolution de conflits en utilisant la volonté à changer. Chaque graphe contient 400 utilisateurs, 400 photos et posts et jusqu'à 160 commentaires et tags par photo ou post. Les générateurs et les algorithmes de résolution sont codés en Python, les graphes sont stockés en Datalog (en utilisant la bibliothèque pyDatalog) et les politiques sont stockées dans des fichiers texte.

3.5.3.2. Analyse comparative par rapport aux stratégies basiques

Pour comparer notre modèle avec les autres stratégies de résolution de conflits, nous avons effectué 100 demandes d'accès sur 10 paires différentes de graphes de provenance et de relations en utilisant notre méthode de résolution des conflits, ainsi que les méthodes DOV,POV et HOV (méthode actuelle de Facebook).

Nous avons observé que notre solution est plus équilibrée en termes de refus et d'autorisations que les algorithmes POV et DOV. En effet, l'algorithme POV a abouti à 97% de décisions d'autorisation, tandis que l'algorithme DOV a abouti à 25 % de décisions d'autorisation. Avec notre méthode de résolution, le nombre moyen de permissions était de 32 %. Nous obtenons une proportion légèrement plus élevée de décisions positives qu'avec la méthode HOV (40 % de permissions).

Cependant, en moyenne, 21 % des demandes reçoivent une réponse différente lorsque notre stratégie est utilisée par rapport à HOV. Cela s'explique par le fait que dans HOV, les décisions d'autorisation sont basées sur les préférences de l'hôte. Notre modèle est plus précis, dans le sens où la sensibilité de l'objet ou l'intérêt à partager peut conduire à un refus d'accès, indépendamment des préférences de l'hôte. Par exemple, lorsque la préférence de l'hôte mène à une autorisation, mais que la politique du propriétaire refuse l'accès au demandeur en raison d'un manque de confiance envers celui-ci, la sensibilité ainsi que le coefficient α seront élevés. Cela peut entraîner

un refus dans notre stratégie. Le scénario inverse peut également se produire.

3.5.3.3. Étude comparative détaillée des approches fines

Le modèle présenté dans SUCH et al. 2015 utilise des notions différentes de celles des autres modèles présentés dans la littérature, ainsi que des nôtres. Afin de comprendre les éventuelles divergences entre notre modèle et le modèle en question, nous avons procédé à la simulation de plusieurs centaines de requêtes d'accès sur des graphes variés, en utilisant notre prototype. Un tiers des accès reçoivent des décisions différentes. Cette divergence se produit lorsque la différence de valeur des coefficients entre α et β est proche de 1, avec une valeur de α très élevée. En effet, dans cette configuration, la confiance que les contrôleurs accordent à la personne effectuant la requête l'emporte sur la sensibilité et l'intérêt de partager, réduisant ainsi l'importance respective de ces deux valeurs.

Lors qu'on regarde le modèle présenté dans SUCH et al. 2015, qui se concentre sur l'importance du conflit, cette importance est déterminée par la différence entre la confiance que le contrôleur accorde à la personne qui effectue la requête et la sensibilité de l'objet. Ainsi lorsque la valeur absolue de cette différence est importante, le conflit est important aux yeux du contrôleur. Dans cette approche, contrairement à la nôtre, une personne qui accorde une confiance élevée à un individu tout en refusant de partager avec lui peut engendrer une importance significative du conflit, ce qui peut conduire à la refus de l'accès. Lorsque les coefficients alpha et bêta sont fortement polarisés, notre gestion du conflit engendre une réponse opposée à celle de la stratégie de SUCH et al. 2015.

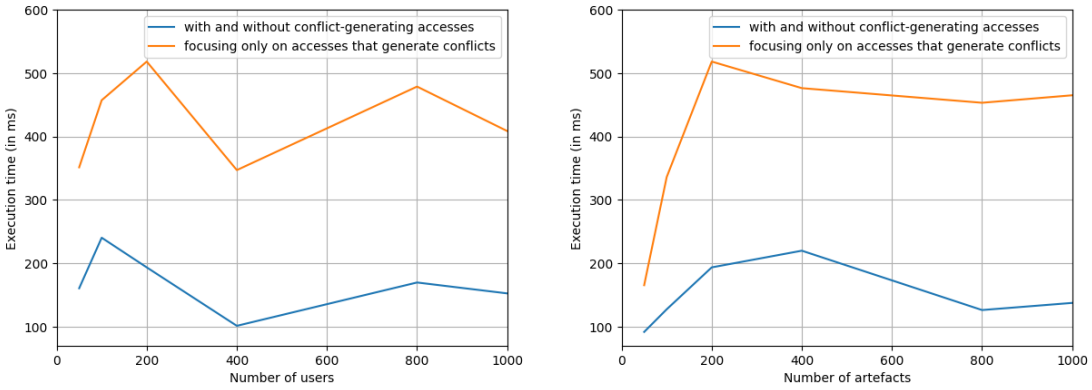
Nous pouvons conclure qu'en dehors de ces cas limites, notre modèle obtient des résultats semblables à ceux de modèles détaillés se basant sur différentes notions.

3.5.3.4. Performances

Pour évaluer la faisabilité de notre méthode nous avons regardé les variations de temps d'exécution de notre méthode ainsi que le temps nécessaire pour résoudre un conflit en utilisant l'algorithme de combinaison DOV. Ces expériences ont été réalisées sur un MacBook Pro équipé d'une puce Apple M2 et de 16 Go de RAM.

Les Figures 3.3 montrent les résultats des performances de notre méthode lorsqu'on agrandit la taille du graphe en augmentant le nombre d'utilisateurs (Figure 3.3a) ou le nombre d'artefacts (Figure 3.3b). Le temps nécessaire pour notre prototype est entre 100 et 250 millisecondes en moyenne lorsqu'on prend en compte les accès générant ou pas un conflit. Lorsque on considère uniquement le temps mis par les accès ayant nécessité une résolution de conflits, les valeurs peuvent monter jusqu'à 530 millisecondes, restant bien en dessous de une seconde. Par ailleurs, on observe qu'au-delà d'une certaine taille, le temps nécessaire pour résoudre le conflit n'augmente pas en fonction du nombre d'utilisateurs ou d'objets dans le graphe. Ceci est une bonne propriété, car les réseaux sociaux contiennent une grande quantité d'utilisateurs et d'objets. Lors de l'utilisation d'un algorithme de combinaison le résultat est quasi

instantané (entre 2 et 10 millisecondes). Ceci s'explique par l'absence de recherche dans le graphe de provenance, et donc l'ouverture et fermeture du fichier du graphe et la recherche à l'intérieur.



(a) Évolution du temps de résolution en fonction du nombre d'utilisateurs de notre solution

(b) Évolution du temps de résolution en fonction du nombre d'artefacts de notre solution

FIGURE 3.3. – Performances de notre solution de résolution de conflits

3.6. Conclusion et Discussion

Dans ce chapitre, nous avons proposé un modèle de prise de décision collaborative à granularité fine pour les réseaux sociaux, utilisant des facteurs de prise de décision calculés automatiquement qui quantifient les aspects pertinents de l'objet demandé et des relations entre les contrôleurs et le demandeur, produisant une formule de seuil. Notre modèle nous permet de répondre au critère de gestion de conflit mis en avant dans PACI et al. 2018 et peut compléter la fonction d'évaluation de notre algorithme de contrôle d'accès présenté dans le chapitre précédant.

Le calcul des facteurs de prise de décision repose sur l'utilisation des données de provenance et les relations sociales pour comprendre les caractéristiques des liens entre les contrôleurs, les demandeurs et les objets. Ce modèle est objectif grâce à des calculs indépendants des entrées des utilisateurs, permettant pour chaque requête de calculer la sensibilité de l'objet demandé ainsi que l'intérêt à le partager avec l'utilisateur effectuant la requête.

La sensibilité des objets est calculée à partir de la confiance en la communauté des contrôleurs et de la sensibilité héritée de l'objet. Cela permet de prendre en compte la protection de la vie privée des utilisateurs. L'intérêt à partager un objet est déterminé par l'exactitude des objets et leur propagation, ce qui permet d'estimer les avantages potentiels du partage.

Les coefficients de crédibilité ajoutent une flexibilité supplémentaire en pouvant inclure des notions additionnelles.

Une validation de la pertinence de nos choix par comparaison à d'autres méthodes de résolution de conflits à granularité fine est fournie. Un prototype de preuve de concept a été développé pour montrer la faisabilité en termes de performance et pour tester sur un grand nombre de scénario notre modèle par rapport à des algorithmes de combinaison ou des modèles basés sur le seuil. Plusieurs scénarios sont simulés en utilisant différents graphes sociaux et de provenance afin de valider notre approche. Une validation par une étude utilisateur est laissée pour des travaux futurs. Pour ce faire, une interface devrait être développée comme dans H. HU, AHN et JORGENSEN 2011 et SUCH et al. 2015 pour présenter le modèle à un panel d'individus et évaluer à travers leurs commentaires la pertinence et la perception de cette stratégie de résolution.

Dans le chapitre suivant, nous présentons un autre aspect de l'utilisation de la provenance pour vérifier la conformité du traitement des données personnelles.

4. Vérification automatisée de lois

Sommaire

4.1. Introduction . . . 128
4.2. Lois relatives aux systèmes informatiques . . . 129
4.2.1. Règlement Général sur la Protection des Données . . . 130
4.2.1.1. Principes du RGPD . . . 131
4.2.1.2. Droits de la personne concernée . . . 132
4.2.1.3. Le RGPD au Royaume-Uni . . . 134
4.2.2. Protection de données à travers le monde . . . 134
4.3. Conformité et sa vérification dans la littérature . . . 135
4.3.1. Du langage juridique au langage logique . . . 136
4.3.2. Conformité dès la conception . . . 136
4.3.3. Vérification de la conformité : audit et surveillance . . . 137
4.4. Défis de la conformité du traitement des données . . . 138
4.4.1. Traçabilité des finalités . . . 139
4.4.2. Automatisation de la vérification . . . 140
4.4.3. Réplication des données . . . 141
4.5. Utilisation de la provenance pour vérifier la conformité . . . 142
4.5.1. Graphe de provenance : nœuds et attributs pour la conformité . . . 143
4.5.2. Patterns pour vérifier la conformité . . . 145
4.5.2.1. Principes du RGPD . . . 145
4.5.2.2. Droit des utilisateurs . . . 151
4.6. Prototype et validation . . . 154
4.6.1. Architecture . . . 155
4.6.2. Validation . . . 160
4.6.3. Améliorations et travaux futurs . . . 163
4.7. Conclusion et discussion . . . 165

4.1. Introduction

Avec l'augmentation du nombre des systèmes informatiques et la multiplication des services en ligne, tels que l'informatisation des données médicales, les achats en ligne, les démarches administratives, voire la digitalisation des cartes d'identité [1], la collecte de données personnelles connaît une croissance exponentielle. Cette évolution

1. https://france-identite.gouv.fr/, consulté en septembre 2024

rend indispensable l'élaboration de lois régissant et protégeant ces données sensibles. Cependant, il ne suffit pas d'avoir des cadres législatifs ; il est également crucial de développer les technologies appropriées pour mettre en œuvre ces mesures et garantir la conformité aux régulations en vigueur. La conformité réglementaire désigne le respect des règles et normes applicables à une organisation dans l'ensemble de ses pratiques. Cette conformité se manifeste à différents niveaux : lors de la définition des procédures et processus généraux, mais aussi dans leur mise en œuvre, dans le choix de l'architecture des systèmes informatiques et de stockage des données, ainsi que dans les technologies et protocoles employés. La vérification de la conformité consiste donc à s'assurer que l'organisation implémente effectivement à ces exigences. Manuellement, cette tâche représente un travail considérable. Toutefois, avec l'informatisation de la plupart des traitements de données, il est envisageable d'utiliser des processus automatisés, ce qui permettrait d'automatiser partiellement la vérification de la conformité.

C'est dans ce contexte que s'inscrit le travail présenté dans ce chapitre. Nous présentons comment à partir d'une formalisation des données et des processus de traitement des données d'un système en graphe de provenance, une vérification automatisée ou semi-automatisée est possible pour certaines des notions clés de la protection du traitement des données personnelles. Un outil de vérification, utilisant Prolog, un langage de programmation logique, nous permet d'implémenter et vérifier l'application des patterns de provenance.

Dans ce chapitre, nous commencerons par examiner en détail les principales lois relatives à la protection des données sensibles dans les systèmes informatiques, avec un focus particulier sur l'Europe et le RGPD (Section 4.2). Ensuite, dans les Sections 4.3 et **??**, nous passerons en revue les différentes solutions proposées dans la littérature ainsi que les défis à relever. Enfin, nous présenterons notre approche pour automatiser la vérification de la conformité d'un système aux réglementations, en tirant parti de la provenance des données introduite en Section 1.3 (Section 4.5). La Section 4.6 présente le prototype d'un outil implémentant notre approche ainsi qu'une validation de notre modèle.

4.2. Lois relatives aux systèmes informatiques

Depuis quelques années, de plus en plus de données personnelles sont stockées par différentes organisations, telles que les gouvernements, les sites web, les boutiques en ligne, les entreprises, les hôpitaux, dans des systèmes informatiques. C'est dans ce contexte que de nombreux pays ont mis en place des règlementations encadrant le stockage et le traitement de ce type de données.

Avec l'essor des technologies et de l'information ainsi que de la mondialisation de l'économie à la fin des années 1970, l'Organisation de Coopération et de Développement Économiques (OCDE) a adopté les lignes directrices sur la protection de la vie privée et les flux transfrontières de données à caractère personnel en 1980

(OECD 2002). L'objectif consistait à harmoniser les lois sur la protection des données à l'échelle internationale, tout en facilitant la libre circulation des informations entre les pays. Huit principes fondamentaux sont mis en avant pour la protection des données, incluant la collecte équitable et légale, la limitation des finalités, la qualité des données, la sécurité, la transparence, et les droits d'accès des individus.

Dans ce même but, l'Union Européenne avait aussi adopté une première directive en 1995 pour la protection des données, la Directive 95/46/CE également connue sous le nom de Directive sur la Protection des Données (THE EUROPEAN PARLIAMENT AND THE COUNCIL 1995). Elle introduit les notions de consentement éclairé, droit d'accès et de rectification des données ainsi que les devoirs du responsable de traitement. Ainsi, cette directive impose des règles strictes sur la collecte, le traitement, et la conservation des données personnelles dans le but de garantir le respect de la vie privée des individus. C'est cette directive qui a servi de fondement au RGPD, qui l'a remplacée en 2018.

Dans cette section, nous présentons un aperçu des réglementations et des principes existants en matière de protection des données, en mettant l'accent principalement sur le RGPD, qui s'applique aux États membres de l'UE. Nous mettons ensuite en lumière la similarité des principes de ce règlement avec ceux d'autres réglementations à travers le monde, que nous décrivons brièvement.

4.2.1. Règlement Général sur la Protection des Données

Le RGPD (*General Data Protection Regulation* s. d.) [2] est une réglementation de l'UE qui précise quelles données personnelles doivent être collectées, traitées et partagées, ainsi que les modalités de ces actions. Proposé en 2016 comme évolution de la Directive 95/46/CE, le RGPD est appliqué uniformément par tous les États membres de l'UE depuis le 25 mai 2018. Une version britannique, le GDPR-UK, a été mise en place au Royaume-Uni après le Brexit, utilisant les mêmes principes que le RGPD. Ce règlement vise à renforcer la protection des données personnelles et les droits des individus. Il concerne tout organisme, public ou privé, basé sur le territoire de l'UE ou ciblant directement les résidents européens. Par exemple, un site de commerce en ligne états-unien dont le contenu est rédigé en français et qui expédie ses produits en France devra respecter le RGPD.

Pour établir la portée et le contexte d'application du règlement, l'article 4 définit les principaux concepts.
Ainsi, les *données personnelles* sont définies comme toute information se rapportant à une personne physique identifiée ou identifiable, appelée *personne concernée*.
Le *traitement des données personnelles* est défini comme toute opération ou ensemble d'opérations effectuées sur des données personnelles. Ces opérations peuvent inclure

2. La version française de ce texte peut être consultée à l'adresse https://www.cnil.fr/fr/reglement-europeen-protection-donnees.

la collecte, l'enregistrement, l'organisation, la conservation, l'adaptation, la modification, l'extraction, la consultation, l'utilisation, la communication par transmission, la diffusion ou toute autre forme de mise à disposition, ainsi que le rapprochement. Il est important de noter que ces opérations peuvent ne pas être informatisées, mais elles sont tout de même concernées par cette réglementation.
Le *responsable du traitement* des données est la personne ou l'organisation en charge de la collecte, du stockage et du traitement des données et responsable du maintien de la conformité au RGPD.
Le RGPD définit également le *consentement* comme l'accord explicite et librement donné par la personne concernée au traitement d'une donnée personnelle, pour une finalité précise. Le consentement de la personne concernée doit être donné par une personne informée.

Le RGPD stipule que tout traitement de données doit avoir un objectif précis, légitime et légal, en lien avec l'activité professionnelle de l'organisation réalisant ce traitement. Ce règlement redéfinit et introduit ainsi de nouveaux concepts tels que le consentement, la transparence du traitement des données vis-à-vis des individus, ainsi que l'application de sanctions. De nouvelles obligations apparaissent pour les organisations, telles que la conformité par conception de leurs systèmes et processus, ainsi que les notifications obligatoires en cas de violation de données. Les principaux points du RGPD sont appelés les principes et son présentés dans le Chapitre II du règlement. Dans ce règlement, de nombreux droits sont aussi introduits pour les *personnes concernées* (Chapitre III du RGPD). Ils sont divisés en trois catégories : l'accès aux données, la rectification et l'effacement des données et, enfin, l'opposition.

En Sections 4.2.1.1 et 4.2.1.2 , nous donnons un résumé de ces principes et droits et récapitulons en Section 4.2.1.3 les principales différences entre la règlementation au sein de l'UE et le Royaume-Uni.

4.2.1.1. Principes du RGPD

Plusieurs principes sont définis dans le RGPD pour encadrer le traitement les informations personnelles, sur la base des définitions précédentes. Ces principes sont la licéité, loyauté, transparence, la limitation des finalités, la minimisation des données, l'exactitude, la limitation du stockage, l'intégrité et la confidentialité, la responsabilité (Art. 5).

Licéité, loyauté, transparence La licéité du traitement dans le RGPD signifie que toute utilisation de données personnelles doit s'appuyer sur une base juridique légitime, comme le consentement, l'exécution d'un contrat, une obligation légale, la protection d'intérêts vitaux, une mission d'intérêt public, ou les intérêts légitimes du responsable du traitement. Sans l'une de ces justifications, le traitement est jugé illégal. Les données doivent alors être traitées dans le respect du consentement de la personne concernée et de la loi. Toute utilisation des données personnelles doit être transparente pour la personne concernée.

Limitation des finalités Les données ne peuvent être traitées qu'aux fins pour lesquelles la personne concernée a donné son consentement.

Minimisation des données Les données personnelles utilisées par un processus doivent être réduites au maximum, c'est-à-dire que seules les données indispensables au bon déroulement du processus peuvent et doivent être utilisées.

Exactitude Les données collectées doivent être exactes et la personne concernée a le droit de les corriger si elles ne le sont pas.

Limitation de stockage Les données collectées seront stockées si et seulement si elles sont utilisées et dès que les données ne seront plus nécessaires, elles devront être effacées.

Intégrité et confidentialité Les données collectées et stockées ne doivent pas être compromises et doivent être protégées pour garantir l'intégrité des données et leur confidentialité.

Responsabilité Le responsable du traitement des données doit être en mesure de prouver que toutes les données personnelles du système ont été traitées de manière conforme aux six principes évoqués ci-dessus.

Les différents chapitres et articles du RGPD viennent préciser ces principes en abordant les droits de la *personne concernée*, les différentes responsabilités des responsables, les autorités de contrôles, les sanctions, etc... Le travail de RHAHLA et al. 2021 établit un lien entre les principes et les articles correspondants dans lesquels ils sont traités. Cette correspondance est illustrée dans le Tableau 4.1.

La conformité (ou l'absence de conformité) de certains articles se reflète dans les évènements du système. Les journaux de ces événements peuvent permettre d'automatiser la vérification de la conformité du système pour certains aspects de ces principes.

4.2.1.2. Droits de la personne concernée

La conformité d'un système passe aussi par le respect des droits de la personne concernée. Certains de ces droits sont directement intégrés aux principes, comme la transparence, tandis que d'autres sont plus distincts des principes fondamentaux. Les droits d'une personne concernées sont le droit d'accès (lié donc au concept de transparence du traitement), le droit à la rectification, le droit à l'oubli ou à l'effacement, le droit à la limitation du traitement, le droit à la portabilité des données et le droit d'opposition. Ils sont séparés dans le RGPD en trois catégories : les droits liés à l'accès aux données, ceux liés à la rectification et à l'effacement et enfin ceux d'opposition.

Principes du RGPD	Articles du RGPD	Considérants du RGPD [3]
Légalité, équité et transparence	5(1)(a), 6, 7, 12, 13, 14, 15, 19, 24(1), 25(1,2), 30, 32(1), 33, 34,	39(1–4), 42, 43, 58, 60, 61,62, 63, 74(1–3), 78, 86, 87
Limitation des finalités	5(1)(b), 6(4), 22, 24(2)	29, 39(6), 39(9), 50(1), 71(1–2)
Minimisation des données	5(1)(c), 25(2)	39(7–8), 39(10), 78(2–3), 156
Exactitude	5(1)(d), 16, 17, 18, 21, 13(2)(c),14(2)(d), 15(1)(e)	39(11), 63(4), 65(1-4), 66, 67(1–3), 69(1–2), 71
Limitation de la conservation	32(1)(b,c,d), 5(1)(e), 13(2)(a),20(1,2), 15(1)	49, 39(10), 63, 68(3–6), 83(3)
Intégrité et confidentialité	5(1)(f), 6(4)(f),25(1), 25(2), 28(3)(b), 29, 32(1)(a), 32(1)(b), 32(2),40(2)(d)	28, 29, 39(12), 49(1–2),75, 78, 81, 83(2–3), 84, 88, 89, 90, 91, 92, 95
Responsabilité	5(2), 24, 30, 32(1)(a,b,c), 34, 35(11),37(7)	82

TABLEAU 4.1. – Correspondance entre les principes du RGPD, les articles et les considérants associés

Accès aux données En lien avec l'accès aux données, la personne concernée à le **droit d'accès** (Art. 15). Les individus ont le droit d'obtenir certaines informations concernant leurs données personnelles traitées par une organisation. En particulier, ils ont le droit de savoir si leurs données sont traitées (confirmation du traitement), et de connaître les données personnelles que l'organisation possède (accès aux données), ainsi que de recevoir une copie de ces données. La réponse de l'organisation doit contenir un certain nombre d'informations (les finalités de traitement, la durée de conservation, l'existence des droits de l'individu, etc.). Ils ont aussi le droit de savoir si les données ont été transférées à des tiers.

Rectification et Effacement La personne concernée a le **droit à la rectification** de ces données (Art.16) dans les meilleurs délais de la part du responsable du traitement si les données sont inexactes ou incomplètes.

Le **droit à l'oubli** (Art. 17) permet à la personne concernée de demander l'effacement des données sauf dans certaines situations (obligation légale, intérêt publique, justice, etc.). L'individu a le droit de demander l'effacement de ses données personnelles lorsque les données ne sont plus nécessaires, le consentement a été retiré, il s'oppose au traitement des données ou le traitement est illégal.

Dans cette catégorie de droits, on trouve le **droit à la limitation du traitement** (Art. 18), applicable en cas d'inexactitude des données ou de traitement illicite, ainsi que le **droit à la portabilité des données** (Art. 20), qui facilite le transfert des données entre différents services, par exemple lors d'un changement de prestataire de services numériques.

Opposition Le **droit d'opposition** (Art. 21) donne à la personne concernée la possibilité de s'opposer au traitement de ses données, en particulier pour des raisons tenant à leur situation particulière ou lorsqu'il s'agit de marketing direct. Ceci mène à l'arrêt du traitement de ses données par le responsable du traitement en dehors des motifs légitimes et impérieux ou à la défense de droits en justice.

Comme pour les principes, on peut voir le respect ou non de ces droits à travers des traces d'exécution du système et de ses évènements.

4.2.1.3. Le RGPD au Royaume-Uni

Après le Brexit, afin de garantir la continuité des règles de protection des données personnelles au Royaume-Uni, le RGPD a été adapté pour correspondre au contexte juridique britannique (*UK General Data Protection Regulation (UK GDPR)* 2024). Ce texte est aussi appelé le Data Protection Act. Les principes fondamentaux sont les mêmes, cependant certaines différences existent quant au transfert de données dans les pays tiers.

4.2.2. Protection de données à travers le monde

À l'instar de l'UE avec le RGPD, de nombreux pays ont choisi de faire évoluer ou de créer leur législation en matière de protection des données personnelles. Par exemple au niveau international, l'OCDE a fait évoluer ses lignes directives en 2013 pour refléter les changements technologiques et les nouvelles pratiques en matière de protection des données (OECD 2013). Les principales modifications incluent le renforcement de la responsabilisation des organisations ainsi que l'approche de la sécurité. À l'échelle nationale, la législation a également évolué. De nombreux pays ont renforcé leurs règles en matière de protection des données, en durcissant les principes. D'autres ont introduit des lois et règlements permettant un contrôle plus strict du traitement des données, comme le montrent les exemples ci-dessous.

Par exemple, le Canada avait proposé en 1996 un cadre de protection des données, le Canadian Standards Association Model Code for the Protection of Personal Information (CSA 1996) en réponse aux préoccupations en rapport à la protection de la vie privée dans un environnement de plus en plus numérique. Il a été incorporé dans la législation canadienne à travers la Loi sur la protection des renseignements personnels et les documents électronique (GOUVERNEMENT DU CANADA 2000) en vigueur depuis 2000 (avec des modifications réalisées en 2015 et 2019). Les principes sont la limitation de la collecte des données et de leur utilisation, le consentement et la responsabilité des organisations.

Les États-Unis possèdent aussi depuis des années des normes concernant les données personnelles dans le cadre du médical. En effet, depuis 1996, le Health Insurance Portability and Accountability Act (HIPAA) (U.S. CONGRESS 1996) vise à protéger les informations médicales personnelles en établissant des règles pour la gestion des

données de santé, notamment en ce qui concerne leur stockage, partage et transmission. Certains états des États-Unis ont développé des législations plus récentes pour la protection des données personnelles en général, comme l'état de Californie avec le « California Consumer Privacy Act » (CALIFORNIA STATE LEGISLATURE 2018) entré en vigueur en 2020, modifiée par le « California Privacy Rights Act » en 2023. Reposant sur les mêmes principes que les règlementations précédentes, il définit les droits de consommateurs ainsi que les obligations des organisations.

Nous trouvons des législations semblables en Amérique Latine avec par exemple la « Lei Geral de Proteção de Dados » (BRAZILIAN NATIONAL CONGRESS 2018) entrée en vigueur en 2020 au Brésil. On voit aussi apparaître des lois semblables dans d'autres continents comme en Chine avec la « Personal Information Protection » (NATIONAL PEOPLE'S CONGRESS (NPC) s. d.) entrée en vigueur fin 2021.

Toutes ces lois et réglementations partagent des principes similaires à ceux du RGPD, bien que des différences puissent exister dans la terminologie ou la structuration des principes. Elles établissent des règles sur la manière dont les organisations doivent collecter, traiter et transmettre les données personnelles. De plus, elles définissent les droits des individus concernant leurs données ainsi que les responsabilités des organisations en matière de protection des données.

4.3. Conformité et sa vérification dans la littérature

De nombreuses lois régulent les activités numériques à travers le monde, rendant la vérification de la conformité complexe et chronophage. Compte tenu de cette complexité, l'automatisation de cette vérification apparaît comme une solution prometteuse. En effet, l'automatisation de la conformité et de sa vérification aux textes légaux dans les systèmes informatiques constitue un domaine de recherche en pleine expansion, situé à l'intersection de l'informatique, du droit et de l'intelligence artificielle. Ces travaux visent à développer des outils capables d'assurer de manière efficace et continue le respect des régulations, en réduisant les erreurs humaines et en s'adaptant à l'évolution rapide des cadres législatifs. L'automatisation de la vérification légale poursuit principalement trois objectifs : réduire les erreurs humaines, accélérer les processus de vérification et garantir une conformité continue. Toutefois, traduire le langage juridique en langage logique présente des défis considérables. Certains aspects des lois ne se prêtent pas à une vérification automatique. Par exemple, dans le cadre du RGPD, il est impossible d'automatiser la vérification de la clarté des demandes de consentement vis-à-vis des personnes concernées, car cette évaluation repose sur des nuances contextuelles et subjectives que les algorithmes ne peuvent pas entièrement saisir.

Ces dernières années, certains travaux ont étudié l'application du RGPD dans les systèmes existants, en particulier la conformité : les systèmes sont-ils conformes ? Comment les rendre conformes ? Existe-t-il un moyen automatisé de vérifier s'ils sont

conformes ? Trois axes de travail sont étudiés : des études analytiques de la conformité dans des systèmes réels, des méthodologies et des outils pour surveiller et vérifier la conformité de manière semi-automatique et des cadres et lignes directrices pour mettre en œuvre la conformité dès la conception. La plupart des travaux se centre sur la dernière piste, en particulier sur la conformité « by design » (SALTARELLA et al. 2021) ou en modifiant les systèmes pour l'inclure. Par exemple les auteurs présentent dans RHAHLA et al. 2021 une étude approfondie sur les outils informatiques existants (académiques et industriels) facilitant l'intégration de la conformité au RGPD dans un système. Cependant peu de travaux se centrent sur la vérification de la conformité des actions réalisées par le système.

Dans cette section, nous présentons les différentes pistes explorées dans la littérature pour automatiser ou semi-automatiser la vérification de lois et règlementations, en particulier en lien avec le RGPD. Nous centrons notre attention en particulier sur les travaux proposant des solutions pour la surveillance de la conformité.

4.3.1. Du langage juridique au langage logique

Traduire les textes juridiques en logique est une tâche complexe et nuancée, qui nécessite de capturer avec précision les significations et les implications du langage juridique dans un format formalisé. Plusieurs approches ont été explorées dans ce domaine comme le montre le travail ESTEVES et al. 2024, notamment l'utilisation des ontologies pour modéliser les relations entre les concepts et entités juridiques. Les ontologies aident à structurer les connaissances juridiques, facilitant ainsi la traduction des dispositions légales en règles logiques. Un autre effort important dans ce domaine est le développement de LegalRule ML (ATHAN et al. 2015), une extension XML conçue pour représenter les normes et le raisonnement juridiques dans un format lisible par machine. Ces méthodologies contribuent à combler le fossé entre le langage juridique et le formalisme logique, permettant une automatisation et une analyse plus efficaces du raisonnement juridique. D'autres travaux se centrent sur rendre la logique compréhensible par les experts juridiques. Dans BARTOLINI et al. 2018, les auteurs examinent comment la logique « Reifed I/O logic » (RIO) permet de créer une représentation lisible par l'humain des articles du RGPD, pouvant être validée par un expert juridique. Cette approche facilite la vérification de la conformité juridique tout en maintenant la précision du formalisme logique.

4.3.2. Conformité dès la conception

Beaucoup de travail a été fait depuis 2016 pour créer des systèmes conformes au RGPD par conception. En particulier des systèmes de confidentialité par conception, car c'est l'une des exigences du RGPD.

Dans TAUQEER et al. 2022, les auteurs extraient du RGPD des TOms (Mesures Techniques Organisationnelles) pour créer un outil, qui peut être intégré dans un système pour le rendre conforme au RGPD. Ils ont basé leur travail sur les contrats existant

entre un utilisateur et l'organisation. Ils les représentent à l'aide de l'ontologie SmashHitCore, issue d'un projet européen ainsi que de graphes de connaissances.

Le travail dans CHHETRI et al. 2022 est la continuation de TAUQEER et al. 2022 où les auteurs donnent un petit aperçu de la conformité au RGPD et améliorent l'outil présenté dans leur travail précédent en ajoutant la notion de consentement. Leur travail est orienté sur les villes intelligentes et les appareils IoT, donc le but du travail est de minimiser l'opération effectuée par leur outil dans le système.

L'article AGOSTINELLI et al. 2019 montre comment créer des modèles pour rendre les processus métier (« business process » en anglais) conformes au RGPD. Le travail se concentre principalement sur les obligations énoncées dans le RGPD pour le responsable du traitement des données. L'objectif des auteurs est de proposer une méthode pour ajouter la confidentialité dès la conception des processus métier afin de les rendre conformes au RGPD. Il n'existe aucun outil ou algorithme pour automatiser la création du modèle, seulement une étude de cas d'un opérateur téléphonique.

Comme dans AGOSTINELLI et al. 2019, BARATI et al. 2020 utilise également les modèles métier pour ajouter la conformité dès la conception. Les auteurs centrent leur travail sur les appareils IoT et introduisent pour la vérification de la conformité les contrats intelligents et la blockchain. Ils réutilisent la méthodologie proposée dans BASIN et al. 2018. Le travail se concentre sur certains principes du RGPD comme la confidentialité, le consentement de la personne concernée, la minimisation des données, la protection et le transfert des données.

4.3.3. Vérification de la conformité : audit et surveillance

Malgré plusieurs travaux sur la conformité au RGPD, à notre connaissance, il existe peu de travaux sur la surveillance et l'automatisation de l'audit des systèmes après leur conception. Les auteurs de ARFELT et al. 2019 proposent une traduction d'un sous-ensemble des articles du RGPD en formules de logique temporelle pour automatiser la vérification de la conformité au RGPD. Ils utilisent MonPoly qui prend un ensemble de journaux générés à partir des événements système et traduits en logique MFOTL, puis compare aux formules de logique du RGPD et renvoie s'il y a eu une violation de la conformité au RGPD. Cependant, les articles du RGPD considèrent qu'il s'agit d'un sous-ensemble très restreint et l'un des principes importants du RGPD, la finalité, n'est pas abordé. L'article PALMIRANI et al. 2018 présente un cadre utilisant différentes techniques pour la vérification de la conformité au RGPD. Ce travail a été réalisé dans le cadre d'un projet européen Cloud4Eu (*Cloud for Europe* s. d.), pour la mise en place d'un environnement de cloud computing sécurisé pour le secteur public. Les auteurs ont proposé une traduction semi-automatisée d'un texte de loi ou de règlement en règles logiques. Leur modèle est dynamique, car il peut suivre les modifications et la création de nouvelles réglementations et mettre à jour les règles. Les règles sont ensuite utilisées pour vérifier la conformité sur un ensemble de systèmes de logs. Le cadre peut également être utilisé en conformité par conception, car il peut générer à partir du modèle métier, toutes les traces possibles et vérifier la conformité des traces.

D'autres travaux utilisent le machine learning pour automatiser leur approche, en

particulier l'étape de traduction d'un texte (HAMDANI et al. 2021), cependant certains concepts du RGPD sont difficilement capturés par ces approches.

Notre approche est similaire à celle de ARFELT et al. 2019, mais offre une couverture plus étendue des articles grâce aux propriétés de la provenance utilisées dans notre modélisation.

4.4. Défis de la conformité du traitement des données

Les défis liés au traitement des données avec l'apparition de cadres juridiques précis sont nombreux. D'une part, les organisations doivent surmonter des obstacles organisationnels et techniques pour se conformer aux lois, en interprétant les textes juridiques et en adaptant leurs processus internes. En particulier, une des principales difficultés est la prise en compte du contexte dans lequel se déroulent les processus et dans lequel sont utilisé les données. C'est le concept de finalité (« purpose » en anglais).

D'autre part, ces réglementations exigent également que les organisations démontrent leur conformité à travers des audits, qu'ils soient internes ou externes, c'est le principe de responsabilité.
Cependant, au vu de l'étendue des articles du RGPD, démontrer la conformité n'est pas une tâche triviale. Pour respecter les contraintes instaurées dans les articles, les organisations doivent documenter leurs processus de traitement des données, les bases légales pour ce traitement, les consentements obtenus, et les mesures de sécurité mises en place mais aussi des audits rigoureux qui peuvent porter sur l'examen des processus de traitement des données, des mesures de sécurité, et des procédures de réponse aux violations de données. Pour simplifier cette tâche, le défis est d'automatiser au maximum les audits, afin d'alléger la tâche aux auditeurs.

Un autre défi majeur est celui d'identification des données qui contiennent des informations personnelles (par exemple, un rapport contenant l'adresse d'un utilisateur), tout en assurant (et en vérifiant) la cohérence des modifications sur les données répliquées.

Dans ce travail, nous nous concentrons sur la traçabilité des finalités et le suivi des données répliquées dans le cadre de l'automatisation de la vérification des principes du RGPD. Certains articles du RGPD font référence à des actions qui peuvent être vues dans les journaux du système. En regardant les traces d'exécution d'un système, nous pouvons voir la conformité ou l'absence de conformité aux articles du RGPD. Les exemples suivants illustrent à travers de courtes traces d'exécution les défis cités ci-dessus. Le scénario repose sur un système de boutique en ligne. Un utilisateur peut créer un compte, acheter des articles, demander un remboursement, laisser des avis sur les produits. Pour réduire les traces, les scénarios se concentrent sur un défi ou principe du RGPD à la fois.

4.4.1. Traçabilité des finalités

Garder en mémoire les finalités consenties par une personne concernée est important pour déterminer quels processus respectent ou non les principes du RGPD. Imaginons la situation suivante.

Alice est une cliente (*personne concernée*) d'une boutique en ligne et Bob travaille au service client. Alice a donné son consentement à certaines fins sur certaines de ses données personnelles. Lorsqu'elle achète un article, la boutique en ligne peut utiliser son nom et son adresse physique pour envoyer l'article, son adresse e-mail pour envoyer le reçu et la tenir au courant de l'expédition et son numéro de compte bancaire pour le paiement. Pour les besoins de remboursement, elle autorise uniquement la boutique en ligne à utiliser son numéro de compte bancaire.

Les deux traces suivantes illustrent la conformité à deux des principes clé du RGPD.

Exemple 23 (Limitation des finalités) *Comme expliqué précédemment, un système est conforme au principe de limitation des finalités si les données personnelles sont utilisées uniquement aux fins consenties par la personne concernée. Ici, Alice a acheté un article, mais il contient des défauts. Elle demande donc un remboursement.*

Le Tableau 4.2 contient une trace conforme au principe (colonne de gauche) et une le violant (colonne de droite).

Étape	Trace conforme	Trace non conforme
1	Alice demande un remboursement pour la commande 4352.	Alice demande un remboursement pour la commande 4352.
2	Bob effectue le remboursement en utilisant son numéro de compte bancaire.	Bob effectue le remboursement en utilisant le numéro de compte bancaire MAIS AUSSI son adresse physique.

TABLEAU 4.2. – Traces d'exécution illustrant la limitation des finalités

On observe dans la trace non conforme que l'adresse physique est utilisé pour le remboursement, cette fin n'est pas consentie par Alice.

Exemple 24 (Licéité du traitement) *Le principe de limitation des fins est directement lié au principe de licéité du traitement dans le cadre du RGPD, car un traitement de données ne peut être considéré comme légal que s'il respecte les finalités définies initialement et n'est pas utilisé de manière incompatible avec ces principes.*

Comme de nombreux sites Internet, la boutique en ligne de notre scénario utilise des cookies pour améliorer l'expérience des clients. Les cookies sont utilisés à différentes fins : à des fins d'analyse pour aider la boutique en ligne à améliorer son site, pour proposer au client des articles adaptés en fonction des éléments qu'il a consultés et ils peuvent également être utilisés pour être envoyés à des tiers. Alice a donné son consentement pour les trois finalités. Nous présentons ensuite dans le Tableau 4.3 deux scénarios montrant une situation conforme et une situation non conforme.

Étape	Trace conforme	Trace non conforme
1	Alice a recherché un grille-pain et une cuisinière.	Alice a recherché un grille-pain et une cuisinière.
2	Un cookie est envoyé à l'équipe de maintenance du site (cookie d'analyse).	Un cookie est envoyé à l'équipe de maintenance du site (cookie d'analyse).
3	Un cookie est envoyé à un site partenaire vendant également des ustensiles de cuisine (cookie tiers).	Alice retire son consentement pour les cookies tiers.
4	Alice retire son consentement pour les cookies tiers.	Un cookie est envoyé à un site partenaire vendant également des ustensiles de cuisine (cookie tiers).

TABLEAU 4.3. – Traces d'exécution illustrant la licéité du traitement

Dans le cas de la trace non-conforme, le cookie destiné à un tiers est envoyé après qu'Alice ait retiré son consentement.

Ces deux traces illustrent que la distinction entre conformité et non-conformité aux principes réside dans la divergence des finalités consenties. Sans accès aux finalités et aux modifications apportées par la personne concernée, il est impossible de conclure de manière fiable sur la conformité.

4.4.2. Automatisation de la vérification

La vérification de certains aspects du RGPD peut s'étendre dans le temps, car certaines règles imposent des durées ou des délais spécifiques. Par exemple, les données personnelles peuvent être conservées sans être utilisées pendant une période donnée. De plus, le responsable du traitement dispose d'un délai pour répondre aux demandes des personnes concernées. Ces contraintes temporelles exigent une vigilance accrue de la part des auditeurs. Assurer une vérification exhaustive de la conformité peut rapidement devenir un défi complexe et difficile à gérer.

L'exemple suivant montre comment ce délai peut se traduire dans la vraie vie.

Exemple 25 (Limitation de stockage) *Le RGPD stipule qu'une durée maximale de stockage doit être fixée pour les données qui ne sont plus utilisées.*

La boutique en ligne a une limite de 5 ans pour la conservation des données personnelles. Si une donnée n'a pas été utilisée pendant 5 ans, elle doit être supprimée.

Alice, une ancienne cliente de la boutique en ligne, a utilisé pour la dernière fois son compte pour laisser un avis sur l'un de ses achats (son e-mail est nécessaire et elle avait donné son consentement à cet effet). Depuis ce moment, il y a 7 ans, elle n'est plus revenue sur son compte de boutique en ligne. Sept ans plus tard, elle essaie de se connecter au compte. Les traces d'exécution conforme et non-conforme des évènements liés à la tentative de connexion sont représentées dans le Tableau 4.4.

Étape	Trace conforme	Trace non conforme
1	Alice laisse un avis sur son achat.	Alice laisse un avis sur son achat.
2	5 ans plus tard... Le responsable du traitement supprime les données personnelles d'Alice.	7 ans plus tard... Alice essaie de se connecter à son compte.
3	2 ans plus tard... Alice essaie de se connecter à son compte.	La connexion est réussie, elle a accès à son compte.
4	Un message d'erreur apparaît, son compte a été supprimé.	

TABLEAU 4.4. – Comparaison des traces conforme et non conforme concernant la gestion des comptes utilisateurs

Dans le cas de non-conformité, le compte persiste après 7 ans, alors que pour se connecter, l'e-mail est requis et aurait dû être supprimé 5 ans après la dernière utilisation.

4.4.3. Réplication des données

Une dernière complication pour maintenir et vérifier la conformité aux RGPD réside dans la gestion des données répliquées. Si une action de restriction de finalités ou autre impacte une donnée personnelle, elle doit s'appliquer à toutes les copies de cette dernière. Garder les liens entre les données est donc important comme le montre l'exemple suivant.

Exemple 26 (Droit à la rectification) *Imaginons que l'adresse e-mail d'Alice a été copiée de la base des données où sont stockées les données du compte d'Alice, à la base de données servant au service client pour contacter les acheteurs pour évaluer leur commande. Alice demande par la suite de rectifier son adresse e-mail. Le Tableau 4.5 illustre les traces d'exécution conforme et non-conforme à ce scénario.*

Étape	Trace conforme	Trace non conforme
1	L'adresse e-mail est copiée dans une deuxième base de données.	L'adresse e-mail est copiée dans une deuxième base de données.
2	Alice demande à rectifier son adresse e-mail.	Alice demande à rectifier son adresse e-mail.
3	L'adresse e-mail du compte est modifiée ainsi que la copie du service client.	L'adresse e-mail du compte est modifiée.

TABLEAU 4.5. – Comparaison des traces conforme et non conforme concernant la gestion des modifications des copies

Dans le cas conforme, les deux apparitions de l'adresse e-mail ont été mises à jour. Dans la trace non conforme, la copie du service client n'a pas reçu de mise à jour, violant le droit de rectification de la personne concernée.

Ces exemples simples illustrent les défis énoncés au-dessus et montrent comment, à travers les événements, il est possible de vérifier la conformité à certains principes et le respect des droits des personnes concernées. Pour automatiser cette vérification, il est nécessaire de formaliser ces traces afin de les convertir en un modèle exploitable, permettant un raisonnement automatique. Comme expliqué dans le Chapitre 1, la provenance des données offre une représentation efficace des événements d'un système. De plus, l'utilisation de la provenance des données permet de relever un autre défi majeur : identifier quelles données contiennent des informations personnelles, tout en assurant (et en vérifiant) la cohérence des modifications sur les données répliquées.

4.5. Utilisation de la provenance pour vérifier la conformité

Vérifier la conformité d'un système par rapport à un texte de loi peut impliquer plusieurs aspects. Par exemple, cela peut inclure des conditions sur l'architecture du système, la manière de stocker les données, les rôles et responsabilités des managers, gestionnaires et autres agents, ainsi que l'exécution de divers processus.

Les articles d'un texte de loi peuvent ainsi aborder différentes thématiques. Certains articles se limitent à des définitions générales, et ne sont donc pas directement liés à la conformité. D'autres imposent des obligations qui ne concernent pas directement le système informatique, comme certaines obligations du responsable de traitement dans le cadre du RGPD. Il existe également des articles qui nécessitent une vérification supplémentaire, par exemple la clarté d'un questionnaire ou le contenu d'une notification.

Dans ce travail, nous nous concentrons sur la vérification de la conformité du système, en particulier celle des événements liés au traitement des données. Nous illustrons l'utilisation de la provenance dans le cadre de la vérification des principaux principes et droits du RGPD. Nous nous concentrons en particulier sur l'automatisation des articles qui portent sur la conformité des événements du système, ainsi que sur la semi-automatisation de ceux nécessitant une vérification supplémentaire.

Pour raisonner sur les événements du système, nous utilisons le modèle de provenance introduit dans le Chapitre 1. Ces informations nous permettent de suivre l'historique du système et de vérifier a posteriori sa conformité à une norme ou à une réglementation, comme le RGPD, et permettre à l'intérieur d'une organisation de faire les corrections nécessaires.

Dans cette section nous introduisons d'abord les nœuds et attributs nécessaires à la vérification de la conformité (Section 4.5.1), puis détaillerons dans la Section 4.5.2 la modélisation de la notion de consentement ainsi que celle des patterns permettant de

voir dans le graphe si le système est conforme aux principes (Section 4.5.2.1) et droits (Sections 4.5.2.2) du RGPD.

4.5.1. Graphe de provenance : nœuds et attributs pour la conformité

Comme montré dans le Chapitre 1, les données d'un système et les processus manipulant ces données peuvent être représentés sous forme d'un graphe de provenance. Les chemins de ce graphe nous permettent d'observer les liens de causalité entre les différentes données, processus et agents. Dans le cadre du RGPD, il est important de représenter les données personnelles, pour connaître les processus qui les utilisent et déterminer les finalités pour lesquelles elles sont utilisés, garder une trace des consentements donnés par les personnes concernées et vérifier que les droits de ces utilisateurs sont bien respectés.
Nous considérons que les agents sont des personnes concernées, des responsables de traitement et plus généralement des individus interagissant avec le système. Les données personnelles sont représentées sous forme d'artefacts, ainsi que tout le contenu d'un site web (Figure 1.3).

La Figure 4.1 donne un exemple de site de forum en ligne représenté sous la forme d'un graphe de provenance. Ici, Bob, se crée un compte sur ce site (processus *create_account*). Il renseigne un certain nombre de données personnelles (adresse e-mail, téléphone, ...) qui sont stockées. Lors de la création du compte, un numéro d'identité a été associé à son compte, et un mur personnel lui a été attribué. Une liste d'amis a également été associée à son profil. Toutes ces données sont des données personnelles. Suite à la création de son compte, il a dû choisir les finalités pour lesquelles il souhaite que ses informations personnelles soient utilisées. Bob accepte que ses informations personnelles soient utilisées à des fins de marketing (tierces parties), ainsi que d'analyse et d'amélioration : ceci est représenté par l'artefact $consent_bob_v0$. Après un certain temps, il change d'avis : il met à jour son consentement et supprime la finalité marketing, reflétait dans le graphe par l'artefact $consent_bob_v1$. Le paragraphe suivant détaille ce type d'artefact. Bob décide de rejoindre un groupe qui partage son intérêt pour la cuisine. Le système crée un cookie, qui est envoyé à un vendeur tiers, vendant des ustensiles de cuisine. L'agent *DC* est le responsable du traitement (« data controller » en anglais) de ce système. Après un certain temps, Bob demande à accéder à ses données, une requête à laquelle le responsable du traitement répond. Bob demande également la suppression de son numéro de téléphone (processus ask_erase), une demande qui n'a pas encore été satisfaite.

Dans ce graphe, les horodatages sont exprimés en minutes et les attributs des nœuds ne sont pas explicitement représentés.

Nœuds spécifiques Comme défini en Section 1.2.2, les sites web possèdent souvent un modèle de choix possibles pour les individus afin de renseigner leur consente-

ment pour l'utilisation des données personnelles (*modèle de politique de confidentialité*). Ce modèle est représenté dans le graphe de provenance, comme un nœud de type artefact.
Une fois que l'utilisateur a donné ses préférences, on obtient son consentement. Le consentement donné par les personnes concernées est également représenté sous forme d'artefact. Lorsque le consentement est modifié, étant donné que c'est un graphe de provenance, un nouvel artefact sera créé et lié au précédent consentement à travers l'arête wasDerivedFrom. Par exemple, nous observons dans la Figure 4.1 que le consentement donné par Bob a été mis à jour. Ceci se traduit par la création d'un nouvel artefact *consent_bob_v1*, qui est une nouvelle version *consent_bob_v0* dont les finalités ont été modifiées. Les finalités sont quand elles modélisées sous la forme d'attributs.
Il est important de rappeler que les artefacts représentent des objets immuables. Lorsqu'une donnée ou objet du système subit des modifications, cela se traduit dans le graphe de provenance par l'apparition d'un nouvel artefact, lié par une dépendance wasDerivedFrom avec un rôle *mise-à-jour* à l'objet ayant subi la modification.

Attributs spécifiques Nous utilisons deux types d'attributs principalement pour préciser notre modèle : le premier étant pour caractériser les artefacts qui sont des données personnelles. En effet le RGPD régule le traitement des données personnelles. Les autres données du système ne sont pas impactées sauf si elles contiennent des données personnelles. Nous notons dp cet attribut et il peut avoir comme valeur $\mathcal{V}_{dp} = \{Vrai, Faux\}$. Par exemple, $dp(e-mail_bob) = Vrai$ signifie que l'attribut dp de l'artefact $e-mail_bob$ a pour valeur *Vrai*. Cet attribut montre que l'e-mail de Bob est une donnée personnelle. Pour simplifier la notation, nous écrirons aussi D_dp au lieu de $dp(D)$

Le deuxième type d'attribut est nécessaire pour représenter les finalités consenties. Une finalité peut être vue comme une action réalisée par un processus (achat, demande d'information, etc.). Sa finalité est représentée par l'attribut *action*, attribut déjà introduit précédemment. La notion de finalité est également liée au consentement. En effet, un consentement spécifie les finalités pour lesquelles les données personnelles d'une personne concernée peuvent être utilisées. Ces finalités sont représentées sous la forme d'un attribut *purpose* dans l'artefact de consentement. Cet attribut peut être conçu comme une liste où chaque élément est un tuple associant une donnée personnelle à la liste des finalités autorisées. Si le site web n'admet pas ce degré de précision en ce qui concerne les données personnelles et le consentement est appliqué de la même façon à toutes les données personnelles, l'attribut *purpose* est uniquement une liste contenant les finalités.
Dans le but de garder une généralité, dans les deux cas on note l'attribut *purposes* de l'artefact *cons* : $purposes(cons) = [(artefact_1, list_1), ..., (artefact_k, list_k)]$ où $k = 1$ et $artefact_1 = _$ lorsque les finalités sont globales, signifiant que l'artefact n'est pas important ici.

Exemple 27 (Attribut finalité) *Dans le scénario représenté en Figure 4.1, on observe que les finalités sont globales. Les finalités consenties ne dépendent pas des données renseignées. L'attribut purpose de l'artefact consent_bob_v0 est formalisé comme suit.*

$$consent_bob_v0_purposes = [(_, [\text{"}thirdParties\text{"}, \text{"}analysis\text{"}, \text{"}improvement\text{"}])]$$

On rappelle que _ est utilisé lorsque les finalités s'appliquent à tous les artefacts d'une même personne concernée. Si l'on regarde le scénario développé en Section 4.4, on observe que Alice consent à des finalités différentes selon les données personnelles. Ceci peut s'exprimer de la façon suivante :

$$\begin{aligned} consent_alice_v0_purposes \quad = \quad & [(name_alice, [\text{"}buy\text{"}]), (address_alice, [\text{"}buy\text{"}]), \\ & (e-mail_alice, [\text{"}buy\text{"}]), \\ & (bank_account_alice, [\text{"}buy\text{"}, \text{"}refund\text{"}])] \end{aligned}$$

Il est important de rappeler que les attributs ne sont pas directement représentés dans le graphe de provenance afin de garder un certain degré de lisibilité.

Notre travail propose d'utiliser les graphes de provenance pour représenter l'évolution du système au fil du temps et de modéliser les principes et les droits définis dans le RGPD à l'aide de patterns, définis comme des expressions de chemin.

4.5.2. Patterns pour vérifier la conformité

Un pattern est une expression de chemin, qui est composée de conjonctions ou disjonctions de dépendances causales, avec éventuellement des contraintes temporelles ou des contraintes sur les attributs. Les contraintes sur les attributs consistent à vérifier que la valeur de l'attribut dans le graphe correspond à celle du pattern. Pour le cas des attributs avec une valeur booléenne ou simple comme par exemple la valeur de l'attribut action, il suffit de vérifier que la valeur de l'attribut dans le graphe correspond à celle dans le pattern. Si la valeur est une liste, comme dans le cas de l'attribut *purposes* on va regarder si la valeur est dans la liste. On note $\mathsf{isPurpose}(PU, D, C)$ si la finalité PU a été consentie pour la donnée personnelle D dans l'artefact consentement C. Si les finalités sont des finalités globales, la notation est $\mathsf{isPurpose}(PU, _, C)$. Lorsque nous utilisons la négation dans nos patterns, nous adoptons le principe de la négation par échec, comme en Prolog. Autrement dit, la négation d'un élément est considérée comme vraie s'il n'existe aucune instanciation possible pour ce pattern.

4.5.2.1. Principes du RGPD

Certains des principes du RGPD introduits en Section 4.2.1.1 sont directement liés au traitement des données, signifiant que la vérification de la conformité peut être automatisé ou semi-automatisé. Nous formalisons à l'aide de notre modèle les patterns permettant cela pour certains des principes.

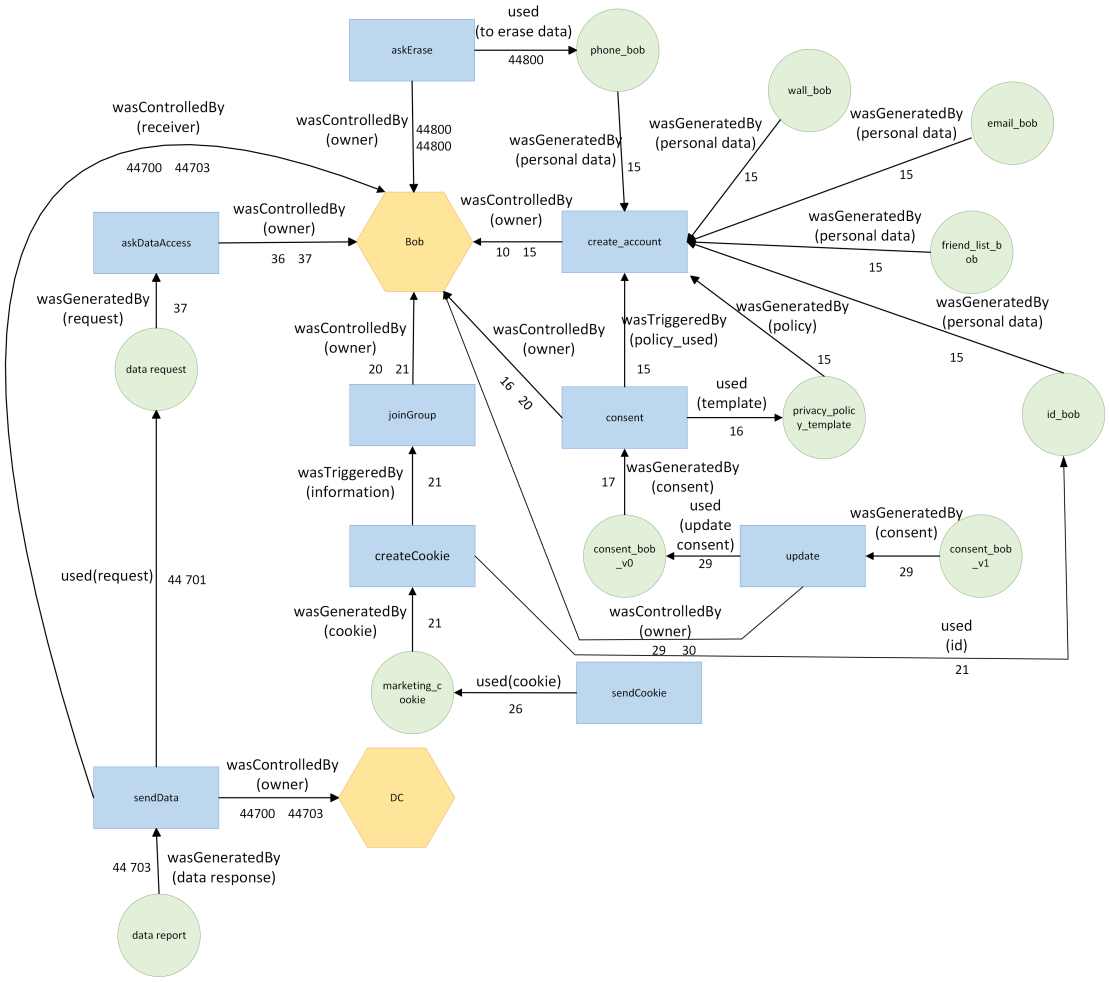


FIGURE 4.1. – Extrait de graphe de provenance d'un forum en ligne.

Consentement et limitation des finalités Comme introduit précédemment, le consentement et la finalité sont très liés. En effet le RGPD stipule dans ces principes de base que tout traitement de données doit être consenti et impose la limitation des finalités comme principe. On met en avant dans cette section comment formellement définir les patterns liés à la finalité et la présence ou non du consentement.

Une donnée personnelle ne peut être utilisée si la personne concernée n'a pas donné son consentement pour la finalité prévue (Art. 6.1.a). Pour savoir si la personne concernée a consenti à utiliser sa donnée personnelle D pour la finalité PU, il faut qu'il existe un artefact consentement dont PU est une finalité pour D. L'expression de chemin suivante formalise ceci :

$$\begin{aligned} \mathsf{consent}(C, D, PU, T) \quad = \quad & \mathsf{wasControlledBy}(P1, S, "owner", TB, TE) \wedge \\ & \mathsf{wasGeneratedBy}(C, P1, "consent", T) \wedge \\ & \mathsf{isPurpose}(PU, D, C) \end{aligned}$$

où C représente l'artefact consentement contenant la finalité PU, PU la finalité pour

laquelle D la donnée personnelle a été utilisée. Le consentement peut être celui donné initialement par la personne concernée (par exemple le processus `consent` dans la Figure 4.1) ou une mise à jour (processus `update` dans la Figure 4.1). T, TB, TE sont les horodatages et $P1$ est le processus générant les préférences de consentement pour une certaine *personne concernée*. Il peut exister plusieurs instanciations pour C et T. Si la valeur de T est intanciée, C ne correspondra qu'à un seul artefact.

Exemple 28 (Pattern pour le consentement) *Dans le scénario présenté dans la Figure 4.1, le pattern* consent(*consent_bob_v0*, *e – mail_bob*, "*analysis*", 17) *est évalué à vrai. En effet, le chemin*

$$\begin{aligned}&\mathsf{wasControlledBy}(\texttt{consent}, \mathit{Bob}, \texttt{"owner"}, 16, 20) \wedge \\ &\mathsf{wasGeneratedBy}(\mathit{consent_bob_v0}, \texttt{consent}, \texttt{"consent"}, 17)\end{aligned}$$

existe dans le graphe de provenance. Le consentement consent_bob_v0 contient la finalité globale "analysis" *et l'horodatage de sa génération est bien* 17.

Dans le cas de finalités globales, c'est-à-dire indépendantes des données personnelles, la variable D, sera remplacée par _ comme expliqué précédemment.

Comme indiqué dans l'article 7.3 du RGPD, la personne concernée doit pouvoir retirer son consentement à tout moment. Si le consentement C est révoqué, cela correspond au chemin suivant dans le graphe :

$$\mathsf{revoke}(C, T) = \mathsf{used}(P, C, \text{"}\mathit{revokeConsent}\text{"}, T).$$

où C est l'artefact de consentement, T l'horodatage indiquant quand il a été révoqué et P le processus de révocation du consentement. Toutes les finalités de tous les artefacts sont révoquées, le prédicat ne dépend donc pas de la donnée ou de la finalité.

Les préférences de consentement peuvent également être mises à jour par la personne concernée, pour ajouter ou supprimer des finalités. Cette situation est représentée par la création d'un nouvel artefact dérivé des données de consentement précédentes, par un processus effectuant une mise à jour. Cette mise à jour est effectuée par la personne concernée.

Avec le prédicat suivant, nous pouvons vérifier si un artefact de consentement C a été mis à jour ou non par la personne concernée S à un certain moment T.

$$\begin{aligned}\mathsf{nextConsent}(C, C1, T) \quad = \quad &\mathsf{wasControlledBy}(P1, S, \text{"}owner\text{"}, TB, TE) \wedge \\ &\mathsf{used}(P1, C, \text{"}updateConsent\text{"}, TU) \wedge \\ &\mathsf{wasGeneratedBy}(C1, P1, \text{"}consent\text{"}, T)\end{aligned}$$

En particulier $\mathsf{nextConsent}(C, C1, T)$ nous donne l'artefact représentant la mise-à-jour suivant directement la version du consentement représentée par C. Il est important de rappeler que les artefacts étant des représentations d'objets immuables, une

modification sur l'objet représenté, se traduit par la création d'un nouvel artefact. Il ne peut donc exister au plus qu'une instance de $C1$ pour un C donné.

Nous pouvons facilement vérifier si un artefact de consentement est la dernière version existante, avec le prédicat

$$\mathsf{lastConsent}(C) = \mathsf{consent}(C, D, PU, T) \wedge \neg(\mathsf{nextConsent}(C, C1, TU))$$

Il est important de noter que $\neg(\mathsf{nextConsent}(C, C1, TU))$ est une négation par échec, le pattern $\mathsf{lastConsent}(C)$ sera vrai si C est un consentement et il n'existe aucun $C1, TU$ tel que $\mathsf{nextConsent}(C, C1, TU)$ existe dans le graphe.

Exemple 29 (Mise-à-jour du consentement) *Dans le graphe de provenance de la Figure 4.5, Bob a mis à jour son consentement une fois, créant un nouvel artefact consent_bob_v1.*
Le pattern $\mathsf{nextConsent}(C, C1, T)$ *peut être instancié avec les valeurs* $C =$ *consent_bob_v0,* $C1 =$ *consent_bob_v1 et* $T = 29$. *Ainsi* $\mathsf{lastConsent}($*consent_bob_v0*$)$ *retournera Faux alors que* $\mathsf{lastConsent}($*consent_bob_v1*$)$ *est Vrai.*

Licéité Comme indiqué dans l'article 5 du RGPD, un artefact a été utilisé légalement s'il a été utilisé à des fins consenties. C'est à dire que chaque processus ayant utilisé la donnée le faisait pour les finalités consenties. Cela implique que le consentement a été donné par la personne concernée et qu'il n'a pas été révoqué au moment où le processus était en cours.

On obtient ainsi deux cas de figures dans lequels le traitement est licite, pour chaque processus P. Tout d'abord quand la donnée traitée n'est pas personnelle, et ne contient pas de données personnelles, c'est-à dire est dérivée uniquement de données avec l'attribut dp à *Faux*. Ceci s'écrit à l'aide des chemins du graphe de provenance

$$(D_dp = False) \vee \Big(\forall D'\big(\mathsf{wasDerivedFrom}^{+}(D, D', TS, TE) \wedge (D'_dp = False)\big)\Big)$$

où D est la donnée traitée.
Le deuxième cas est si la donnée personnelle D a été traitée par un processus P pour une certaine finalité PU et que le consentement pour cette finalité C été valide lors de l'utilisation de la donnée par le processus, ou lorsque toutes les données personnelles D' ayant mené à D ont un consentement valide lors de l'utilisation de D par le processus P pour la finalité PU.

L'expression de chemin donnant la validité d'un consentement d'une donnée personnelle D pour une finalité PU a un certain moment T s'écrit :

$$\begin{array}{ll}\big(\mathsf{consent}(C, D, PU, TG) \wedge (TG < T) \wedge \mathsf{lastConsent}(C) \wedge \neg\big(\mathsf{revoke}(C, TU)\big)\big) & \vee \\ \big(\mathsf{consent}(C, D, PU, TG) \wedge (TG < T) \wedge \mathsf{lastConsent}(C) \wedge \mathsf{revoke}(C, TU) \wedge (TU > T)\big) & \vee \\ \big(\mathsf{consent}(C, D, PU, TG) \wedge (TG < T) \wedge \mathsf{nextConsent}(C, C1, TG1) \wedge (TG1 > T)\big)\Big) & \end{array}$$

Ceci signifie qu'il existe un consentement C ayant été donné avant le temps T ($\mathsf{consent}(C, D, PU, TG) \wedge (TG < T)$) et qu'il n'a pas été révoqué ($\neg(\mathsf{revoke}(C, TU))$), ou du moins pas avant T, ($\mathsf{revoke}(C, TU) \wedge (TU > T)$). Il faut aussi vérifier que les finalités consenties n'ont pas été modifiées $\mathsf{lastConsent}(C)$ ou que si elles ont été modifiées, elles l'aient été après l'action : $nextConsent(C, C1, TG1) \wedge (TG1 > T)$.

On note cette expression de chemin $\mathsf{validConsent}(D, PU, C, T, TG, TU)$.
L'action de P sur un artefact D a donc été effectuée légalement si et seulement le pattern suivant trouve une instanciation dans le graphe de provenance :

$$\Big((D_dp = \texttt{False}) \vee \Big(\forall D'\big(\mathsf{wasDerivedFrom}^+(D, D', TS, TE) \wedge (D'_dp = False)\big)\Big) \quad \vee$$
$$\Big((D_dp = \texttt{True}) \wedge \mathsf{used}(P, D, R, T) \wedge \mathsf{action}(P) = PU \wedge \mathsf{validConsent}(D, PU, C, T, TG, TU)\Big) \quad \vee$$
$$\Big(\mathsf{used}(P, D, R, T) \wedge \mathsf{action}(P) = PU \wedge \big(\forall D'(\mathsf{wasDerivedFrom}^+(D, D', TS, TE) \wedge$$
$$(D'_dp = \texttt{True}) \wedge \mathsf{validConsent}(D', PU, C, T, TG, TU))\big)\Big)$$

Ici P est le processus qui a effectué une action PU sur l'artefact D, C est le consentement de l'artefact contenant le consentement pour effectuer PU, R représente un rôle et $T, TG, TU, TG1$ sont des horodatages. Cette expression de chemin est notée $\mathsf{legal_}DP(P, D, C, TG, TU)$. Une action est légale si pour toutes les données personnelles utilisées par P le consentement a été donné avant que l'action ne soit effectuée. Si la donnée n'est pas de type personnel (première ligne de l'expression), l'action est automatiquement considérée comme conforme à ce principe.

Exemple 30 (Traitement licite) *Dans le scénario de la Figure 4.1, un cookie de type marketing est créé à $T = 21$. Le processus qui le crée createCookie a pour finalité la création de cookies pour des tiers (« thirdParties ») et il a utilisé l'identité de Bob sur ce site (id_bob). Ce cookie a été créé de façon licite si et seulement si une des conjonctions précédentes est vraie, ici la dernière. L'instanciation des variables P = createCookie, PU = thirdParties, D = id_bob, $T = 21$, R = id, C = consent_bob_v0, C = consent_bob_v1, $TG = 17$ et $TG1 = 29$ respecte la dernière conjonction.*

Dans le texte du RGPD, ce principe inclut deux autres notions : la loyauté et la transparence. Toutefois, ces notions ne peuvent être vérifiées automatiquement, car elles concernent la manière de communiquer, ce qui ne peut être capturé par le graphe de provenance.

Limitation de stockage La conservation des données personnelles est réglementée par l'article 5.1(e) du RGPD. Dans cet article, il est stipulé que les données personnelles ne doivent pas être conservées après qu'elles ne soient plus nécessaires (sauf à des fins d'archivage). Les données personnelles ne doivent pas être conservées

sans être utilisées plus longtemps que la période de temps définie par le système, en fonction de ses besoins.

Le système est conforme au principe de limitation de stockage pour une donnée personnelle D si celle-ci n'a pas été utilisée au-delà de la durée maximale de conservation, laquelle est déterminée à partir de la dernière utilisation de la donnée. Autrement dit, il ne doit jamais s'écouler un intervalle de temps supérieur à la durée de conservation entre deux utilisations consécutives de la donnée D. Le pattern exprimant ceci peut s'écrire :

$$\neg\big(\mathsf{used}(P,D,RT) \wedge \mathsf{used}(P',D,R'T') \wedge (T' > T) \wedge (T' - T > TLIMIT)\big)$$

La négation correspond à une négation par échec ce qui signifie que s'il existe une instanciation pour P et P' ayant utilisé D, alors cette expression sera évaluée à faux. Les variables R et R' représentent les rôles liés aux dépendances, sans importance dans ce cas. Entre la dernière utilisation et le moment présent, soit la donnée D a été supprimée (présence de $\mathsf{notAvailable}(D,T)$ dans le graphe de provenance), soit la durée maximale de conservation n'a pas encore été dépassée (la condition de temps $TCURRENT - TU < TLIMIT$ est vraie). Cela garantit que la donnée est soit régulièrement utilisée dans les délais prescrits, soit supprimée une fois que la période de conservation est expirée, assurant ainsi la conformité avec le principe de limitation de stockage.

Un système est alors conforme pour un artefact D représentant une donnée personnelle, utilisé en dernier lieu à l'instant T si le pattern suivant trouve une instanciation :

$$\begin{aligned}\mathsf{storageLimitation}(D) \quad = \quad & \neg\big(\mathsf{used}(P,D,RT1) \wedge \mathsf{used}(P',D,R'T2) \wedge (T2 > T1) \\ & \wedge (T2 - T1 > TLIMIT) \qquad\qquad \wedge \\ & \big(\mathsf{used}(P,D,R,TU) \wedge \neg\big(\mathsf{used}(P',D,R',TU') \wedge (TU < TU')\big) \wedge \\ & (\neg(\mathsf{notAvailable}(D,T)) \wedge (TCURRENT - TU < TLIMIT)) \quad \vee \\ & (\mathsf{notAvailable}(D,T) \wedge (T - TU <= TLIMIT))\end{aligned}$$

L'expression de chemin $\mathsf{used}(P,D,R,TU) \wedge \neg(\mathsf{used}(P',D,R',TU') \wedge (TU < TU'))$ permet de récupérer le processus P et l'horodatage TU de la dernière utilisation de la donnée personnelle D. Le système est conforme à la limitation de stockage pour la donnée personnelle D si la donnée a bien été supprimée dans le délai alloué depuis sa dernière utilisation (dernière conjonction) ou si le délai n'est pas encore terminé représenté par condition de temps ($TCURRENT - TU < TLIMIT$) où $TCURRENT$ est la variable représentant le temps courant de la vérification et $TLIMIT$ est la période de temps définie par le système. L'exemple suivant illustre ce pattern en se basant sur le scénario présenté dans la Section 4.4.

Exemple 31 (Limitation de stockage des données personnelles) *Nous utilisons l'Exemple 25. Dans les deux traces, nous pouvons trouver dans le graphe le chemin*

$$\mathsf{used}(P, e-mail_alice, R, TU) \wedge \neg(\mathsf{used}(P', e-mail_alice, R', TU') \wedge (TU < TU'))$$

avec P étant une instance du processus créant un avis. Dans le cas de la trace conforme, la dépendance notAvailable$(e - mail_alice, T)$ *apparaît avec* $T = TU + TLIMIT$ *(dans le scénario* $TLIMIT = 5$ *ans). Le pattern correspond à un chemin dans le graphe. Dans la cas de la trace non conforme, si la vérification se fait avant qu'Alice tente de se reconnecter à son compte, l'arrête* notAvailable$(e - mail_alice, T)$ *n'existe pas et et la condition* $TCURRENT - TU < TLIMIT$ *est fausse car la date limite a été dépassée. Si la vérification est faite après la connexion d'Alice, c'est le premier ensemble de conjonctions qui sera évalué à faux.*

Il est important de noter que ces patterns ne retournent que vrai ou faux selon si une instanciation a été trouvé dans le graphe. Il faut cependant faire une distinction entre la conformité réelle d'un processus ou du traitement d'une donnée, c'est à dire que cela restera vrai dans le futur, et l'absence de non-conformité car le délai d'action n'est pas arrivé à expiration. Dans ce deuxième cas, il est nécessaire de réaliser une vérification a posteriori. Par exemple, si la vérification dans l'exemple précédent avait été effectuée trois ans après la dernière utilisation de l'e-mail d'Alice, il s'agirait d'un cas de conformité. Il sera alors nécessaire de vérifier ultérieurement que la suppression de l'e-mail a bien eu lieu dans les délais impartis, ou que les données ont été réutilisées avant l'échéance de la période de conservation.

4.5.2.2. Droit des utilisateurs

De la même façon que certains des articles se référant aux principes peuvent avoir une vérification automatisée à l'aide des patterns, il est de même pour certains des droits, ou du moins certaines parties des articles de ces droits. Ici nous donnons l'exemple avec deux droits : le droit à l'oubli et le droit d'accès. Comme pour les principes, les patterns peuvent donner qu'un cas d'absence de non-conformité, nécessitant des vérifications dans le futur. De façon semblable, certains articles ont besoin d'une vérification supplémentaire manuelle, en particulier lorsqu'ils comportent des conditions sur le contenu à transmettre à une personne ou sur la quantification et expressivité des explications et messages à fournir.

Droit à l'oubli Conformément à l'article 17a du RGPD, une personne concernée a le droit de demander la suppression de certaines données, et le responsable du traitement doit le faire avant un certain délai.

Lorsque la personne concernée U demande la suppression d'un artefact D, on retrouve dans le graphe de provenance le chemin suivant :

$$\text{used}(P, D, R, T) \wedge \text{wasControlledBy}(P, U, 'owner', TB, TE)$$

où T, TB, TE sont les horodatages, l'action du processus P est "*askErase*".

Nous pouvons alors définir le pattern $askErase$ comme :

$$\begin{aligned}\mathsf{askErase}(D,T) \quad = \quad & \mathsf{used}(P,D,R,T) \wedge \mathsf{action}(P,"askErase") \wedge \\ & \neg\big(\mathsf{used}(P',D,R,T') \wedge \mathsf{action}(P"askErase") \wedge (T' < T)\big)\end{aligned}$$

Un utilisateur peut demander la suppression plusieurs fois avant de recevoir une réponse. Pour vérifier la conformité il est donc nécessaire de récupérer la date de la première demande.

Un système est conforme à l'article 17a si et seulement si, la suppression de l'artefact est effectuée avant la date limite autorisée. Il est important de noter qu'il faut que la donnée soit supprimée ainsi que toutes les copies de cette donnée. La suppression de la donnée D est représentée par l'expression de chemin :

$$\mathsf{askErase}(D,T) \wedge \mathsf{notAvailable}(D,TU) \wedge (TU - T < TLIMIT)$$

avec $TLIMIT$ étant la date limite pour effectuer l'action de façon conforme. L'expression de chemin pour les données copiées est la suivante :

$$\begin{aligned}\mathsf{askErase}(D,T) \wedge \forall D'(\mathsf{wasDerivedFrom}^+(D,D',"copy",TU',TG) \wedge \\ \mathsf{notAvailable}(D',TU') \wedge (TU' - T < TLIMIT))\end{aligned}$$

L'effacement de toutes les données est représenté par l'expressions suivante

$$\begin{aligned}\mathsf{eraseComplete}(D) \quad = \quad & \mathsf{askErase}(D,T) \wedge \mathsf{notAvailable}(D,TU) \wedge (TU - T < TLIMIT) \wedge \\ & \forall D'(\mathsf{wasDerivedFrom}^+(D,D',"copy",TU',TG) \wedge \\ & \mathsf{notAvailable}(D',TU') \wedge (TU' - T < TLIMIT))\end{aligned}$$

Cependant, comme évoqué précédemment, une vérification de conformité peut être effectuée à tout moment. Nous pouvons avoir le cas où une demande d'effacement n'est pas encore satisfaite, mais la date limite est encore dans le futur.

$$\begin{aligned}\mathsf{eraseNotDoneYet}(D) \quad = \quad & \mathsf{askErase}(D,T) \wedge \neg(\mathsf{notAvailable}(D,TU)) \wedge \\ & (TCURRENT - T < TLIMIT)\end{aligned}$$

où $TCURRENT$ représente la date à laquelle la vérification est effectuée.

Le pattern pour vérifier la conformité du droit à l'oubli pour une certaine donnée personnelle D :

$$\begin{aligned}\mathsf{eraseCompliant}(D) \quad = \quad & \mathsf{askErase}(D,T) \wedge \mathsf{eraseComplete}(D)\, W \quad \vee \\ & \mathsf{askErase}(D,T) \wedge \mathsf{eraseNotDoneYet}(D)\end{aligned}$$

L'exemple suivant illustre le fonctionnement de ce pattern.

Exemple 32 (Droit à l'oubli) *Supposons que dans le système présenté en Figure 4.1, le responsable du traitement dispose de 40 jours pour procéder à l'effacement d'une donnée personnelle lorsque la demande a été faite, donc* $TLIMIT = 40 \times 24 \times 60 = 57600$ *(les horodatages sont donnés en minutes). Après avoir reçu le rapport sur ses données per-*

sonnelles dans le forum, Bob demande à supprimer son numéro de téléphone. Comme on voit sur le graphe de provenance, son numéro de téléphone. Le pattern peut s'instancier comme suit eraseCompliant(*phone_bob*) *si et seulement* askErase(*phone_bob*, T) ∧ eraseNotDoneYet(*phone_bob*) *existe. Au moment de la vérification de la conformité, 12 jours se sont écoulés depuis la réception du rapport donc* $TCURRENT = 61983$. *L'expression de chemin*

$$\mathsf{askErase}(D, T)$$

avec D = *phone_bob et* $T = 44800$ *existe,* eraseNotDoneYet(*phone_bob*) *aussi car* $TCURRENT - T = 17183 < TLIMIT$. *Le principe est respecté, mais une vérification a posteriori devra être réalisé pour confirmer que la demande a été remplie dans les temps.*

Droit d'accès Le RGPD confère à la personne concernée le droit d'accéder à toutes les données d'un système et à la manière dont ses informations personnelles ont été utilisées. L'article 15.1 donne les détails de la manière dont le responsable du traitement des données d'un système doit réagir. Ce point de vérification ne peut être que partiellement automatisé car les journaux d'événements ne contiennent pas les informations contenues dans un document. Ici, les prédicats vérifient qu'une réponse a été envoyée à la personne concernée après sa demande, mais pas si toutes les informations se trouvent dans la réponse.

Comme pour le droit à l'oubli, une demande d'accès a été effectuée si le chemin suivant est présente dans le graphe :

$$\begin{aligned}\mathsf{askAccess}(U, TB, TE) \quad = \quad & \mathsf{wasControlledBy}(P, U, \text{"}owner\text{"}, TB, TE) \wedge \\ & \mathsf{action}(P, \text{"}askAccess\text{"}) \wedge \\ & \neg\big(\mathsf{wasControlledBy}(P', U, \text{"}owner\text{"}, TB', TE') \wedge \\ & \mathsf{action}(P\,\text{"}askAccess\text{"}) \wedge (TE' < TE)\big)\end{aligned}$$

Le pattern askAccess(U, TB, TE) récupère le temps de la première demande d'accès d'un personne concernée U.

Le système est conforme si le responsable du traitement S' répond à la demande d'accès de la personne concernée U dans le délai imparti $TLIMIT$ si l'un des deux conjonctions du pattern trouve une instanciation dans le graphe de provenance :

$$\begin{aligned}\mathsf{rigthAccess}(U) = \quad & \big(\mathsf{askAccess}(U, TB, TE) \wedge \mathsf{wasControlledBy}(P', S', \text{"}owner\text{"}, TB', TE') \\ & \wedge \mathsf{action}(P', \text{"}sendData\text{"})(TE' - TE < TLIMIT)\big) \vee \\ & \big(\mathsf{askAccess}(U, TB, TE) \wedge (TCURRENT - TE < TLIMIT)\big)\end{aligned}$$

P et P' sont des processus, et TB, TB', TE, TE' sont des horodatages. On note ce pattern rigthAccess(U) et l'exemple suivant illustre son fonctionnement avec le scénario de la Figure 4.1.

Exemple 33 (Droit d'accès) *Supposons que dans le cas du forum, lorsque le responsable du traitement reçoit une demande d'accès aux données, il dispose d'un délai de 30 jours pour répondre à la demande de la personne concernée, donc* $TLIMIT = 43200$.

Le graphe montre que Bob a reçu une réponse data_report ($TE' = 44730$) à sa demande data_request ($TE = 37$), ce qui correspond à la première conjonction du pattern. L'expression de chemin est respectée, cependant la condition sur le temps n'est pas respectée : Bob a reçu la réponse 44693 *minutes (environ 32 jours) après sa demande ce qui est supérieur au temps limite.*

Le droit d'accès de Bob n'a pas été respecté.

Les expressions de chemins et donc les patterns permettent de vérifier pour un artefact représentant une donnée personnelle, pour un processus ou pour une personne concernée la conformité à certains principes ou droits du RGPD. Cependant, ils ne sont pas suffisants pour extraire les nœuds du graphe pour lesquels il est nécessaire une vérification plus approfondie ou dans le futur. C'est pour répondre à cette idée que nous avons développé un prototype permettant de choisir les principes et droits à vérifier, ainsi que d'avoir un retour sur les vérifications à approfondir ou à revérifier dans le futur. Ce prototype doit permettre aussi de récupérer les données nécessaires en cas de non-conformité pour une analyse détaillée des auditeurs.

4.6. Prototype et validation

Dans la Section 4.4, nous avons mis en avant les problématiques liés à l'automatisation de vérifications qui peuvent durer dans le temps ou qui nécessitent une vérification supplémentaire (vérifier le contenu d'une notification par exemple). Ces problèmes ne peuvent pas être directement traités par les expressions de chemins ou les conditions temporelles, mais ils peuvent être résolus au niveau de l'implémentation de l'outil de vérification.

Dans ce contexte, nous avons développé un outil en Java conçu pour vérifier partiellement la conformité des systèmes au RGPD, en implémentant les patterns décrits dans la section précédente à l'aide de Prolog. Grâce à cet outil, un auditeur peut obtenir des détails sur les non-conformités détectées ainsi que des avertissements pour des vérifications supplémentaires. L'outil permet également de sélectionner les principes à vérifier, de choisir les données et les processus à analyser, offrant ainsi la possibilité de cibler l'audit de manière précise. Une version préliminaire de cet outil a été présentée dans l'article Di Salvo-Cilia et al. 2024.

Dans cette section, nous présentons d'abord l'architecture du prototype en détaillant son fonctionnement et les différentes interactions entre les modules ainsi que les adaptations faites aux patterns pour obtenir des retours précis (Section 4.6.1). Nous donnons en Section 4.6.2 une validation de nos patterns à l'aide de ce prototype, sur des scénarios comme ceux présentés dans les Sections 4.4 et 4.5, mais aussi sur des scénarios avec un nombre de nœuds plus important, générés à l'aide d'un générateur de graphes. Les pistes d'amélioration de cet outil sont évoquées en Section 4.6.3.

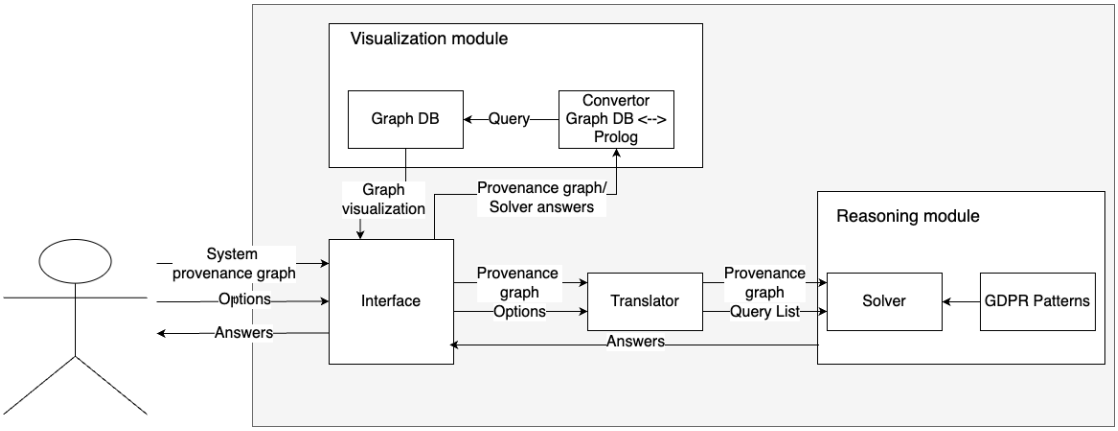


FIGURE 4.2. – Architecture de l'outil de vérification de la conformité

4.6.1. Architecture

Les principaux composants du prototype sont représentés en Figure 4.2. A l'aide de l'interface, l'auditeur peut fournir le système à auditer et préciser les principes, les données personnelles ou personnes concernées à vérifier. Ces options vont être traduites en requêtes Prolog par le traducteur. Le solveur va interroger le graphe de provenance qui représente le système et renvoyer la réponse à l'interface, qui l'affichera sous forme de texte ou de graphe à l'utilisateur. La réponse affichée contient le détail des non-conformités trouvées durant la vérification.

Interface Via l'interface, un auditeur peut spécifier le système qu'il souhaite vérifier et éventuellement un sous-ensemble de processus (ou un sous-ensemble de données personnelles) sur lequel il souhaite se concentrer. Il peut aussi préciser les personnes concernées sur lesquelles on veut vérifier le respect de leurs droits. L'interface récupère les fichiers journaux du système sous la forme d'un graphe de provenance [4]. L'auditeur doit donc donner le graphe sous forme de fichier Prolog. Il a aussi la possibilité via cette interface de préciser les données personnelles, les utilisateurs ou les processus spécifiques à vérifier ainsi que les principes s'il désire se concentrer sur un sous-ensemble du système plutôt que de faire une vérification globale (par défaut, l'ensemble du système est audité). Ces paramètres sont envoyés au module traducteur (« Translator » dans la Figure 4.2). Les choix donnés à l'auditeur sont présentés sous la forme d'un menu comme illustré dans la Figure 4.3a.

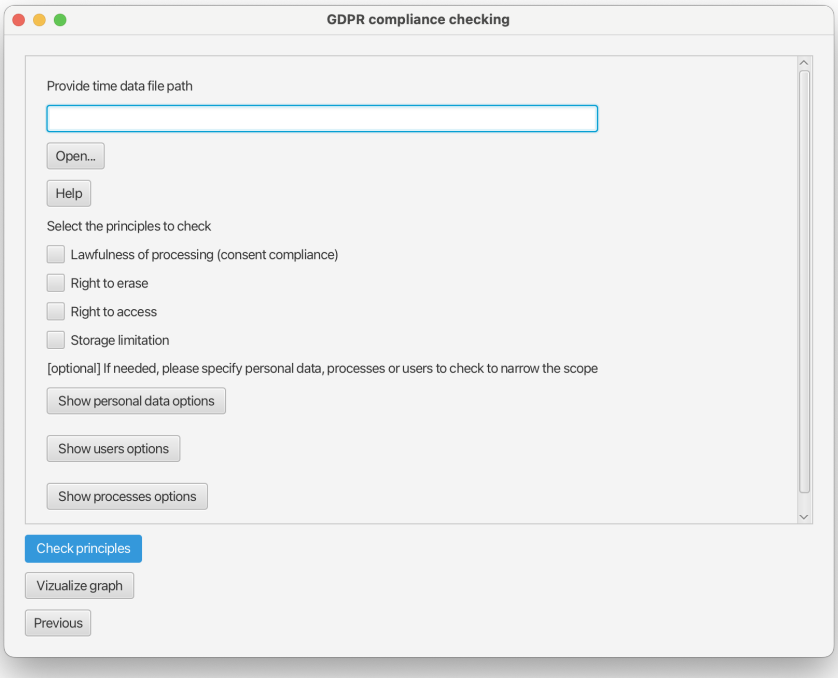
L'interface affiche les résultats obtenus par le solveur, sous la forme de texte (Figure 4.3b). Ce module interagit aussi avec le module de visualisation, permettant à l'utilisateur de visualiser le graphe de provenance du système ou alors les réponses du solveur de forme graphique (Figure 4.4). L'interface a été développée sous Java et utilise comme librairie graphique JavaFX [5].

4. Comme dans les prototypes des Chapitres 2 et 3, on n'implémente pas le système de traduction des journaux d'évènements en graphe de provenance.

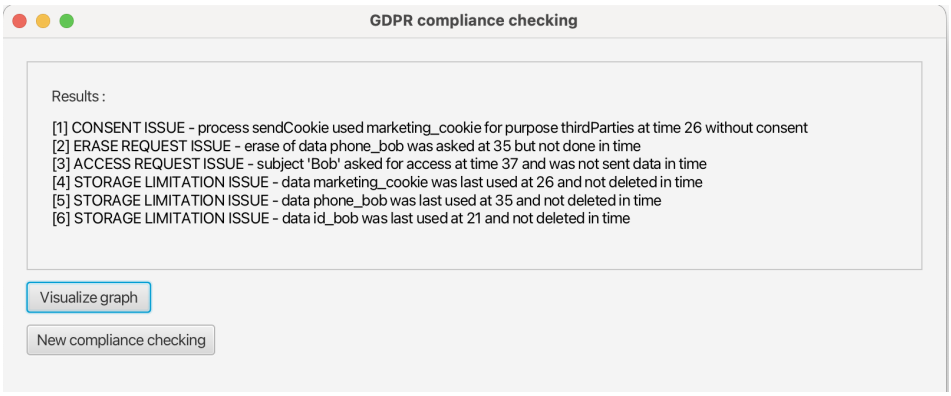
5. https://openjfx.io/, consulté en septembre 2024

Traducteur Le traducteur remplit un double rôle. D'une part il permet de traduire les spécifications renseignées par l'auditeur en requêtes Prolog paramétrées. D'autre part, il permet d'extraire du graphe de provenance les données personnelles, les personnes concernées ainsi que les processus. Ces entités sont nécessaires lors de la proposition des choix à l'auditeur. Ils vont aussi permettre d'instancier les variables dans les requêtes Prolog. Selon les principes et les nœuds sélectionnés, le traducteur envoie une liste de requêtes au solveur Prolog ainsi que les données système, en utilisant la bibliothèque JPL[6].

6. https://jpl7.org/, consulté en septembre 2024



(a) Fenêtre de l'interface présentant les options



(b) Fenêtre de l'interface donnant les résultats

FIGURE 4.3. – Interface du prototype

Par exemple, en se basant sur le scénario présenté dans la Figure 4.1, si un auditeur veut vérifier que l'ensemble des principes concernant l'e-mail de bob, c'est-à-dire la licéité de son traitement, le droit à l'oubli et la limite de stockage, le module traducteur

crée les requêtes suivantes [7] :

$$\begin{aligned}&\mathsf{legal}(P,'\mathtt{e-mail_bob}',C,TG,T).\\&\mathsf{eraseComplete}('\mathtt{e-mail_bob}').\\&\mathsf{storageLimitation}('\mathtt{e-mail_bob}').\end{aligned}$$

Dans le cas du principe de liécité, le principe sera vérifié pour tous les processus P utilisant l'e-mail de Bob. En plus de la liste des requêtes et du graphe de provenance à vérifier, le liste des processus sera envoyé pour que le solveur puisse intancier P avec ces données.

Module de raisonnement Le module de raisonnement contient le solveur Prolog, qui résout les requêtes de chemin en fonction du graphe de provenance renseigné, et la spécification des prédicats RGPD, correspondant aux définitions des patterns des principes et des droits formalisés dans la section précédente.

Chaque principe RGPD est codé sous forme de règle Prolog. Dans le but de récupérer les informations nécessaires pour notifier l'auditeur de la vérification future ou de la non-conformité, pour chaque principe et droit, on décompose chaque prédicat en conjonctions et disjonctions de sous-prédicats. Ces prédicats sont de la forme[7] :

```
prédicat(paramètres):-
    (vérification paramètres,
      (vérification conformité;
      (vérification délai, affichage vérification future);
      (\+ (vérification conformité),\+(vérification délai),
          affichage donnée non conforme))
    (\+(vérification paramètres), affichage pas de paramètres).
```

L'idée est de sectionner les patterns présentés dans la Section 4.5 pour obtenir des retours interprétables par l'auditeur. La définition du prédicat de chaque principe ou droit RGPD contient donc plusieurs règles :

— Le sous-prédicat `vérification paramètres` permet de limiter la vérification aux nœuds concernés. Par exemple, le principe de licéité ne s'applique qu'aux données personnelles. $\mathsf{used}(P,D,R,T), \mathsf{action}(P) = PU, isPersonal(D)$ [8]. Si D n'est pas une donnée personnelle ou ne contient pas d'informations personnelles, le prédicat $legal(P,D,C,TG,T)$ ne sera pas appliqué à cet ensemble D. De même, si un processus P de la liste des processus à vérifier du système n'utilise pas D, le prédicat ne sera pas appliqué non plus.

— Les sous-prédicats `vérification conformité` et `vérification délai` sont utilisés pour vérifier la conformité. Ils intègrent le cœur des patterns présentés dans la section précédente. Le premier sous-prédicat inclut la partie du pattern qui vérifie si une action est conforme (par exemple, si elle a été traitée dans

7. Les requêtes et prédicats sont écrits en Prolog. Une rapide introduction au Prolog et sa syntaxe sont données en Annexe B.

8. La règle $isPersonal(D)$ est évaluée à vrai si $dp(D) = True$.

les délais) et le second sous-prédicat concerne l'absence de non-conformité (l'action n'a pas encore été réalisée, mais le délai n'a pas expiré). Dans le cas de `vérification délai`, un message est renvoyé via le prédicat `affichage vérification future`. Ce dernier sert uniquement à fournir un retour détaillé et ne procède à aucune vérification. Le prédicat `vérification délai` n'est pas présent pour tous les principes, par exemple, la licéité ne comporte pas de délai de vérification dans sa définition. En revanche, le droit à l'oubli et le droit d'accès incluent ce prédicat.

— Le sous-prédicat `affichage donnée non conforme` permet d'afficher les informations nécessaires à l'auditeur lorsqu'il y a absence de conformité : il indique quelle donnée, quel processus, et/ou quel principe ou droit a été violé. Il permet également de renvoyer *faux* si cette partie du prédicat est vérifiée.
— Enfin, le sous-prédicat `affichage pas de paramètres` permet de récupérer des informations lorsque les nœuds instanciés ne sont pas affectés par ce principe.

Le prédicat vérifie donc la validité des paramètres avec lesquels certaines variables ont été instanciées. Si les paramètres ne sont pas applicables, on passe directement à la dernière étape et récupère le détail via `affichage pas de paramètres`.

Si les paramètres sont valides, le prédicat renvoie *vrai* uniquement si `vérification conformité` ou `vérification délai` est vérifié (en incluant les informations pour ce dernier). Dans le cas contraire, on affiche les informations via `affichage donnée non conforme` et le résultat sera *faux*. Un exemple du prédicat Prolog pour le droit à l'oubli est présenté dans l'exemple suivant

Exemple 34 (Fichier prolog du prédicat pour le droit à l'oubli) *Le prédicat permettant de vérifier la conformité de la suppression pour une donnée personnelle D est implémenté comme suit :*

```
% message no data erase query by the data controller
writeNoEraseAsked() :- writeln('system compliant for data erasure as no erase
    was asked').
% message if data non compliant
writeEraseNotCompliant(D, T) :- write('erase not compliant : erase of data
    '), write(D), write(' was asked at '), write(T), writeln(' but not done in
    time'), false.
% message if deadline has not expired
writeEraseNotExpired(D, T, TLIMITERASE) :- write('erase not done yet :
    erase of data '), write(D), write(' was asked at '), write(T), writeln('
    but deadline not expired, expire in'), write(T+TLIMITERASE).
% patterns of compliance
askErase(D, T) :- used(P, D, _R, T), action(P, 'askErase').
eraseComplete(D) :- askErase(D, T), used(P, D, _R, TU), action(P,
    'delete'), (tLimit('erase', TLIMITERASE), (T-TU)<TLIMITERASE).
eraseNotDoneYet(D):- askErase(D,T),\+ (used(P,D,_-
    R,_TU), action(P,'delete')), (tCurrent(TCURRENT),
    tLimit('erase',TLIMITERASE), (TCURRENT-T)<TLIMITERASE).

eraseCompliant(D):- (askErase(D, T),
(eraseComplete(D,!);
 (eraseNotDoneYet(D),writeEraseNotExpired(D,T,TLIMITERASE),!)
(\+ dataErasureOk(D),writeEraseNotCompliant(D,T))));
 (\+ askErase(D,_T), writeNoEraseAsked()).
```

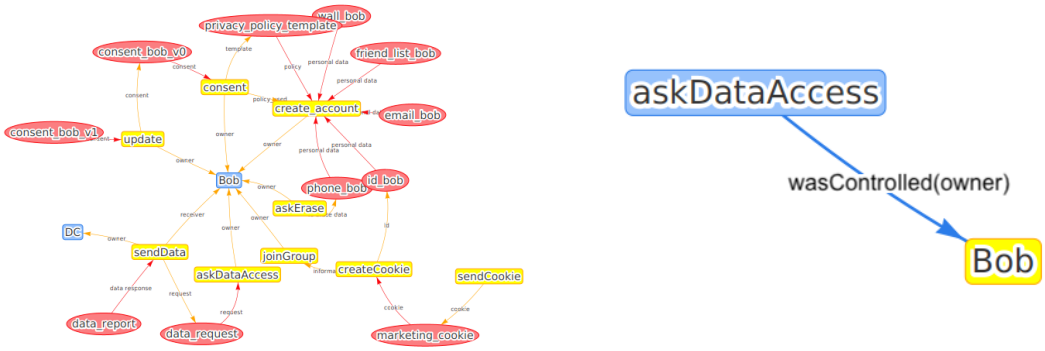
Le résultat ou les résultats sont envoyés à l'interface qui affichera le résultat à l'auditeur.

Visualisation Le module de visualisation a pour but de faciliter la compréhension du retour du solveur, en fournissant une représentation graphique du graphe de provenance et des résultats. Il comporte deux éléments : un traducteur permettant de convertir un fichier Prolog en une base de données graphe et inversement ainsi qu'une base de données graphe de type Neo4J Graph Database[9]. Le traducteur établit le lien entre l'interface et la Neo4J Graph Database, permettant de créer un fichier HTML avec les informations envoyées depuis l'interface : cela peut être le graphe entier (Figure 4.4a) ou les sous-graphes non-conformes aux patterns (Figure 4.4b).

4.6.2. Validation

Dans le but de tester notre approche, les patterns ont été validés à l'aide de variations des scénarios présentés en Sections 4.4 et 4.5. Ces variations consistaient à changer les finalités consentis, les horodatages liés à la génération des consentements ou des actions réponses comme l'envoi de données ou la suppression des données.

9. https://neo4j.com/product/neo4j-graph-database/, consulté en septembre 2024.



(a) Affichage du graphe avec l'outil Neo4J

(b) Affichage d'un cas de non-conformité avec Neo4J

FIGURE 4.4. – Interface du prototype

Ces modifications permettent de faire varier les points de non-conformité présents dans les graphes. Les patterns de licéité, limitation du stockage, droit à l'oubli et droit d'accès ont été testé sur 20 graphes différents issus des scénarios précédemment évoqués. Lors de la vérification de la conformité de la globalité du système sur ces graphes, nous avons obtenus les résultats instantanément. L'outil a mis entre 5 et 50 millisecondes pour afficher la réponse sur l'interface.

Dans le but de valider notre approche sur des scénarios comportant plus d'utilisateurs et un plus grand nombre de dépendances, un générateur de graphes a été développé, nous ayant permis de tester sur des scénarios présentant plus d'utilisateurs. Le générateur supporte la génération de graphes comportant jusqu'à 10 000 utilisateurs. Cependant, la vérification dans ces cas-là de la conformité sur tout le système prend un temps trop élevé (plusieurs heures).

Générateur de graphes Afin de produire des graphes de provenance représentant des scénarios réels, la génération de graphe repose sur l'assemblage de sous-graphes (nommées briques dans la suite) représentant l'exécution d'un processus dans un système. Ces briques peuvent être de type général (accéder à une page du site) ou spécifiques à un contexte (acheter un produit). Chaque contexte contient entre 3 et 5 briques spécifiques, ainsi que 10 briques générales. En plus de ces briques, il existe des briques déclenchées, qui peuvent être activées par certaines autres actions. Chaque brique standard capable de déclencher une brique déclenchée contient, à l'intérieur de ses faits, un lien vers le sous-graphe correspondant à cette brique déclenchée. La Figure 4.5 montre comment la brique générale sendMail permettant de définir le sous-graphe qui code pour l'envoi d'un mail inclut la définition de la brique déclenchée sendAnalysisCookie représentant l'envoi d'un cookie d'analyse. Les variables %VAR% sont instanciés lorsqu'elles sont choisies dans la création du graphe et le mot clé CAN explicite que la brique déclenchée n'est pas obligatoirement

ajoutée (elle sera ajoutée avec une probabilité aléatoire comprise entre 0.1 et 1).

```
wasControlledBy('%PROCESS :sendMail%','DC', 'owner', %T%, %TC%).
wasGeneratedBy('message','%PROCESS :sendMail%', 'mail to send', %T%).
used('%PROCESS :sendMail%','%DATA :mail%', 'address to send to', %T%).
%CAN :/TRIGGERED/sendAnalysisCookie%
used('%PROCESS :sendMail%', 'message', 'mail sent', %TF%).
```

FIGURE 4.5. – Brique sendMail

Trois contextes ont été considérés : site web de type forum ou réseau social, de type boutique en ligne et de service publique (par exemple le site de gestion d'eau). Pour chaque contexte, il a été défini les types de données qu'il peut contenir (adresse e-mail, cookie d'analyse, ...) ainsi que les briques disponibles, avec une précision pour les briques qui peuvent être réutilisées par utilisateur. Par exemple, un utilisateur peut se créer un compte qu'une seule fois mais il peut mettre à jour ses informations une multitude de fois.

Le générateur de graphes prend quatre paramètres, dont le premier qui est obligatoire est le nom du fichier de sortie contenant le graphe sous forme de faits Prolog. Les trois autres sont optionnels : le nombre d'utilisateurs (ou un fichier contenant le nom des utilisateurs à ajouter), le contexte voulu et le facteur. Si ces paramètres ne sont pas présents, le générateur prendra des valeurs aléatoires et la valeur du facteur sera égale à 1. Le facteur, supérieur ou égal à 1, permet d'augmenter la taille du graphe, en gérant la répétition des briques qui peuvent être réutilisées. Le facteur 1 n'autorise pas la répétition, et plus la valeur est grande plus il pourra y avoir des répétitions de briques. Les briques sont choisies aléatoirement et instanciées puis ajoutées dans le graphe. A l'aide des briques déclenchées, on s'assure que si une brique est ajoutée alors toutes celles nécessaires à faire cette action sont présentes dans le graphe.

Un exemple de graphe généré avec 20 utilisateurs et facteur 1 dans le contexte service publique et visualisé avec l'outil Neo4J est donné en Figure 4.6.

Résultats avec les graphes générés Pour la validation de notre modèle, nous avons généré des graphes de 20 utilisateurs avec un facteur allant de 1 à 5 sur les trois contextes en raison du temps nécessaire à vérifier la conformité sur l'ensemble du graphe. On a décidé de rester à 5 minutes par graphe pour la vérification. La vérification du graphe montré en Figure 4.6 a nécessité de 18 secondes. Nous avons observé que plus un graphe est grand en termes de dépendances (indiqué par un facteur plus élevé et des briques spécifiques à un contexte avec plus de dépendances, donc une taille de fichier plus importante), plus le temps pour faire la vérification est élevé. Par exemple, pour des fichiers dont la taille est comprise entre 85 et 100 kilo-octets (Ko), le temps moyen de vérification est de 17 secondes. Pour des fichiers de 100 à 110 Ko, ce temps passe à 41 secondes. Enfin, les fichiers de 110 à 120 Ko nécessitent un temps moyen de 1,5 minute pour la vérification.

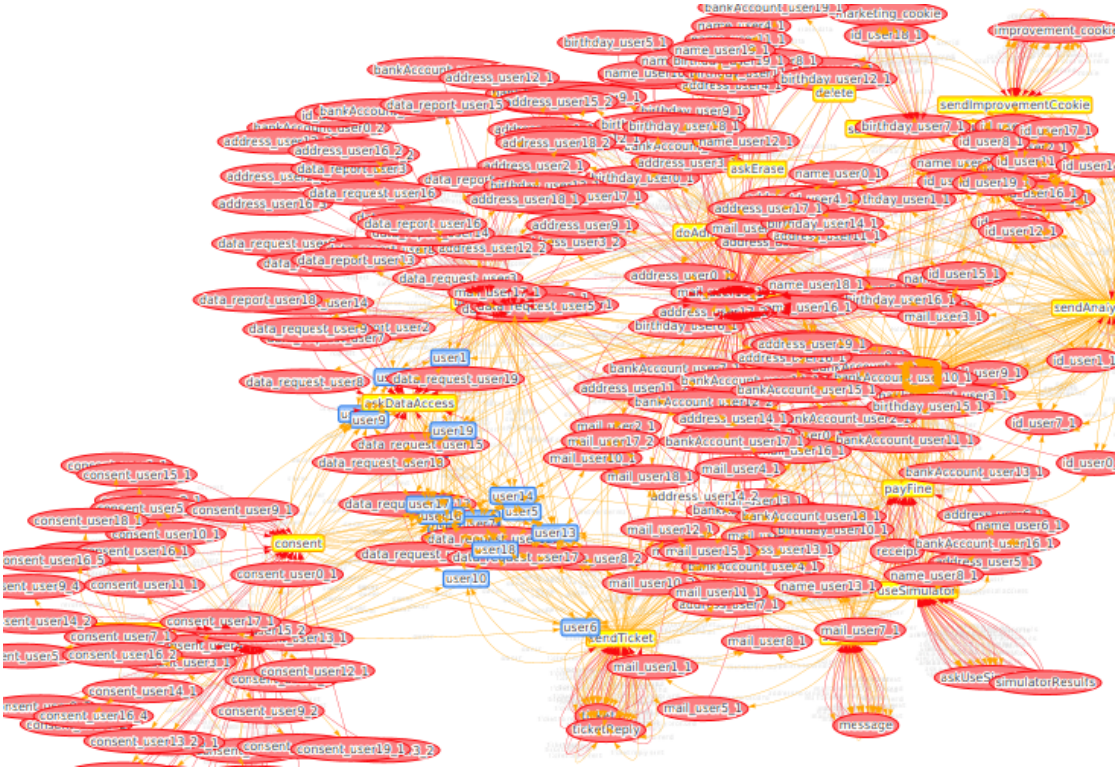


FIGURE 4.6. – Représentation Neo4j d'un extrait de graphe de provenance généré par le générateur avec 20 utilisateurs et facteur 1

On a cependant observé que le temps ne dépend pas uniquement de la taille du graphe globale. En effet, pour des fichiers de même taille, les temps de vérification peuvent varier considérablement, parfois jusqu'au double. Ceci est le cas des graphes représentant des contextes contenant de nombreuses données personnelles. En effet, plus la vérification du prédicat `isPersonal` est nécessaire, plus le temps requis pour assurer la conformité est élevé.

4.6.3. Améliorations et travaux futurs

Plusieurs améliorations sont envisagées sur l'implémentation du prototype et le modèle.

L'un des principaux axes d'amélioration concerne l'implémentation du prototype, en particulier du module visualisation.

Actuellement, l'affichage des réponses émises par le solveur ne donne pas une clarté suffisante, sur le principe ou droit non conforme. Par exemple on ne retrouve pas sur les arêtes les informations liées aux horodatages. Modifier cette visualisation permettrait aussi de fournir une vision plus précise des relations entre les différents éléments. En effet, dans le cas de certains principes. La possibilité d'afficher un sous-graphe plus étendu permettrait de mieux identifier et comprendre les problèmes détectés. Par exemple, en cas d'utilisation d'un cookie à des fins non autorisées, présenter les

données personnelles à partir desquelles ce cookie a été généré permettrait d'offrir à l'auditeur une meilleure précision et une meilleure compréhension.

Un autre point d'amélioration à considérer concerne la transmission des données du système, comme le graphe de provenance et des informations spécifiques comme les délais applicables au système. Actuellement, c'est l'auditeur qui les transmet à l'outil via l'interface. Cette amélioration pourrait permettre la mise en place une vérification périodique pour les avertissements liés à des délais reportés lors de la vérification de la conformité. En effet, périodiquement, l'outil reçoit (par l'auditeur ou de façon automatisée) une version actualisée du graphe de provenance, et vérifie si des avertissements liés à des patterns sont arrivés à la fin du délai autorisé et relance la conformité pour ces patterns-là. Ce qui pourrait mener à une non-conformité (si l'action n'a pas été effectuée dans les temps), à un nouvel avertissement car le délai a été prolongé (par exemple si la donnée a été réutilisée entre temps) ou tout simplement à la disparition de l'avertissement car l'action ou actions nécessaires à garder la conformité ont été réalisées. Cela permettrait de réduire la charge cognitive d'un auditeur en automatisant certaines vérifications.

Pour améliorer notre modèle, en particulier sa complexité en termes de calcul, il serait utile d'étudier l'impact de l'ordre des sous-prédicats Prolog sur les appels au graphe et donc aux performances de l'outil. En effet, Prolog utilise l'unification et le backtracking lorsqu'il y a des conjonctions de sous-prédicats avec des variables libres. Tant qu'il n'a pas testé toutes les options possibles, il ne considère pas qu'un prédicat ne peut pas être satisfait. L'ordre des prédicats dans une disjonction et dans une conjonction peut donc avoir un impact sur le temps nécessaire à évaluer la conformité et la non-conformité, comme on l'a vu durant les tests avec le prédicat `isPersonal`, présent dans certains patterns. Dans cette même perspective, une piste d'amélioration est l'utilisation des requêtes Graph Database pour la partie raisonnement au lieu d'utiliser des requêtes Prolog. Une comparaison des performances permettrait d'évaluer l'efficacité de chacune.

Enfin des améliorations sur le générateur de graphes sont aussi envisagées, pour procéder à des tests plus complets sur notre modèle et prototype. En effet, pour l'instant le générateur de graphes permet de simuler des graphes pour un nombre restreint de types de sites web (forum, boutique en ligne et site web d'information), et comporte un ensemble de briques réduit, permettant d'ajouter aux scénarios qu'une quantité très limitée d'actions différentes. L'augmentation des briques contenues par le générateur, permettrait donc d'augmenter le nombre d'actions pouvant être ajoutées aux graphes, et ainsi étendre les types de systèmes, comme par exemple inclure des sites de divertissement ou de service autre que public.

En outre, étendre le générateur de graphes pour avoir la possibilité de générer des graphes à partir de journaux d'événements réels ou de traces de systèmes, en s'inspirant des méthodes présentées dans la Section 1.5 du Chapitre1, constituerait une extension intéressante. En effet, cette extension permettrait de prendre en compte des scénarios réels, et non plus seulement des scénarios synthétiques.

4.7. Conclusion et discussion

Dans ce chapitre nous avons proposé une modélisation de points clés du RGPD, utilisant les propriétés de la provenance. On propose une représentation de certains principes et droits du RGPD à l'aide de conjonctions et disjonctions de dépendances causales du OPM et de conditions temporelles sur les horodatages ainsi que des conditions sur les attributs des nœuds. Le but était de donner une méthode d'automatisation de la vérification de la conformité de ces principes et droits. Pour atteindre cet objectif, nous complétons notre approche par le prototype d'un outil implémentant les patterns composés de chemins de dépendances causales et permettant ainsi l'automatisation de la vérification de la conformité. Ce prototype permet de vérifier à l'aide d'un solveur Prolog le respect partiel du RGPD, et donne une réponse compréhensible par un auditeur, qu'elle soit sous forme d'explication ou sous forme graphique. Cependant les retours pourraient être améliorés par une modification des informations affichées lors du retour ainsi que par une optimisation de l'ordre des règles dans la définition des prédicats. Pour démontrer le fonctionnement de nos patterns et la faisabilité de notre approche, nous avons testé ces patterns sur des scénarios de petite et de grande envergure à l'aide d'un générateur de graphes. Ce générateur permet de créer des graphes de provenance, intégrant ou non des non-conformités, en se basant sur des actions spécifiques à chaque type de système. L'utilisation de journaux de systèmes réels pour générer les graphes de provenance est envisagée comme amélioration future.

Les patterns de conformité peuvent être étendus à d'autres articles du RGPD ainsi qu'à d'autres législations régissant le traitement des données. Par exemple, le HIPAA impose une durée de conservation des dossier médicaux. Cette condition sur la conservation ressemble à l'expression de chemin du droit à l'oubli, permettant ainsi de l'exprimer sous la forme d'une expression de chemin combinée à des conditions temporelles. De même, la ressemblance des articles du RGPD-UK à ceux du RGPD permettraient une adaptation triviale de nos patterns pour couvrir ce règlement aussi. Une limite est cependant toujours présente quand il s'agit de vérifier le contenu exact, par exemple il peut être compliqué de juger la minimisation de données. Ceci pourrait être combiné avec des spécifications précises des procédures et des processus et vérifier la conformité vis-à-vis de ces documents.

Conclusion

Dans cette thèse nous nous sommes intéressés à la protection des donnés personnelles et de la vie privée en utilisant la provenance de ces données.

Dans un contexte de collaboration accrue entre utilisateurs et de numérisation des traitements des données, la gestion des accès aux données personnelles s'est révélée être un enjeu crucial. En particulier, les systèmes collaboratifs posent des défis significatifs en matière de sécurité et de protection de la vie privée. Les relations interpersonnelles et les interactions au sein de ces systèmes compliquent davantage la gestion des droits d'accès, rendant nécessaires des mécanismes robustes pour prévenir garantir une protection adaptée aux utilisateurs.

Parallèlement, le cadre réglementaire imposé par le Règlement Général sur la Protection des Données (RGPD) a renforcé l'exigence de conformité dans le traitement des données personnelles. Cette réglementation souligne l'importance de la transparence et de la responsabilité dans la gestion des informations sensibles. Il est donc essentiel de développer des outils permettant de vérifier automatiquement la conformité aux exigences du RGPD.

Dans ce mémoire nous avons présenté un modèle de contrôle d'accès et une stratégie de gestion de conflits permettant un meilleur contrôle des informations partagés sur des systèmes type réseaux sociaux, ainsi qu'un outil d'automatisation de la conformité du RGPD.

Pour parvenir à ces contributions nous avons utilisé la provenance des données représentée à l'aide du Modèle Ouvert de Provenance (OPM), que nous avons étendu pour prendre compte des spécificités dynamiques des systèmes étudiés, ainsi que pour pouvoir prendre en compte les interactions entre les différents agents des systèmes.

Le contrôle d'accès proposé utilise les dépendances entre les différents éléments du contenu ainsi qu'entre les relations présentes et passées des utilisateurs donnant la possibilité aux contrôleurs des données d'exprimer de façon fine leurs préférences au sein de politiques. Une stratégie de gestion de conflit est mise en place afin de pouvoir combiner ces politiques en trouvant un équilibre entre la sensibilité des objets et les avantages à donner l'accès. Pour démontrer la faisabilité de notre approche, nous avons développé deux prototypes de type preuve de concept. Ces prototypes nous permettent de simuler le contenu d'un réseau social, de tester notre algorithme et notre stratégie, ainsi que d'évaluer les performances et de comparer nos résultats sur de nombreux scénarios avec ceux d'autres modèles.

De plus, nous avons proposé une représentation des articles du RGPD définissant des règles sur comment traiter les données personnelles à l'aide de la provenance

et de conditions temporelles. Pour parvenir à l'automatisation nous proposons un prototype d'un outil permettant de vérifier à l'aide de Prolog la conformité des traces d'exécution d'un système.

Bien que notre système de contrôle d'accès puisse s'adapter à d'autres systèmes collaboratifs que les réseaux sociaux, notamment dans les contextes de chaînes d'approvisionnement et de gestionnaires de version, nous avons constaté que d'autres travaux sur les workflows proposent des solutions de contrôle d'accès tout aussi adaptées. Par manque de temps, une étude avec des utilisateurs réels n'a pas pu être mené sur le contrôle d'accès ou la gestion de conflits ne permettant pas une validation dans un cadre réel sur l'utilité et l'adaptation de notre approche aux besoins réels des utilisateurs, en particulier sur la gestion de conflit. Enfin, en ce qui concerne l'outil de vérification automatisée de la conformité au RGPD, nous avons rencontré des difficultés à étendre nos patterns pour aborder certains aspects, car ces derniers traitent soit du contenu, soit de points qui ne concernent pas directement le traitement des données. Cela met en évidence la complexité inhérente à la conformité réglementaire et la nécessité de développer des solutions plus flexibles et intégrées.

Afin d'enrichir notre approche, plusieurs pistes peuvent être envisagées, notamment l'extension de notre contrôle d'accès aux systèmes IoT grâce à la gestion des relations transitoires introduite dans ce mémoire. Cela nécessitera des améliorations dans les choix d'implémentation afin de réduire la consommation de ressources lors du calcul des droits d'accès. De plus, mener des études auprès des utilisateurs pourrait fournir une validation solide de notre approche, tant pour le contrôle d'accès que pour la gestion des conflits. Cette démarche permettrait non seulement de confirmer l'efficacité de notre système, mais aussi d'adapter nos solutions aux besoins réels des utilisateurs.

Enfin, l'outil de conformité pourrait bénéficier d'améliorations en matière d'implémentation pour simplifier davantage le travail des auditeurs. Il serait également pertinent d'étendre les patterns supportés afin d'inclure d'autres articles du RGPD, tout en ouvrant la possibilité de vérifier la conformité avec d'autres systèmes réglementaires. Ces évolutions permettraient non seulement d'optimiser l'efficacité de l'outil, mais aussi d'accroître sa polyvalence et son utilité dans divers contextes réglementaires.

Bibliographie

AGOSTINELLI, Simone, Fabrizio Maria MAGGI, Andrea MARRELLA et al. (2019). « Achieving GDPR Compliance of BPMN Process Models ». In : *Information Systems Engineering in Responsible Information Systems*. Sous la dir. de Cinzia CAPPIELLO et Marcela RUIZ. Cham : Springer International Publishing, p. 10-22. ISBN : 978-3-030-21297-1 (cf. p. 137).

AHN, Gail-Joon, Jing JIN et Mohamed SHEHAB (mars 2012). « Policy-Driven Role-Based Access Management for Ad-Hoc Collaboration ». In : *J. Comput. Secur.* 20.2–3, p. 223-257. ISSN : 0926-227X (cf. p. 60).

AKTOUDIANAKIS, Evangelos, Jason CRAMPTON, Steve SCHNEIDER et al. (2013). « Policy templates for relationship-based access control ». In : *Annual Conference on Privacy, Security and Trust*. IEEE, p. 221-228 (cf. p. 59).

ALSHAREEF, Hanaa, Raúl PARDO, Gerardo SCHNEIDER et al. (2020). « A collaborative access control framework for online social networks ». In : *Journal of Logical and Algebraic Methods in Programming* 114, p. 100562. ISSN : 2352-2208. DOI : https://doi.org/10.1016/j.jlamp.2020.100562. URL : https://www.sciencedirect.com/science/article/pii/S235222082030047X (cf. p. 102).

ARFELT, Emma, David BASIN et Søren DEBOIS (2019). « Monitoring the GDPR ». In : *Computer Security – ESORICS 2019*. Sous la dir. de Kazue SAKO, Steve SCHNEIDER et Peter Y. A. RYAN. Cham : Springer International Publishing, p. 681-699. ISBN : 978-3-030-29959-0 (cf. p. 137, 138).

ARORA, Chahal, Syed Zain R. RIZVI et Philip W.L. FONG (2022). « Higher-Order Relationship-Based Access Control : A Temporal Instantiation with IoT Applications ». In : *Symposium on Access Control Models and Technologies*. SACMAT '22. New York, NY, USA : ACM, p. 223-234. ISBN : 9781450393577 (cf. p. 61).

ASIM, Yousra et Ahmad Kamran MALIK (2016). « A Survey on Access Control Techniques for Social Networks ». In : *Innovative Solutions for Access Control Management*. IGI Global, p. 1-32. DOI : 10.4018/978-1-5225-0448-1.ch001. URL : http://dx.doi.org/10.4018/978-1-5225-0448-1.ch001 (cf. p. 58).

ATHAN, Theodore, Guido GOVERNATORI, Monica PALMIRANI et al. (2015). « LegalRuleML : Design Principles and Foundations ». In : *Reasoning Web. Web Logic Rules 2015*. T. 9203. Lecture Notes in Computer Science. Heidelberg : Springer, p. 151-188 (cf. p. 136).

AVESANI, Paolo, Paolo MASSA et Roberto TIELLA (2005). « A trust-enhanced recommender system application : Moleskiing ». In : *ACM Symposium on Applied Computing*. URL : https://api.semanticscholar.org/CorpusID:5219798 (cf. p. 50).

BAFOUTSOU, Georgia et Gregoris MENTZAS (2002). « Review and functional classification of collaborative systems ». In : *International Journal of Information Manage-*

ment 22.4, p. 281-305. ISSN : 0268-4012. DOI : https://doi.org/10.1016/S0268-4012(02)00013-0. URL : https://www.sciencedirect.com/science/article/pii/S0268401202000130 (cf. p. 25).

BAHRI, Leila, Barbara CARMINATI et Elena FERRARI (2015). « CARDS - Collaborative Audit and Report Data Sharing for A-Posteriori Access Control in DOSNs ». In : *Conference on Collaboration and Internet Computing (CIC)*. IEEE, p. 36-45 (cf. p. 61).

BARAB, Sasha A., Rob KLING et James H. GRAY (2004). *Designing for Virtual Communities in the Service of Learning*. A community is "a persistent, sustained [social-technical] network of individuals who share and develop an overlapping knowledge base, set of beliefs, values, history, and experiences". Cambridge, UK : Cambridge University Press (cf. p. 26, 48).

BARATI, Masoud, Omer RANA, Ioan PETRI et al. (2020). « GDPR Compliance Verification in Internet of Things ». In : *IEEE Access* 8, p. 119697-119709. DOI : 10.1109/ACCESS.2020.3005509 (cf. p. 137).

BARBERIO, Vitaliano, Ines KURIC, Markus HÖLLERER et al. (déc. 2017). *PERCEIVE project - Deliverable D3.3 "Descriptive report on the specific role of new media in EU financed projects' communication strategies"*. Technical Report. URL : http://amsacta.unibo.it/5763/ (cf. p. 95).

BARTOLINI, Cesare, Gabriele LENZINI et Cristiana SANTOS (2018). « A Legal Validation of a Formal Representation of Articles of the GDPR ». In : *TERECOM@JURIX* (cf. p. 136).

BASIN, David, Søren DEBOIS et Thomas HILDEBRANDT (2018). « On Purpose and by Necessity : Compliance Under the GDPR ». In : *Financial Cryptography and Data Security*. Sous la dir. de Sarah MEIKLEJOHN et Kazue SAKO. Berlin, Heidelberg : Springer Berlin Heidelberg, p. 20-37. ISBN : 978-3-662-58387-6 (cf. p. 137).

BERTOLISSI, Clara, Jerry HARTOG et Nicola ZANNONE (2019). « Using Provenance for Secure Data Fusion in Cooperative Systems ». In : *Symposium on Access Control Models and Technologies*. ACM, p. 185-194 (cf. p. 33, 35, 36, 40, 44, 60, 69, 70, 81, 99).

BERTOLISSI, Clara, Alba MARTINEZ ANTON, Romain TESTUD et al. (2022). « Towards secure content sharing in social networks ». In : *International Conference on Cyber-Technologies and Cyber-Systems* (cf. p. 88).

BERTOLISSI, Clara, Alba MARTINEZ ANTON et Nicola ZANNONE (2023). « Data Sharing in Social Networks ». In : *Proceedings of the 28th ACM Symposium on Access Control Models and Technologies*. SACMAT '23. Trento, Italy : ACM, p. 181-192. ISBN : 9798400701733. DOI : 10.1145/3589608.3593833. URL : https://doi.org/10.1145/3589608.3593833 (cf. p. 25, 41, 45, 57, 91, 92).

BEYE, Michael, Arjan JECKMANS, Zekeriya ERKIN et al. (oct. 2010). *Literature Overview - Privacy in Online Social Networks*. English. CTIT Technical Report Series TR-CTIT-10-36. Netherlands : Centre for Telematics et Information Technology (CTIT) (cf. p. 26).

BOYD, Danah M. et Nicole B. ELLISON (2013). « Sociality through Social Network Sites ». In : *The Oxford Handbook of Internet Studies*. Sous la dir. de William H. DUTTON. Oxford : Oxford University Press, p. 151-172. ISBN : 978-0-19-958907-4 (cf. p. 26).

BRAZILIAN NATIONAL CONGRESS (2018). *Lei Geral de Proteção de Dados Pessoais (LGPD), Law No. 13,709*. https://www.planalto.gov.br/ccivil_03/_ato2015-2018/2018/lei/l13709.htm. Law No. 13,709, enacted on August 14, 2018 (cf. p. 135).

BRUNS, Glenn, Philip W.L. FONG, Ida SIAHAAN et al. (2012). « Relationship-Based Access Control : Its Expression and Enforcement through Hybrid Logic ». In : *Proceedings of the Second ACM Conference on Data and Application Security and Privacy*. CODASPY '12. New York, NY, USA : ACM, p. 117-124. ISBN : 9781450310918 (cf. p. 59).

BUTIN, Denis, Denise DEMIREL et Johannes BUCHMANN (2016). « Formal Policy-Based Provenance Audit ». In : *Advances in Information and Computer Security*. Sous la dir. de Kazuto OGAWA et Katsunari YOSHIOKA. Cham : Springer International Publishing, p. 234-253. ISBN : 978-3-319-44524-3 (cf. p. 44).

California Consumer Privacy Act (CCPA) (2018). Text of the law available at : https://leginfo.legislature.ca.gov/faces/billTextClient.xhtml?bill_id=201720180AB375. URL : https://leginfo.legislature.ca.gov/faces/billTextClient.xhtml?bill_id=201720180AB375 (visité le 11/07/2024) (cf. p. 20).

CALIFORNIA STATE LEGISLATURE (2018). *California Consumer Privacy Act of 2018*. https://oag.ca.gov/privacy/ccpa. California Civil Code, Title 1.81.5, Section 1798.100 (cf. p. 135).

CARMINATI, Barbara et Elena FERRARI (2011). « Collaborative access control in on-line social networks ». In : *7th International Conference on Collaborative Computing : Networking, Applications and Worksharing (CollaborateCom)*, p. 231-240. DOI : 10.4108/icst.collaboratecom.2011.247109 (cf. p. 103).

CARMINATI, Barbara, Elena FERRARI et Andrea PEREGO (2009). « Enforcing Access Control in Web-Based Social Networks ». In : *ACM Trans. Inf. Syst. Secur.* 13.1 (cf. p. 59).

CHAPMAN, Adriane, Barbara T. BLAUSTEIN, Len SELIGMAN et al. (août 2011). « PLUS : A provenance manager for integrated information ». In : *2011 IEEE International Conference on Information Reuse Integration*, p. 269-275. DOI : 10.1109/IRI.2011.6009558 (cf. p. 54).

CHHETRI, Tek Raj, Anelia KURTEVA, Rance J. DELONG et al. (2022). « Data Protection by Design Tool for Automated GDPR Compliance Verification Based on Semantically Modeled Informed Consent ». In : *Sensors* 22.7. ISSN : 1424-8220. DOI : 10.3390/s22072763. URL : https://www.mdpi.com/1424-8220/22/7/2763 (cf. p. 137).

CLARK, Stanley, Nikolay YAKOVETS, George FLETCHER et al. (2022). « ReLOG : A Unified Framework for Relationship-Based Access Control over Graph Databases ». In : *Data and Applications Security and Privacy XXXVI*. T. 13383. LNCS. Springer, p. 303-315 (cf. p. 59).

Cloud for Europe (s. d.). https ://cordis.europa.eu/project/id/610650. Accessed : 2024-04-05 (cf. p. 137).

COVINGTON, Michael J., Wende LONG, Srividhya SRINIVASAN et al. (2001). « Securing context-aware applications using environment roles ». In : *Proceedings of the Sixth ACM Symposium on Access Control Models and Technologies*. SACMAT '01. Chantilly,

Virginia, USA : Association for Computing Machinery, p. 10-20. ISBN : 1581133502. DOI : 10.1145/373256.373258. URL : https://doi.org/10.1145/373256.373258 (cf. p. 58).

CRAMPTON, Jason et James SELLWOOD (2014). « Path Conditions and Principal Matching : A New Approach to Access Control ». In : *Symposium on Access Control Models and Technologies*. SACMAT '14. New York, NY, USA : ACM, p. 187-198. ISBN : 9781450329392 (cf. p. 60).

CSA (mars 1996). *Model Code for the Protection of Personal Information*. Rapp. tech. CAN/CSA-Q830-96. Toronto : Canadian Standards Association (cf. p. 134).

DAKIC, Dusanka, Darko STEFANOVIC, Teodora LOLIC et al. (2020). « Event Log Extraction for the Purpose of Process Mining : A Systematic Literature Review ». In : *Innovation in Sustainable Management and Entrepreneurship*. Sous la dir. de Gabriela PROSTEAN, Juan José LAVIOS VILLAHOZ, Laura BRANCU et al. Cham : Springer International Publishing, p. 299-312. ISBN : 978-3-030-44711-3 (cf. p. 54).

DAMEN, Stan, Jerry den HARTOG et Nicola ZANNONE (2014). « CollAC : Collaborative access control ». In : *International Conference on Collaboration Technologies and Systems*. IEEE, p. 142-149 (cf. p. 61, 68).

DAMEN, Stan et Nicola ZANNONE (2014). « Privacy Implications of Privacy Settings and Tagging in Facebook ». In : *Secure Data Management*. Springer, p. 121-138 (cf. p. 23, 29, 65).

DE CAPITANI DI VIMERCATI, Sabrina, Sara FORESTI, Pierangela SAMARATI et al. (2007). « Access control policies and languages ». In : *International Journal of Computational Science and Engineering* 3.2, p. 94-102 (cf. p. 102).

DI SALVO-CILIA, Pauline, Alba MARTINEZ ANTON et Clara BERTOLISSI (2024). « Towards Automated Checking of GDPR Compliance ». In : *Proceedings of the CYBER 2024 Conference*. Accepted for presentation. IARIA (cf. p. 154).

ESTEVES, Beatriz et Víctor RODRÍGUEZ-DONCEL (mai 2024). « Analysis of ontologies and policy languages to represent information flows in GDPR ». In : *Semantic Web* 15.3. Guest editors : Michel Dumontier, Sabrina Kirrane, and Oshani Seneviratne, p. 709-743. DOI : 10.3233/SW-223009 (cf. p. 136).

FONG, Philip W. L., Mohd ANWAR et Zhen ZHAO (2009). « A Privacy Preservation Model for Facebook-Style Social Network Systems ». In : *Computer Security*. Berlin, Heidelberg : Springer, p. 303-320. ISBN : 978-3-642-04444-1 (cf. p. 59).

FONG, Philip W.L. (2011). « Relationship-Based Access Control : Protection Model and Policy Language ». In : *Conference on Data and Application Security and Privacy*. CODASPY '11. New York, NY, USA : ACM, p. 191-202. ISBN : 9781450304665 (cf. p. 59, 61).

FONG, Philip W.L., Pooya MEHREGAN et Ram KRISHNAN (2013). « Relational Abstraction in Community-Based Secure Collaboration ». In : *SIGSAC Conference on Computer & Communications Security*. New York, NY, USA : ACM, p. 585-598. ISBN : 9781450324779 (cf. p. 59, 61).

GAO, Hongyu, Jun HU, Tuo HUANG et al. (2011). « Security Issues in Online Social Networks ». In : *IEEE Internet Computing* 15.4, p. 56-63 (cf. p. 23, 69).

GATES, Carrie (2007). « Access Control Requirements for Web 2.0 Security and Privacy ». In : *IEEE Web 2.0 Privacy and Security Workshop* (cf. p. 59).

General Data Protection Regulation (s. d.). https ://gdpr-info.eu/. Accessed : 2024-03-28 (cf. p. 130).

GOLBECK, Jennifer Ann et James HENDLER (2005). « Computing and Applying Trust in Web-Based Social Networks ». AAI3178583. Thèse de doct. USA. ISBN : 0542183242 (cf. p. 50, 111).

GOUVERNEMENT DU CANADA (2000). *Loi sur la protection des renseignements personnels et les documents électroniques*. Loi S.C. 2000, c. 5. URL : https://laws-lois.justice.gc.ca/fra/lois/p-8.6/index.html (cf. p. 134).

HAMDANI, Rajaa El, Majd MUSTAPHA, David Restrepo AMARILES et al. (2021). « A Combined Rule-Based and Machine Learning Approach for Automated GDPR Compliance Checking ». In : *Proceedings of the Eighteenth International Conference on Artificial Intelligence and Law*. ICAIL '21. São Paulo, Brazil : Association for Computing Machinery, p. 40-49. ISBN : 9781450385268. DOI : 10.1145/3462757.3466081. URL : https://doi.org/10.1145/3462757.3466081 (cf. p. 138).

HASAN, Ragib, Radu SION et Marianne WINSLETT (2007). « Introducing secure provenance : problems and challenges ». In : *Proceedings of the 2007 ACM Workshop on Storage Security and Survivability*. StorageSS '07. Alexandria, Virginia, USA : Association for Computing Machinery, p. 13-18. ISBN : 9781595938916. DOI : 10.1145/1314313.1314318. URL : https://doi.org/10.1145/1314313.1314318 (cf. p. 45).

HO, Ai Thanh (2012). « Towards a privacy-enhanced social networking site ». AAINR78960. Thèse de doct. CAN. ISBN : 9780494789605 (cf. p. 20).

HSU, Andy Chunliang et Indrakshi RAY (2016). « Specification and Enforcement of Location-Aware Attribute-Based Access Control for Online Social Networks ». In : *Proceedings of the 2016 ACM International Workshop on Attribute Based Access Control*. ABAC '16. New Orleans, Louisiana, USA : Association for Computing Machinery, p. 25-34. ISBN : 9781450340793. DOI : 10.1145/2875491.2875495. URL : https://doi.org/10.1145/2875491.2875495 (cf. p. 60).

HU, Hongxin, Gail-Joon AHN et Jan JORGENSEN (2011). « Detecting and Resolving Privacy Conflicts for Collaborative Data Sharing in Online Social Networks ». In : *Proceedings of the 27th Annual Computer Security Applications Conference*. ACSAC '11. Orlando, Florida, USA : ACM, p. 103-112. ISBN : 9781450306720. DOI : 10.1145/2076732.2076747. URL : https://doi.org/10.1145/2076732.2076747 (cf. p. 104-106, 113, 121, 127).

– (2013). « Multiparty Access Control for Online Social Networks : Model and Mechanisms ». In : *IEEE Transactions on Knowledge and Data Engineering* 25.7, p. 1614-1627. DOI : 10.1109/TKDE.2012.97 (cf. p. 61, 103, 104).

HU, Hongxin, Gail-Joon AHN, Ziming ZHAO et al. (2014). « Game theoretic analysis of multiparty access control in online social networks ». In : *ACM Symposium on Access Control Models and Technologies*. URL : https://api.semanticscholar.org/CorpusID:6380293 (cf. p. 103, 104).

HU, Rui, Zheng YAN, Wenxiu DING et al. (mars 2020). « A survey on data provenance in IoT ». en. In : *World Wide Web* 23.2, p. 1441-1463 (cf. p. 44).

ILIA, Panagiotis, Barbara CARMINATI, Elena FERRARI et al. (2017). « SAMPAC : Socially-Aware Collaborative Multi-Party Access Control ». In : *Conference on Data and Application Security and Privacy*. CODASPY '17. New York, NY, USA : ACM, p. 71-82. ISBN : 9781450345231 (cf. p. 61).

ILIA, Panagiotis, Iasonas POLAKIS, Elias ATHANASOPOULOS et al. (2015). « Face/Off : Preventing Privacy Leakage From Photos in Social Networks ». In : *Proceedings of the 22nd ACM SIGSAC Conference on Computer and Communications Security*. CCS '15. Denver, Colorado, USA : ACM, p. 781-792. ISBN : 9781450338325. DOI : 10.1145/2810103.2813603. URL : https://doi.org/10.1145/2810103.2813603 (cf. p. 103).

ISAAK, Jim et Mina J HANNA (2018). « User data privacy : Facebook, Cambridge Analytica, and privacy protection ». In : *Computer* 51.8, p. 56-59 (cf. p. 20).

JAIN, Ankit Kumar, Somya Ranjan SAHOO et Jyoti KAUBIYAL (2021). « Online social networks security and privacy : comprehensive review and analysis ». In : *Complex & Intelligent Systems* 7.5, p. 2157-2177 (cf. p. 23, 69).

JIANG, Wenjun et Guojun WANG (2011). « SWTrust : Generating Trusted Graph for Trust Evaluation in Online Social Networks ». In : *IEEE 10th International Conference on Trust, Security and Privacy in Computing and Communications*, p. 320-327. DOI : 10.1109/TrustCom.2011.251 (cf. p. 50).

JIANG, Wenjun, Guojun WANG, Md Zakirul Alam BHUIYAN et al. (31 mars 2017). « Understanding Graph-Based Trust Evaluation in Online Social Networks : Methodologies and Challenges ». In : *ACM Computing Surveys* 49.1, p. 1-35. ISSN : 0360-0300, 1557-7341. DOI : 10.1145/2906151. URL : https://dl.acm.org/doi/10.1145/2906151 (visité le 06/10/2023) (cf. p. 50).

KALUVURI, Samuel Paul, Alexandru Ionut EGNER, Jerry den HARTOG et al. (2015). « SAFAX - An Extensible Authorization Service for Cloud Environments ». In : *Frontiers ICT* 2, p. 9. DOI : 10.3389/fict.2015.00009. URL : https://doi.org/10.3389/fict.2015.00009 (cf. p. 99).

KING, Samuel T. et Peter M. CHEN (2005). « Backtracking intrusions ». In : *ACM Trans. Comput. Syst.* 23.1, p. 51-76. DOI : 10.1145/1047915.1047918. URL : https://doi.org/10.1145/1047915.1047918 (cf. p. 54).

KOKOLAKIS, Spyros (2017). « Privacy attitudes and privacy behaviour : A review of current research on the privacy paradox phenomenon ». In : *Computers Security* 64, p. 122-134. ISSN : 0167-4048. DOI : https://doi.org/10.1016/j.cose.2015.07.002. URL : https://www.sciencedirect.com/science/article/pii/S0167404815001017 (cf. p. 15).

KRASNOVA, Hanna, Sarah SPIEKERMANN, Ksenia KOROLEVA et al. (juin 2010). *Online Social Networks : Why We Disclose*. en. DOI : 10.1057/jit.2010.6. URL : http://dx.doi.org/10.1057/jit.2010.6 (cf. p. 15, 101, 106, 109).

LAMPSON, Butler W. (jan. 1974). « Protection ». In : *SIGOPS Oper. Syst. Rev.* 8.1, p. 18-24. ISSN : 0163-5980. DOI : 10.1145/775265.775268. URL : https://doi.org/10.1145/775265.775268 (cf. p. 56).

LAROUSSE (n.d.). *Provenance*. Consulté le 24 juillet 2024. URL : https://www.larousse.fr/dictionnaires/francais/provenance/64634 (cf. p. 33).

Larousse - Vie privée (2024). Larousse en ligne. URL : https://www.larousse.fr/dictionnaires/francais/vie_priv%C3%A9e/81570 (visité le 11/07/2024) (cf. p. 19).

LEE, Brian, Abir AWAD et Mirna AWAD (2015). « Towards Secure Provenance in the Cloud : A Survey ». In : *2015 IEEE/ACM 8th International Conference on Utility and Cloud Computing (UCC)*, p. 577-582. DOI : 10.1109/UCC.2015.102 (cf. p. 45).

LEVENS, Watson, Alex SZORKOVSZKY et David J. T. SUMPTER (2022). « Friend of a friend models of network growth ». en. In : *Royal Society open science* 9.10. ISSN : 2054-5703. DOI : 10.1098/rsos.221200. URL : http://dx.doi.org/10.1098/rsos.221200 (cf. p. 90, 124).

LI, Ninghui, J. MITCHELL et W. WINSBOROUGH (2002). « Design of a role-based trust-management framework ». In : *Symposium on Security and Privacy*. IEEE, p. 114-130 (cf. p. 60).

LI, Ninghui, Qihua WANG, Wahbeh QARDAJI et al. (juin 2009). « Access control policy combining : Theory meets practice ». In : p. 135-144. DOI : 10.1145/1542207.1542229 (cf. p. 102).

LIM, Chunhyeok, Shiyong LU, Artem CHEBOTKO et al. (2011). « Storing, reasoning, and querying OPM-compliant scientific workflow provenance using relational databases ». In : *Future Generation Computer Systems* 27.6, p. 781-789. ISSN : 0167-739X. DOI : https://doi.org/10.1016/j.future.2010.10.013. URL : https://www.sciencedirect.com/science/article/pii/S0167739X10002062 (cf. p. 44).

LIU, Ling, Yuzhong SHI et Yuting ZHAO (2013). « MeTrust : A Trust Model for Social Networks ». In : *Proceedings of the 2013 International Conference on Data Mining Workshops*. IEEE, p. 45-50. URL : https://ieeexplore.ieee.org/document/6729663 (cf. p. 51).

MADANI, Mohamed Amine, Abdelmounaim KERKRI et Mohammed AISSAOUI (2023). « MC-ABAC : An ABAC-based Model for Collaboration in Multi-Cloud Environment ». In : *International Journal of Advanced Computer Science and Applications* 14.6. DOI : 10.14569/IJACSA.2023.01406126. URL : http://dx.doi.org/10.14569/IJACSA.2023.01406126 (cf. p. 60).

MAHMUDLU, Rauf, Jerry HARTOG et Nicola ZANNONE (2016). « Data Governance and Transparency for Collaborative Systems ». In : *Data and Applications Security and Privacy XXX*. T. 9766. LNCS. Cham : Springer, p. 199-216 (cf. p. 61, 69).

MARTINEZ ANTON, Alba, Clara BERTOLISSI et Jean-Marc TALBOT (2024). « Solving Access Control Conflicts in Multi-User Systems ». In : *Proceedings of the 21st International Conference on Security and Cryptography - Volume 1 : SECRYPT*. INSTICC. SciTePress, p. 492-499. ISBN : 978-989-758-709-2. DOI : 10.5220/0012768500003767 (cf. p. 25, 45, 101).

MCCALLISTER, Erika, Tim GRANCE et Karen SCARFONE (2010). *Guide to Protecting the Confidentiality of Personally Identifiable Information (PII)*. Rapp. tech. NIST SP 800-

122. National Institute of Standards et Technology. URL : https://nvlpubs.nist.gov/nistpubs/Legacy/SP/nistspecialpublication800-122.pdf (cf. p. 21).

MEHREGAN, Pooya et Philip W.L. FONG (2016). « Policy Negotiation for Co-Owned Resources in Relationship-Based Access Control ». In : *Symposium on Access Control Models and Technologies*. SACMAT '16. New York, NY, USA : ACM, p. 125-136. ISBN : 9781450338028 (cf. p. 61).

MOREAU, Luc, Ben CLIFFORD, Juliana FREIRE et al. (2011). « The Open Provenance Model core specification (v1.1) ». In : *Future Generation Computer Systems* 27.6, p. 743-756 (cf. p. 33, 35, 36, 41).

MOSCA, Francesca et Jose SUCH (jan. 2022). « An explainable assistant for multiuser privacy ». In : *Autonomous Agents and Multi-Agent Systems* 36.1. ISSN : 1573-7454. DOI : 10.1007/s10458-021-09543-5. URL : http://dx.doi.org/10.1007/s10458-021-09543-5 (cf. p. 104, 106, 113, 121).

MUELLER, Milton (sept. 2006). « Privacy and Security in Highly Dynamic Systems ». In : *Communications of the ACM* 49. DOI : 10.1145/1151030.1151051 (cf. p. 20).

MUNISWAMY-REDDY, Kiran-Kumar, David A. HOLLAND, Uri BRAUN et al. (2006). « Provenance-aware storage systems ». In : *Proceedings of the Annual Conference on USENIX '06 Annual Technical Conference*. ATEC '06. Boston, MA : USENIX Association, p. 4 (cf. p. 54).

NARAYANAN, Arvind et Vitaly SHMATIKOV (2008). « Robust De-anonymization of Large Sparse Datasets ». In : *2008 IEEE Symposium on Security and Privacy (sp 2008)*, p. 111-125. DOI : 10.1109/SP.2008.33 (cf. p. 21).

NATIONAL PEOPLE'S CONGRESS (NPC) (s. d.). *Personal Information Protection Law (PIPL)*. https://personalinformationprotectionlaw.com/. Accessed : 2024-09-04 (cf. p. 135).

NGUYEN, Dang, Jaehong PARK et Ravi SANDHU (2013). « A provenance-based access control model for dynamic separation of duties ». In : *2013 Eleventh Annual Conference on Privacy, Security and Trust*, p. 247-256. DOI : 10.1109/PST.2013.6596060 (cf. p. 45, 60, 99).

OASIS (2013). *eXtensible Access Control Markup Language (XACML) Version 3.0*. OASIS Standard (cf. p. 60, 70, 73, 101, 102).

OECD (fév. 2002). *OECD Guidelines on the Protection of Privacy and Transborder Flows of Personal Data*. OECD. ISBN : 9789264196391. DOI : 10.1787/9789264196391-en. URL : http://dx.doi.org/10.1787/9789264196391-en (cf. p. 130).

– (juillet 2013). *The OECD Privacy Framework*. Rapp. tech. C(2013)79. Pour garantir la confiance dans l'économie numérique, il est impératif que les politiques de protection de la vie privée favorisent non seulement la transparence et la responsabilité des gestionnaires de données, mais aussi la capacité des individus à contrôler et à protéger leurs informations personnelles. Paris : Organisation for Economic Co-operation et Development ((cf. p. 134).

Oxford - Privacy (2024). Oxford Languages. URL : https://www.lexico.com/definition/privacy (visité le 11/07/2024) (cf. p. 19).

OXFORD UNIVERSITY PRESS (n.d.). *Provenance*. Consulté le 24 juillet 2024. URL : https://www.oed.com/search/dictionary/?scope=Entries&q=provenance (cf. p. 33).

PACI, Federica, Anna SQUICCIARINI et Nicola ZANNONE (jan. 2018). « Survey on Access Control for Community-Centered Collaborative Systems ». In : *ACM Comput. Surv.* 51.1. ISSN : 0360-0300. DOI : 10.1145/3146025. URL : https://doi.org/10.1145/3146025 (cf. p. 24, 57-59, 61, 62, 68, 99, 101, 108, 126).

PAIVA MELO MARIN, Regina Paiva Melo (sept. 2015). « Enhancing privacy protection in social network systems through decentralization and policy conflict management ». Theses. CentraleSupélec. URL : https://theses.hal.science/tel-01242091 (cf. p. 104).

PALMIRANI, Monica et Guido GOVERNATORI (2018). « Modelling Legal Knowledge for GDPR Compliance Checking ». In : *International Conference on Legal Knowledge and Information Systems* (cf. p. 137).

PARK, Jaehong, Dang NGUYEN et Ravi SANDHU (2012). « A provenance-based access control model ». In : *2012 Tenth Annual International Conference on Privacy, Security and Trust*, p. 137-144. DOI : 10.1109/PST.2012.6297930 (cf. p. 44, 60).

PAWAR, Ambika, Swati AHIRRAO et Prathamesh P. CHURI (2018). « Anonymization Techniques for Protecting Privacy : A Survey ». In : *2018 IEEE Punecon*, p. 1-6. DOI : 10.1109/PUNECON.2018.8745425 (cf. p. 23).

RAJTMAJER, Sarah, Anna SQUICCIARINI, Christopher GRIFFIN et al. (2016a). « Constrained Social-Energy Minimization for Multi-Party Sharing in Online Social Networks ». In : *International Conference on Autonomous Agents & Multiagent Systems*. AAMAS '16. Richland, SC : International Foundation for Autonomous Agents et Multiagent Systems, p. 680-688. ISBN : 9781450342391 (cf. p. 61).

– (2016b). « Constrained Social-Energy Minimization for Multi-Party Sharing in Online Social Networks ». In : *Proceedings of the 2016 International Conference on Autonomous Agents & Multiagent Systems*. AAMAS '16. Singapore, Singapore : International Foundation for Autonomous Agents et Multiagent Systems, p. 680-688. ISBN : 9781450342391 (cf. p. 103).

RHAHLA, Mouna, Sahar ALLEGUE et Takoua ABDELLATIF (2021). « Guidelines for GDPR compliance in Big Data systems ». In : *Journal of Information Security and Applications* 61, p. 102896. ISSN : 2214-2126. DOI : https://doi.org/10.1016/j.jisa.2021.102896. URL : https://www.sciencedirect.com/science/article/pii/S221421262100123X (cf. p. 132, 136).

SALTARELLA, Marco, Giuseppe DESOLDA et Rosa LANZILOTTI (2021). « Privacy Design Strategies and the GDPR : A Systematic Literature Review ». In : *HCI for Cybersecurity, Privacy and Trust : Third International Conference, HCI-CPT 2021, Held as Part of the 23rd HCI International Conference, HCII 2021, Virtual Event, July 24–29, 2021, Proceedings*. Springer, p. 241-257 (cf. p. 136).

SANDHU, Ravi (1996). « Roles versus groups ». In : *Proceedings of the First ACM Workshop on Role-Based Access Control*. RBAC '95. Gaithersburg, Maryland, USA : Association for Computing Machinery, 7-es. ISBN : 0897917596. DOI : 10.1145/270152.270163. URL : https://doi.org/10.1145/270152.270163 (cf. p. 58).

SCHNEBLE, Christophe Olivier, Bernice Simone ELGER et David Martin SHAW (fév. 2020). « Google's Project Nightingale highlights the necessity of data science ethics review ». In : *EMBO Molecular Medicine* 12.3. ISSN : 1757-4684. DOI : 10.15252/emmm.202012053. URL : http://dx.doi.org/10.15252/emmm.202012053 (cf. p. 22).

SCHWARTZ, Paul et Daniel SOLOVE (déc. 2011). « The PII Problem : Privacy and a New Concept of Personally Identifiable Information ». In : *New York University Law Review* 86 (cf. p. 21).

SCHWARTZ, Shalom H (2012). « An overview of the Schwartz theory of basic values ». In : *Online readings in Psychology and Culture* 2.1, p. 11 (cf. p. 24, 105).

Security and Privacy Controls for Information Systems and Organizations (2020). Special Publication 800-53 Rev. 5. Accessed : 2024-07-12. National Institute of Standards et Technology. URL : https://doi.org/10.6028/NIST.SP.800-53r5 (cf. p. 20).

SHEN, HongHai et Prasun DEWAN (1992). « Access Control for Collaborative Environments ». In : *Proceedings of the 1992 ACM Conference on Computer-Supported Cooperative Work*. CSCW '92. Toronto, Ontario, Canada : ACM, p. 51-58. ISBN : 0897915429. DOI : 10.1145/143457.143461. URL : https://doi.org/10.1145/143457.143461 (cf. p. 102).

SIKOS, Leslie F et Dean PHILP (sept. 2020). « Provenance-aware knowledge representation : A survey of data models and contextualized knowledge graphs ». en. In : *Data Sci. Eng.* 5.3, p. 293-316 (cf. p. 54).

SMARI, Waleed W., Patrice CLEMENTE et Jean-Francois LALANDE (2014). « An extended attribute based access control model with trust and privacy : Application to a collaborative crisis management system ». In : *Future Generation Computer Systems* 31. Special Section : Advances in Computer Supported Collaboration : Systems and Technologies, p. 147-168. ISSN : 0167-739X. DOI : https://doi.org/10.1016/j.future.2013.05.010. URL : https://www.sciencedirect.com/science/article/pii/S0167739X1300109X (cf. p. 60).

SQUICCIARINI, Anna, Federica PACI et Smitha SUNDARESWARAN (2014). « Prima : A Comprehensive Approach to Privacy Protection in Social Network Sites ». In : *Special Issue of Annals of Telecommunications on Privacy-aware electronic society* 69 (cf. p. 59).

SUCH, Jose M. et Natalia CRIADO (2015). « Resolving Multi-Party Privacy Conflicts in Social Media ». In : *IEEE Transactions on Knowledge and Data Engineering* 28, p. 1851-1863. URL : https://api.semanticscholar.org/CorpusID:169880620 (cf. p. 49, 104-106, 111, 118, 119, 122, 123, 125, 127).

SWISSA, Adi et Ehud GUDES (2021). « An ABAC Model with Trust and Gossiping (ABAC–TG) for Online Social Networks ». In : *Data and Applications Security and Privacy XXXV*. Sous la dir. de Ken BARKER et Kambiz GHAZINOUR. Cham : Springer International Publishing, p. 377-392. ISBN : 978-3-030-81242-3 (cf. p. 60).

TAUQEER, Amar, Anelia KURTEVA, Tek Raj CHHETRI et al. (2022). « Automated GDPR Contract Compliance Verification Using Knowledge Graphs ». In : *Information* 13.10. ISSN : 2078-2489. DOI : 10.3390/info13100447. URL : https://www.mdpi.com/2078-2489/13/10/447 (cf. p. 136, 137).

THE EUROPEAN PARLIAMENT AND THE COUNCIL (oct. 1995). « Directive 1995/46/EC of the European Parliament and of the Council of 24 October 1995 on the protection of individuals with regard to the processing of personal data and on the free movement of such data ». In : *Official Journal of the European Communities* (cf. p. 130).

THOMAS, Roshan K. (1997). « Team-based access control (TMAC) : a primitive for applying role-based access controls in collaborative environments ». In : *Proceedings of the Second ACM Workshop on Role-Based Access Control*. RBAC '97. Fairfax, Virginia, USA : Association for Computing Machinery, p. 13-19. ISBN : 0897919858. DOI : 10.1145/266741.266748. URL : https://doi.org/10.1145/266741.266748 (cf. p. 58).

THOMAS, Roshan K. et Ravi S. SANDHU (1998). « Task-based authorization controls (TBAC) : a family of models for active and enterprise-oriented authorization management ». In : *Database Security XI : Status and Prospects*. Sous la dir. de T. Y. LIN et Shelly QIAN. Boston, MA : Springer US, p. 166-181. ISBN : 978-0-387-35285-5. DOI : 10.1007/978-0-387-35285-5_10. URL : https://doi.org/10.1007/978-0-387-35285-5_10 (cf. p. 58).

TikTok : Recent Data Privacy and National Security Concerns (2023). Accessed : 2024-07-12. URL : https://crsreports.congress.gov/product/pdf/IN/IN12131 (cf. p. 21).

TOLONE, William, Gail-Joon AHN, Tanusree PAI et al. (mars 2005). « Access control in collaborative systems ». In : *ACM Comput. Surv.* 37.1, p. 29-41. ISSN : 0360-0300. DOI : 10.1145/1057977.1057979. URL : https://doi.org/10.1145/1057977.1057979 (cf. p. 58).

TRIVELLATO, Daniel, Nicola ZANNONE et Sandro ETALLE (2014). « GEM : A distributed goal evaluation algorithm for trust management ». In : *Theory and Practice of Logic Programming* 14.3, p. 293-337. DOI : 10.1017/S1471068412000397 (cf. p. 60).

U.S. CONGRESS (1996). *Health Insurance Portability and Accountability Act of 1996*. https://aspe.hhs.gov/reports/health-insurance-portability-accountability-act-1996. Public Law 104-191, consulté 4 septembre 2024 (cf. p. 134).

UK General Data Protection Regulation (UK GDPR) (2024). Accessed : 2024-08-15. URL : https://uk-gdpr.org/ (cf. p. 134).

UNION, European (2016). *Regulation (EU) 2016/679 of the European Parliament and of the Council of 27 April 2016*. URL : https://eur-lex.europa.eu/eli/reg/2016/679/oj (cf. p. 20).

VISHWAMITRA, Nishant, Yifang LI, Kevin WANG et al. (2017). « Towards PII-Based Multiparty Access Control for Photo Sharing in Online Social Networks ». In : *Proceedings of the 22nd ACM on Symposium on Access Control Models and Technologies*. SACMAT '17 Abstracts. Indianapolis, Indiana, USA : ACM, p. 155-166. ISBN : 9781450347020. DOI : 10.1145/3078861.3078875. URL : https://doi.org/10.1145/3078861.3078875 (cf. p. 61, 103).

WANG, Yilun et Michal KOSINSKI (fév. 2018). « Deep neural networks are more accurate than humans at detecting sexual orientation from facial images. » In : *Journal of Personality and Social Psychology* 114.2, p. 246-257. ISSN : 0022-3514. DOI : 10.1037/pspa0000098. URL : http://dx.doi.org/10.1037/pspa0000098 (cf. p. 21, 24).

WARREN, Samuel D et Louis D BRANDEIS (1890). « The right to privacy ». In : *Harvard Law Review* 4.5, p. 193-220 (cf. p. 20).

WESTIN, Alan F. (1967). *Privacy and Freedom*. New York : Atheneum (cf. p. 20).

WILTON, Robin (2017). « After Snowden – the evolving landscape of privacy and technology ». In : *Journal of Information, Communication and Ethics in Society* 15.3, p. 328-335. DOI : 10.1108/JICES-02-2017-0010. URL : https://doi-org.lama.univ-amu.fr/10.1108/JICES-02-2017-0010 (cf. p. 20).

XIONG, Jing, Guohui XIAO, Tahir E. KALAYCI et al. (2023). « A virtual knowledge graph based approach for object-centric event logs extraction ». English. In : p. 466. ISBN : 3031278151 ;9783031278150 ;9783031278143 ;3031278143 ; (cf. p. 54).

YUAN, Eric et Jin TONG (2005). « Attributed based access control (ABAC) for Web services ». In : *IEEE International Conference on Web Services (ICWS'05)*, p. 569. DOI : 10.1109/ICWS.2005.25 (cf. p. 60).

ZHONG, Haoti, Anna SQUICCIARINI et David MILLER (2018). « Toward Automated Multiparty Privacy Conflict Detection ». In : *Proceedings of the 27th ACM International Conference on Information and Knowledge Management*. CIKM '18. Torino, Italy : ACM, p. 1811-1814. ISBN : 9781450360142. DOI : 10.1145/3269206.3269329. URL : https://doi.org/10.1145/3269206.3269329 (cf. p. 104).

ANNEXES

A. Fichier d'une politique

Ci-dessous un exemple de fichier politique manipulés par les prototypes présentés dans les Chapitres 2 et 3. Le prototype du Chapitre 3 utilise uniquement des politiques sans horodatages sur les chemins.

```
<photo>
        not ( notAvailable(objet,*))
        and (
                        ( owns(tyler,objet,*) and action == 'view' )
                or
                        ( owns(tyler,objet,*)
                        and areRelated(tyler,utilisateur,friends,*,*)
                        and not ( areRelated(colin,utilisateur,friends,*,∞))
                        and action == 'comment' )
        )
</photo>
<comment>
        not ( {notAvailable(objet,*) )
        and (
                ( owns(tyler,objet,*)
                and action == 'share' )
        )
</comment>
<share>
        not ( notAvailable(objet,*) )
        and (
                        ( owns(tyler,objet,*)
                        and areRelated(tyler,utilisateur,friends,tb0,∞)
                        and wasGeneratedBy(objet,P,R,t0) and (tb0 == t0)
                        and action == 'comment' )
                or
                        ( owns(tyler,objet,*)
                        and areRelated(tyler,utilisateur,friends,tb1,*)
                        and wasGeneratedBy(objet,P,R,t1) and (tb1 >= t1)
                        and not ( areRelated(israel,utilisateur,friends,*,*))
                        and not ( areRelated(alisa,utilisateur,friends,*,*))
                        and not ( areRelated(sally,utilisateur,friends,*,∞) )
                        and action == 'view' )
        )
</share>
```

B. Prolog : syntaxe et patterns

B.1. Syntaxe Prolog

Prolog est un langage de programmation basé sur la logique, utilisant des faits, des règles et des requêtes. Tous ces éléments sont écrits à l'aide de termes. Un terme est composé d'un foncteur suivi d'arguments. Par exemple $animal(chat)$, le foncteur est $animal$ et il n'a qu'un seul argument, $chat$. Il existe également des variables en Prolog, une variable est une chaîne de caractères commençant par une majuscule (par exemple X,Chat,...).

Un fait est une relation explicite (prédicat) entre des éléments qui est vraie, par exemple $ville(Marseille)$. Dans notre cas, les faits sont les dépendances causales du graphe de provenance.

Une règle est un prédicat contenant une implication logique (codée comme $:-$), le côté gauche de la règle n'est vrai que si les faits du côté droit sont vrais. Le côté droit peut contenir des conjonctions, des disjonctions et des négations de faits. La conjonction est représentée par une virgule (,), par exemple $p(a) \wedge q(b)$ s'écrit $P(a), q(b)$. Pour écrire un prédicat avec une disjonction d'expressions, on peut utiliser le symbole point-virgule (;) ou écrire autant de prédicats qu'il y a de clauses dans la disjonction. Par exemple $q = p(A) \vee p(b)$ s'écrit $q :- P(A)$.
$q :- p(b)$. Chaque prédicat se termine par un point (.). Pour la négation, deux notations peuvent être utilisées selon l'environnement Prolog : \+ ou NOT. La négation du prédicat $predicate$, est alors $\backslash + \ predicate$ par exemple.

Une requête en Prolog commence toujours par le symbole ?−.Prolog peut être interrogé avec des prédicats sans variables sur les faits et les règles du fichier (par exemple $? - wasControlledBy(process, Alice, "owner", 14, 15)$) ou avec des variables (par exemple $? - wasControlledBy(P, Alice, "owner", T1, T2)$, où $P, T1$ et $T2$ sont des variables).
Lorsque la requête n'a pas de variables, la réponse donnée par Prolog sera *true* ou *success* (selon l'environnement) s'il trouve pour le prédicat les faits et les règles nécessaires pour l'affirmer. Le programme retournera *false* ou *fail* dans le cas contraire.
Dans le cas avec des variables, Prolog trouvera toutes les instanciations possibles des variables, et les retournera une par une. A la fin, si le programme sait qu'il n'y a plus de possibilités, il terminera la dernière réponse par un point. Si le programme ne trouve pas d'autres possibilités, mais qu'il n'en est pas certain, il renverra un *false* ou *fail* après la dernière instanciation. S'il n'y a pas d'instanciations, il renverra directement *false* ou *fail*.

B.2. Patterns

B.2.1. Représentation des attributs

Le Prolog représentant les attributs est un prédicat de la forme $att(n, v)$, correspondant à dire l'attribut att du nœud n a pour valeur val.

Attribut finalité

$$purposes(consent, artifact1, [purpose1, purpose2, ...])$$
$$purposes(consent, artifact2, [purpose3, purpose4, ...])$$
$$...$$

Lorsque les finalités sont globales, le paramètre $artifact$ est remplacé par _, qui fait référence à n'importe quel artefact. $purposes(consent, _, [purpose1, purpose2, ...])$ Si la personne concernée a son consentement pour une certaine finalité PU pour un artefact D, le prédicat suivant renverra l'index dans la liste des finalités associées Q, Faux si la finalité n'est pas dans la liste.

$$isPurpose(PU, D, C) :- purposes(C, D, Q), nth0(X, Q, PU).$$

Le prédicat $nth0(X, Q, P)$ est un prédicat Prolog qui permet de vérifier si un élément PU est dans une liste Q d'index X. Si X est une variable, il renverra l'index de P s'il existe dans Q, False sinon. Ici, C est l'artefact représentant le consentement, et le prédicat renverra l'index de PU s'il s'agit d'une finalité valide d'utiliser D, False sinon. Dans le cas de finalités globales, la variable D sera remplacée par _. Par exemple, en se basant sur le scénario présenté en Figure 4.1, l'attribut $purposes$ de l'artefact consentement $consent_bob_v0$ est la liste $purposes("consent_bob_v0", ["thirdParties", "analysis", "improvement"])$. Ceci signifie que Bob donne le même consentement pour toutes ses données personnelles. Ses données personnelles peuvent être utilisées pour des fins d'analyse, d'amélioration et par des tiers.

Attribut action et donnée personnelle Chaque processus contient un attribut action, qui indique la finalité de son exécution. Par exemple le processus en charge de créer le consentement a pour action *donner le consentement*. De même, les artefacts possèdent un attribut, qui peut avoir comme valeur vrai ou faux, afin de savoir s'il représente une donnée personnelle ou non.

B.2.2. Représentation des arrêtes

Les arrêtes du graph de provenance sont représentées sous la forme de faits Prologs. Par exemple, le fait $\mathsf{wasControlledBy}(review1, Alice, "owner", 75, 75)$ signifie que le graphe de provenance contient la dépendance causale wasControlledBy reliant le processus *review1* à l'agent *Alice* en temps que propriétaire. La conjonction de plusieurs dépendances ou la recherche d'un chemin est représenté par une conjonction de faits ou de règles. Par exemple, si une règle correspond au chemin $\mathsf{wasControlledBy}(P1, S, "owner", TB, TE) \wedge \mathsf{wasGeneratedBy}(C, P1, "consent", T)$, il sera défini en Prolog comme $\mathsf{wasControlledBy}(P1, S, "owner", TB, TE), \mathsf{wasGeneratedBy}(C, P1, "consent", T)$.