POLYTECHNIQUE MONTRÉAL
affiliée à l'Université de Montréal

Vérification automatique de la confidentialité différentielle

LOTFI LARBAOUI
Département de génie informatique et génie logiciel

Mémoire présenté en vue de l'obtention du diplôme de *Maîtrise ès sciences appliquées*
Génie informatique

Mars 2020

POLYTECHNIQUE MONTRÉAL
affiliée à l'Université de Montréal

Ce mémoire intitulé :

Vérification automatique de la confidentialité différentielle

présenté par **Lotfi LARBAOUI**
en vue de l'obtention du diplôme de *Maîtrise ès sciences appliquées*
a été dûment accepté par le jury d'examen constitué de :

Giuliano ANTONIOL, président
John MULLINS, membre et directeur de recherche
Foutse KHOMH, membre

DÉDICACE

Je dédie ce travail
Aux êtres qui me sont les plus chers au monde,
À mes chers parents Ahmed Larbaoui et Oumnail Bensalah
À mes frères et mes soeurs
Fateh, Hanane, Leila, Bilal et Islam
À tous mes amis.

Lotfi Larbaoui

REMERCIEMENTS

C’est avec un grand plaisir que nous réservons ces quelques lignes en signe de gratitude et de profonde reconnaissance à tous ceux qui ont contribué à l’aboutissement de ce travail.

Avec une grande sincérité, nous remercions M. John Mullins, Professeur à Polytechnique Montréal, pour sa disponibilité, son accompagnement, la précision et la pertinence de ses remarques qui ont été déterminantes sur le fond et sur la forme du travail présenté.

Nous adressons aussi nos remerciements à M. Sardaouna Hamadou, associé de recherche à Polytechnique Montréal, pour ses conseils et pour sa collaboration dans ce projet.

Nous adressons finalement tous nos remerciements aux membres de jury pour nous avoir honorés en acceptant d’évaluer notre travail.

RÉSUMÉ

Ce rapport étudie la vérification quantitative de la confidentialité différentielle dans les systèmes distribués. Tout d'abord, nous examinons l'applicabilité de la vérification des modèles probabilistes pour fournir des garanties sur le comportement des système différentiellement confidentiels. Ensuite, nous concevons des méthodes qui extraient automatiquement les modèles des systèmes à partir d'une description de haut niveau, puis nous effectuons une vérification probabiliste de ces modèles. Nous développons à cette fin une nouvelle méthodologie de la vérification quantitative. Nous décrivons des méthodes formelles pour analyser un large éventail de propriétés de confidentialité, notamment la précision et la perte de la confidentialité. Nous avons également réexprimé la notion de confidentialité différentielle pour raisonner sur deux exécutions de programmes similaires. À notre connaissance, il s'agit des analyses de confidentialité les plus générales pour les systèmes distribués. Deuxièmement, nous fournissons des preuves de couplage basées sur les relations de levage approximatives pour prouver la confidentialité différentielle dans les chaînes de Markov. Nous proposons également des algorithmes de vérification symbolique de la confidentialité. L'avantage de notre approche est que ces algorithmes peuvent être facilement implémentés dans n'importe quel outil de vérification de modèles probabilistes. Enfin, nous définissons une approche pour extraie des contre-exemples qui peuvent être utilisés pour fin de débogage similaires en fournissant une exécution qui viole la confidentialité.

Mots-clés : Confidentialité différentielle, Vérification automatique, Processus de Décision Markovien, Model Checking probabiliste, contre-exemple, raisonnement probabiliste.

ABSTRACT

This report studies the quantitative verification of differential privacy in distributed systems. First, we examine the applicability of probabilistic model checking to provide guarantees on the behavior of differentially private systems. Next, we design methods that automatically extract the models of the systems from a high-level description, then we perform a probabilistic verification of these models. To this end, we are developing a new methodology for quantitative verification. We describe formal methods for analyzing a wide range of privacy properties, including accuracy and privacy loss. We have also re-expressed the notion of differential privacy to reason about two executions of similar programs. To our knowledge, this is the most general privacy analysis for distributed systems. Second, we provide evidence of coupling based on approximate lifting relationships to prove differential privacy in Markov chains. We also offer symbolic algorithm for verification of confidentiality. The advantage of our approach is that these algorithms can be easily implemented in any probabilistic model checker tool. Finally, we define an approach for extracting counterexamples that can be used for similar debugging purposes by providing an execution that violates confidentiality.

Keywords: Differential privacy, Automated verification, Markov Decision Processes, Probabilistic Model Checking, Counter-example, Probabilistic reasoning.

TABLE DES MATIÈRES

DÉDICACE iii

REMERCIEMENTS iv

RÉSUMÉ v

ABSTRACT vi

TABLE DES MATIÈRES vii

LISTE DES TABLEAUX xi

LISTE DES FIGURES xii

LISTE DES SIGLES ET ABRÉVIATIONS xiii

LISTE DES ANNEXES xiv

CHAPITRE 1 INTRODUCTION 1
- 1.1 Motivation 2
- 1.2 Définitions et concepts de base 3
  - 1.2.1 La vie privée 3
  - 1.2.2 Vérification automatique 3
  - 1.2.3 La confidentialité différentielle 3
- 1.3 Problématique et objectifs du mémoire 4
- 1.4 Méthodologie de la recherche 5
- 1.5 Contributions 7
- 1.6 Plan du mémoire 7

CHAPITRE 2 NOTIONS PRÉLIMINAIRES 8
- 2.1 Model-checking probabiliste 8
  - 2.1.1 Définitions basiques 8
  - 2.1.2 Chaine de markov à temps discret 9
  - 2.1.3 Processus de décision markovien 9
  - 2.1.4 Adversaire 10
  - 2.1.5 DTMC induit : 10

2.1.6 Logique temporelle probabiliste . . . 11
2.1.7 Le langage de modélisation PRISM . . . 12
2.2 Pseudométrique . . . 13
2.3 La confidentialité différentielle . . . 13
2.3.1 Termes de base . . . 15
2.3.2 Mécanismes privée . . . 16
2.3.3 Canal théorique de l'information . . . 16
2.3.4 Système différentiellement confidentiel . . . 17
2.4 Diagrammes de décision algébriques . . . 19
2.5 Preuves d'induction . . . 20
2.6 Induction discrètes . . . 20
2.7 Induction réel . . . 21
2.8 Couplage probabiliste . . . 22
2.8.1 Relation de levage : . . . 23
2.8.2 Utilisation du couplage et levage pour la confidentialité diffrentielle : . . . 23
2.9 Définitions de sécurité basées sur la trace : . . . 24
2.9.1 Relation de déroulement : . . . 24
2.10 Counter-example . . . 25

CHAPITRE 3 REVUE DE LITTÉRATURE . . . 26
3.1 Model-checker probabilistie pour la sécurité . . . 26
3.2 Vérification formelle de la confidentielité diffrentielle . . . 29
3.3 Les techniques de vérification de la confidentialité différentielle . . . 31
3.3.1 Non-interférence différentielle dans les systèmes interactifs : . . . 31
3.3.2 Bisimulation métrique dans les systèmes concurrent : . . . 33

CHAPITRE 4 ANALYSE FORMELLE DE LA CONFIDENTIALITÉ DIFFÉRENTIELLE AVEC PRISM . . . 34
4.1 Introduction . . . 34
4.2 Modèle conceptuel des systèmes différentiellement confidentiels . . . 35
4.3 Confidentialité différentielle : modèle et sémantique . . . 42
4.3.1 Espace de données . . . 42
4.3.2 Mécanisme de confidentialité . . . 43
4.3.3 Raisonnement probabiliste de la confidentialité différentielle . . . 44
4.3.4 Caractérisation de la confidentialité différentielle par PCTL* . . . 50
4.4 Analyse quantitative de la confidentialité différentielle . . . 53
4.4.1 La confidentialité différentielle : . . . 54

4.4.2 Précision : . . . 55
4.4.3 Quantitative de la perte de la confidentialité : . . . 56
4.4.4 La confidentialité différentielle changée : . . . 56
4.5 Notre méthodologie d'analyse de la confidentialité différentielle . . . 57
4.6 Modélisation de système différentiellement privé par PRISM . . . 58
4.7 Spécification de la notion de la confidentialité différentielle . . . 62
4.7.1 Analyse des modèles de la confidentialité differentielle avec PRISM . . . 64
4.8 Études de cas et résultats expérimentaux . . . 65
4.8.1 Modèle de la confidentialité différentielle . . . 66
4.8.2 Les résultats d'analyse : . . . 67
4.9 Conclusion . . . 71

CHAPITRE 5 VERIFICATION SYMBOLIQUE DE LA CONFIDENTIALITÉ DIFFRENTIELLE . . . 72
5.1 Introduction . . . 72
5.2 Approche pour la vérification de la confidentialité différentielle . . . 73
5.2.1 Principe . . . 73
5.3 Vérification de la confidentialité confidentielle . . . 75
5.3.1 Relaxation de la relation de lifting . . . 76
5.3.2 Les canaux d'un système confidentiellement privé . . . 77
5.3.3 Vérification de la confidentialité différentielle dans les chaînes de Markov : 78
5.3.4 Méthode . . . 79
5.4 Vérification symbolique de la confidentialité différentielle . . . 82
5.4.1 L'equivalent checking pour DP . . . 82
5.4.2 MTBDD pour DP . . . 82
5.4.3 Méthode symbolique . . . 89
5.5 Génération de contre-exemples pour la vérification de la confidentialité différentielle 91
5.5.1 Notion de contre-example en confidentialité différentielle : . . . 91
5.5.2 La confidentialité diffrentielle dirigé par les contre-exemples : . . . 95
5.6 Conclusion . . . 96

CHAPITRE 6 CONCLUSION . . . 97
6.1 Perspectives . . . 98
6.1.1 Vérification formelle de la confidentialité au niveau de l'utilisateur : . . . 98
6.1.2 Analyse formelle des mécanismes universellement optimaux : . . . 99
6.1.3 La confidentialité diffrentielle : temps, espace et l'aléa . . . 99

RÉFÉRENCES 101

ANNEXES 105

LISTE DES TABLEAUX

Tableau 1.1	Exactitude vs Assurance	5
Tableau 2.1	Les systèmes différentiellement confidentiels	17
Tableau 3.1	Model-checker probabilistie pour la sécurité	26
Tableau 3.2	Travaux de recherche pour la vérification formelle de la DP	29
Tableau 4.1	PCTL* pour DP	51
Tableau 4.2	Méchanime $\frac{1}{2}$-Géométrique Tronqué	66
Tableau 5.1	Les matrices probabilistes pour la confidentialité différentielle	76
Tableau 5.2	Encodage des actions et des états	83
Tableau 5.3	Les matrices probabilistes pour la confidentialité différentielle	86
Tableau 5.4	La matrice de canaux d'un système différentiellement confidentiel	88
Tableau 5.5	Table des comparaison	94

LISTE DES FIGURES

Figure 2.1	Système différentiellement confidentiel	17
Figure 2.2	MTBDD M, représentant une fonction $f_{\mathsf{M}} : \{0,1\}^4 \to [0,1]$	19
Figure 4.1	Modèle MDP d'un système différentiellement confidentiel	40
Figure 4.2	Mécanisme de confidentialité	43
Figure 4.3	ε-Confidentialité Différentielle	45
Figure 4.4	Confidentialité diffrentielle faible	46
Figure 4.5	Fragment du DTMC induite	48
Figure 4.6	Confidentialité diffrentielle forte	49
Figure 4.7	Méthodologie d'analyse de la confidentialité différentielle	57
Figure 4.8	Système différentiellement privée : synchrones et mutables	65
Figure 4.9	L'exploration d'un modèle DP à l'aide du simulateur	68
Figure 4.10	Probabilités des sorties dans un context dynamique	68
Figure 4.11	Exemple d'exécutions de APMC pour DP changé	69
Figure 4.12	Exemple d'exécutions de APMC pour DP	69
Figure 4.13	Vérificaion de la précision du confidentialité	70
Figure 4.14	Accessibilité attendue de la confidentialité différentielle	70
Figure 5.1	Approche proposée pour la vérification de la DP	74
Figure 5.2	Deux chaînes de Markov à temps discret pour la DP	77
Figure 5.3	Exemple d'un processus de décision markovien pour DP	85
Figure 5.4	MTBDD pour un modèle DP	87
Figure 5.5	ε-MTBDD pour un modèle DP	87
Figure 5.6	Un trou de sécurité dans un Modèle DP.	93

LISTE DES SIGLES ET ABRÉVIATIONS

APMC	Approximate Probabilistic Model Checking
DP	Differential Privacy
DTMC	Discrete-time Markov chain
IH	Induction Hypothesis
MTBDD	Multi-Terminal Binary Decision Diagrams
MDP	Markov decision process
QED	Quod Erat Demonstrandum (End of a mathematical proof)
QIF	Quantitative Information Flow
PMC	Probabilistic Model Checking
PCTL	Probabilistic Computation Tree Logic
PRISM	Probabilistic Symbolic Model Checker

LISTE DES ANNEXES

Annexe A Preuves 105

Annexe B Modèles PRISM pour la confidentialité différentielle 112

CHAPITRE 1 INTRODUCTION

La confidentialité différentielle est devenue l'une des notions de confidentialité les plus étudiées et les plus populaires au cours des dernières années [1]. Il s'agit d'une nouvelle approche à l'analyse de la confidentialité des données. De nombreuses organisations, maintenir de grandes collections des informations personnelles et privées, comme les dossiers des patients, les historiques de navigation, les données de localisation et les données de recensement que sont aujourd'hui disséminées dans une multitude de bases de données statistiques réparties dans différents établissements et services publics. D'une part, il est souvent souhaitable de divulguer des informations sur ces données pour améliorer les services, analyser les tendances, réaliser des études de marché, mener des recherches, etc. D'autre part, les fuites d'informations risquent de compromettre la vie privée des utilisateurs. Shmatikov et al [2] ont montré que des données anonymisées et agrégées, qui à première vue semblent sûres, peuvent en réalité révéler une quantité incroyable d'informations sur chaque utilisateur.

Un système différentiellement confidentiel est un système dans lequel les données sont traitées à plusieurs niveaux de sécurité. Généralement, il y a deux niveaux de sécurité, le bas niveau (*en anglais low level*) représente les données publiques qui sont des informations statistiques et le haut niveau (*en anglais High level*) représente les données privées dans les bases de données. La notion de confidentialité différentielle est similaire à celle des flux d'information basée sur la propriété de non-interférence [3] [4], alors la confidentialité différentielle est définie de manière à limiter les fuites d'informations sur les données privées en exigeant que le système produise des sorties similaires pour des bases de données ne différant que par une entrée. Les approches pratiques pour d'analyse des données privée les plus courantes sont :

1. **Demande de consentement des usagers :** on demande le consentement des usagers, ou bien on peut faire des analyses sans leur consentement malgré les dangers que menaçant les données, et ensuite on va obligatoirement gérer les plaintes. Cette technique arrive assez souvent en pratique, par exemple, un programme pour calculer le nombre des malades dans une ville sans portant révéler l'information privée d'un individu (le fait qu'Alice est malade), dans ce cas, nous attendons avoir un certain flux d'information et nous aurons un effet statistique sur le nombre de malades dans la ville.
2. **Anonymisation des données :** on modifie le contenu des données afin de rendre très difficile la réidentification, par exemple, en remplaçant le prénom de Alice par un identifiant (id=777144), mais cette technique fonctionne moyennement, il y a certains exemples bien connus [2] , où ils ont réussi à déanonymisé l'identifiant d'un individu et retrouver l'identité de quelqu'un à cause des données auxiliaires.

3. **Restreindre les requêtes :** on restreint des requêtes par un interpréteur que contrôlent les requêtes, par exemple, on permet de demander les nombres de malades, mais pas le nombre de malades qu'ont des cheveux bleus et conduisent une Ferrari, parce que quelqu'un pourrait savoir que seule Alice qui a des cheveux bleu et conduit une voiture Ferrari, il peut découvrir que Alice est malade. Cette technique ne fonctionne pas en pratique, parce que l'utilisateur peut déduire des informations confidentielles sur les entités individuelles (un attribut a une valeur particulière ou un attribut n'a pas de valeur particulière). Dans certains cas, une séquence de requêtes peut révéler des informations.
4. **La confidentialité différentielle** : on ajoute des bruits aux sorties, c.-à-d, des quantités aléatoires que changent légèrement les valeurs des sorties, par exemple, on ne donne pas le nombre exact de malades en sorte que la réponse soit compatible avec une situation dans un monde possible où Alice est malade et un autre monde possible où Alice n'est pas malade.

La notion de la confidentialité diffrentielle est une technique ad hoc et parfois inefficace pour les systèmes réel, il dispose des canaux supplémentaires pour avoir divulgué des informations privées.

L'utilité des méthodes formelles pour la description et la vérification de la confidentialité dans les systèmes est aujourd'hui largement reconnue, et certaines propriétés de confidentialité peuvent être étudiées dans un environnement probabiliste. Ce projet porte sur la modélisation formelle des systèmes différentiellement confidentiels. Nous proposons des techniques de vérification automatisées pour l'analyse formelle des propriétés quantitatives de tels systèmes. Nous construisons ainsi des preuves de confidentialité du système.

En conclusion, nous visons à offrir un cadre général uniforme capable d'analyser la confidentialité différentielle d'une grande classe de systèmes probabilistes.

1.1 Motivation

Plusieurs méthodes formelles basées sur la sémantique des programmes, peuvent s'appliquer à la vérification de la confidentialité différentielle. Beaucoup de chercheurs en langage de programmation ont développé des approches basées sur l'analyse dynamique, les systèmes de typage et la logique de programmes afin de prouver formellement la confidentialité différentielle des programmes [5]. La présence de non-déterminisme reste courante dans ce type de systèmes et nécessite donc d'être prise en compte pour assurer une vérification exhaustive.

La vérification formelle de la confidentialité différentielle dans les systèmes distribués est bien établie dans les travaux théoriques. En particulier, dans [6] ils ont développés des algorithmes de vérification basés sur une forme de relation de bisimulation probabiliste pour caractériser le

comportement du système au sens de la confidentialité différentielle. Ils ont modélisé les aspects d'un système différentiellement confidentiel par un automate probabiliste qui admet à la fois un comportement non déterministe et probabiliste.

Cependant, l'implantation de ces algorithmes n'est pas encore réalisée.

1.2 Définitions et concepts de base

1.2.1 La vie privée

Les nouvelles technologies font peser de nouvelles menaces sur les droits à la vie privée. Une théorie sophistiquée appelée "technologie de confiance" a contribué à la base scientifique des méthodes formelles appliquées à la vérification de la protection de la vie privée. En principe, les méthodes formelles sont adaptées pour assurer la confidentialité, mais aucune analyse comparative qui résume quelles méthodes particulières sont les meilleures en pratiques, ou en théorie, n'est disponible. La vérification de la vie privée diffère de la correction fonctionnelle du système, parce que nous devrons inclure non seulement des spécifications logiques, mais des méthodes rigoureuses et précises que peuvent nous aider à documenter les objectifs et servir de base à la compréhension et à la discussion des exigences de confidentialité.

1.2.2 Vérification automatique

Le thème de la vérification automatique étudie la théorie et la pratique de la vérification formelle des systèmes informatiques. La vérification formelle est une approche qui utilise des techniques mathématiques pour confirmer l'exactitude d'un système au cours de sa phase de conception, à partir d'un modèle donné, d'une description de l'environnement dans lequel le système sera exécuté et de spécifications des propriétés de correction que doivent satisfaire le système. De plus, cela nous permet souvent de détecter des scénarios qui pourraient invalider les spécifications. Les méthodes formelles peuvent être un excellent moyen de trouver des bogues dans les conceptions des systèmes différentiellement confidentiels, car elles obligent le concepteur à tout rendre explicite et donc à confronter des choix de conception difficiles qui pourraient autrement être falsifiés.

1.2.3 La confidentialité différentielle

La confidentialité différentielle est une norme robuste pour la vie privée des individus dans les bases de donnés statistiques proposée par Cynthia Dwork. Elle consiste à s'assurer que le résultat d'une requête devrait être formellement indistinguable avec ou sans un seul enregistrement [7]. La confidentialité différentielle protège les individus en fournissant des réponses randomisées aux re-

quêtes statistiques de manière à ce que le résultat d'une requête arbitraire sur deux bases de données ne différant que d'une entrée soit indistinguable. L'accès aux données privées est limité aux fonctions désinfection *(sanitization functions)*, qui sont des algorithmes randomisés, qui ajoutent une quantité appropriée de bruit aux résultats des requêtes. Il existe différentes plateformes pour préserver la confidentialité différentielle, comme PINQ (*Privacy Integrated Queries*) [8], Airavat [9]. Un budget de la confidentialité est attribué par toutes ces plateformes. Il a pour fonction de limiter la divulgation de l'information afin de s'assurer qu'une requête et son agrégation avec les requêtes précédentes ne révèle pas la confidentialité de l'individu. Une fois que le budget de confidentialité d'une base de données est complètement utilisé, la base de données devient inaccessible. Ces plateformes empêchent les analystes d'utiliser la base de données même si une petite partie seulement de la base de données est impliquée dans les requêtes précédentes.

1.3 Problématique et objectifs du mémoire

Nous voulons analyser la notion de la confidentialité différentielle dans les systèmes distribués par les techniques formelles du model checking probabiliste. Parmi les techniques d'analyse des propriétés d'un système, les techniques du model checking probabilistes sont largement utilisées. De nombreux outils d'analyse sont librement disponibles : Storm, mcsta, PRISM, et d'autre [10]. Mais, à notre connaissance, au moment de la rédaction de ce mémoire, il n'existe pas d'outils de vérification permettant des analyses de la confidentialité différentielle directement sur des modèles probabilistes.

La notion de la confidentialité différentielle ne peut pas être exprimée sous forme de propriétés de traces. Elle caractérise le fait qu'aucune information ne provient d'un sujet de haut niveau (par exemple, l'absence ou présence d'un individu) soumis à un sujet de bas niveau (par exemple, public ou analyste). Or ceci ne peut s'exprimer que par une propriété d'ensembles de traces.

Dans ce mémoire, l'objectif global est de développer des algorithmes automatiques pour vérifier la confidentialité différentielle et de proposer un cadre pour analyser formellement certaines propriétés de confidentialité exprimées en logique temporelle probabiliste, sur ces modèles probabilistes. Grâce à l'utilisation de formules probabilistes, il est possible d'exprimer des informations sur le coût et la perte de la confidentialité et ainsi de spécifier des propriétés plus complexes et plus pertinentes dans le contexte de la vérification des mécanismes confidentiels dans un environnement distribué.

1.4 Méthodologie de la recherche

Dans cette section, nous décrivons la méthodologie qui a été suivie lors de ce projet. Nous identifions les questions de recherche que nous avons analysées et nous décrivons les critères de recherche qui ont été suivis.

Questions de recherche : notre objectif dans ce projet est de fournir des algorithmes automatiques pour vérifier la confidentialité différentielle dans les systèmes. À cette fin, nous cherchons à répondre aux trois questions de recherche suivantes :

QR1 : Quels sont les défis lors de la spécification et de la vérification formelle du comportement des systèmes différentiellement confidentiels ?
Réponse : pour tout système informatique nous intéressons à sa spécification, sa mise en œuvre et le problème de la correction de sa mise en œuvre par rapport à sa spécification. Dans le domaine de la confidentialité des systèmes, nous avons un vocabulaire très particulier, et le tableau 1.1 illustre ces différences, alors la garantie que le système soit différentiellement confidentiel ne doit pas dépendre de la confidentialité des données et le secret de sa spécification et son implémentation, il faut vraiment que les politiques de confidentialité soient appropriées et que la mise en œuvre des mécanismes de confidentialité soient correcte. Dans les systèmes interactive, il assez difficile de prévoir toutes les actions de l' analyste des données (ou l'attaquant) et tester tous les cas que pourrait présenté.

Tableau 1.1 Exactitude vs Assurance

Phases	Pour tout système	Système différentiellement confidentiel
Spécification	Définition du fonctionnement de système	Politique de confidentialité
Implémentation	Mise en oeuvre du système	Mécanismes de confidentialité
Correction	Exactitude	Assurance

QR2 : Quels sont les formalismes, outils et approches utilisés pour répondre à la réponse de **QR1** ?
Réponse : Il y a énormément des méthodes pour spécifier et vérifier la correction des systèmes. Dans le domaine de la sécurité et la confidentialité, nous espérons utiliser ces méthodes qu'ont certains succès dans d'autres domaines pour améliorer et assurer la confidentialité différentielle dans les systèmes. Or la confidentialité différentielle a une caractéristique bien spécifique que influence comment et où nous pouvons faire des modèles efficaces et des preuves. Nous voulons avoir des garanties de la confidentialité différentielle avec des hypothèses trés faible sur l'analyste des données, parce que la confidentialité différentielle doit être à l'abri contre le post-traitement [1], c.-à-d. un analyste de données avec ou sans connaissances supplémentaires sur la base de données, il ne peut pas augmenter la perte de la confidentialité d'un individu.

Nous avons utilisé des modèles probabilistes qui présentent les caractéristiques essentielles des systèmes différentiellement confidentiels tout en évitant les détails non pertinents, nous avons été conscients au niveau des abstractions que nous faisons et sensibles aux problèmes de sécurité que nous pouvons ignorer.

La méthodologie proposée repose sur les modèles de Markov non déterministe et probabiliste, et la vérification du modèle probabiliste comme outil de vérification. Il convient de noter que les mécanismes de confidentialité sont des programmes probabilistes et l'interaction entre l'analyste de données et le système est incertaine. Le problème de vérification des modèles pour les systèmes différentiellement confidentiels est en général indécidable, mais ce projet présente, comme une autre contribution originale, nous avons utilisé la vérification de modèle probabiliste pour obtenir la modélisation de canaux de ces systèmes, nous en ayant déduit ensuite les valeurs maximales de la confidentialité différentielle, qui sont des limites robustes en matière de fuite d'informations, quelle que soit la distribution de probabilité conditionnelle sur les valeurs secrètes et les valeurs observables par l'analyste de données.

QR3 : Quelles sont les limites des réponses à **QR2**, et y a-t-il des solutions de développement visant à y faire face ?
Réponse : Les méthodes qui sont basées sur la vérification de modèle, elles sont confrontées au problème d'explosion combinatoire de l'espace d'états. Pour vérifier la confidentialité différentielle, l'outil de vérificateur de modèles probabiliste doit capturer tous les états observables du modèle, et comparer les probabilités de chaque paire de chemins ainsi que chaque chemin avec tous les autres chemins pendant plusieurs exécutions. La confidentialité différentielle dépend de l'adversaire dans les systèmes concurrents, un système est différentiellement confidentiel sous un adversaire particulier peut ne pas être différentiellement confidentiel sous un autre adversaire. Pour cette raison, ils sont confrontés au problème d'explosion de l'espace d'états. Nous avons l'intention d'élaborer les algorithmes symboliques pour détecter l'équivalence des exécutions des programmes probabiliste, ou chaque exécution d'un programme est sous le contrôle de son adversaire. Cette relation d'équivalence basée sur le principe de couplage que décrivant les entrées confidentielles qui ne diffèrent que par une entrée et ne peuvent pas être distinguées par les sorties publiques d'un programme probabiliste dans un scénario d'attaque donné par un adversaire.
Les analyses de la confidentialité différentielle peuvent être précises ou approximatives, d'un point de vue, nous avons proposé également un type de calcul à point fixe pour calculer la perte de confidentialité, car l'énumération des cas dans l'ensemble d'états et les relations d'équivalence entre les transitions probabiliste est trop grande. Au lieu de cela, nous pouvons utiliser les techniques symboliques basé sur les diagrammes de décision binaires multiterminaux. Au lieu de représenter des ensembles d'états et des transitions d'états par une énumération explicite des cas, nous pouvons faites-le symboliquement, en utilisant des formules booléennes.

1.5 Contributions

Les principales contributions de ce mémoire sont de :

- Fournir un cadre complet pour l'analyse formelle de la notion de la confidentialité différentielle pour des systèmes présentant un comportement à la fois probabiliste et non déterministe. Nous le faisons en développant conjointement les fondements théoriques, y compris les modèles et les spécifications des propriétés, ainsi que la mise en œuvre des techniques et leur application à plusieurs études de cas pertinentes de systèmes différentiellement confidentiels concurrents ;
- Développer la première approche de modélisation, simulation et vérification de la confidentialité différentielle avec un model-checker probabiliste et appliquer à une série d'études de cas approfondies ;
- Proposer des algorithmes symboliques pour la vérification de ces systèmes représentés à l'aide de diagrammes de décision binaires multiterminaux (MTBDD).
- Proposer une méthode d'extraction de contrexemple de la confidentialité différentielle.

1.6 Plan du mémoire

Ce mémoire est divisé en six chapitres et deux annexes.

Nous commençons par détailler dans le chapitre 2 le contexte scientifique et technique de ce mémoire. Le chapitre 3 présente une revue de la littérature et compare nos travaux, présentés dans ce mémoire, aux travaux connexes existants.
Nous proposons au chapitre 4 un cadre théorique pour modéliser les systèmes différentiellement confidentiels dont la sémantique s'exprime par des Processus de Décision Markoviens. Nous proposons une méthode de vérification de la notion la confidentialité de ces systèmes en utilisant l'outil PRISM.
Dans le chapitre 5, nous présentons un algorithme de vérification de la confidentialité différentielle dans les chaînes de Markov basé sur les couplage et la relation de *lifting*. Nous développons aussi une méthode symbolique de vérification de la confidentialité dans les Processus de Décision Markoviens. Finalement, nous proposons, une méthode de génération de contrexemplel, c'est-à-dire d'exécutions qui violent la confidentialité différentielle lors de la vérification.

Nous terminons ce document en établissant une conclusion sur l'ensemble des travaux présentés ainsi qu'une analyse des perspectives possibles.

En annexe, Nous exposons les preuves, le pseudo-code des algorithmes proposés au chapitre 5 et le modèle complet du système présenté au chapitre 4.

CHAPITRE 2 NOTIONS PRÉLIMINAIRES

Nous présentons dans ce chapitre le contexte scientifique des travaux présentés dans ce mémoire, c'est-à-dire la vérification formelle probabiliste et la notion de la confidentialité différentielle que pouvant être vérifiées dans les systèmes distribués. Nous fournissons aussi le contexte mathématique nécessaire, puis établissons un lien profond entre le raisonnement probabiliste et les programmes différentiellement confidentiels, cette observation est la principale contribution conceptuelle et constitue le fondement de ce projet.

2.1 Model-checking probabiliste

Dans cette section, nous donnons une brève introduction de la vérification de modèle probabiliste. *Model-checking probabiliste* [11] est une méthode formelle de vérification automatique et d'analyse quantitative des systèmes probabilistes. Les objectifs d'un algorithme de vérification de modèle probabiliste est pour décider si le modèle satisfait une propriété de logique temporelle probabiliste et pour trouver la valeur numérique (probabilité ou récompense) dans laquelle la propriété est valide.

2.1.1 Définitions basiques

Soit $\mathbb{R}$ l'ensemble des nombres réels et $\mathbb{N}$ l'ensemble des entiers non négatifs.

Définition 1. *Une distribution de probabilité discrète sur un ensemble dénombrable Q est une fonction $\mu : Q \to [0, 1]$ tel que $\sum_{q \in Q} \mu(q) = 1$. Soit $Dist(Q)$ l'ensemble des distributions de probabilités discrètes sur Q. Pour un ensemble éventuellement non dénombrable Q, laissez $Dist(Q')$ soit l'ensemble des distributions de probabilité discrètes sur des sous-ensembles dénombrables de Q'.*

On note le support d'une mesure discrète μ, c'est-à-dire l'ensemble des éléments de Q qui ont une mesure non nulle, par $supp(\mu)$. Formellement : $supp(\mu) \triangleq \{q \in Q \mid \mu(q) > 0\}$.

Définition 2. *Une sous-distribution discrète sur un ensemble dénombrable Q est une fonction $\mu : Q \to [0, 1]$ prenant pour chaque élément de Q à un poids tel que la somme de leurs poids est au plus 1 :*

$$\sum_{q \in Q} \mu(q) \leqslant 1$$

Définition 3. *Une variable aléatoire sur un ensemble dénombrable Q est une fonction $X : Q \longrightarrow \mathbb{R}$. Afin de définir les mesures de probabilité, le concept de plus petite σ-algèbre est utilisé. Plus de détails peuvent être trouvés dans [12].*

2.1.2 Chaine de markov à temps discret

Une chaîne de Markov à temps discret (DTMC) est un modèle probabiliste qui définit la probabilité de passer d'un état à un autre. Ils peuvent être utilisés pour modéliser un seul système probabiliste ou plusieurs systèmes similaires composés de manière synchrone.

Une DTMC est un tuple $D = (S, \bar{s}, \mathbf{P}, L)$

- S est un ensemble fini d'états ;
- $\bar{s}$ est un état initial ;
- $\mathbf{P} : S \times S \to [0, 1]$ est la matrice de transition probabiliste.
- $L : S \to 2^{AP}$ est la fonction d'étiquetage.

$\mathbf{P}$ est une matrice stochastique qui satisfait $\sum_{s' \in S} \mathbf{P}(s, s') = 1$ pour tout $s \in S$. Un chemin (fini ou infini) ρ est une séquence (finie ou infinie) $\rho = s_0 s_1 ...$ d'états tels que $\mathbf{P}(s_i, s_{i+1}) > 0$ pour tous $i \geq 0$. Un chemin fini $\rho = s_0 s_1 ... s_n$ a une longueur $|\rho| = n$; pour des chemins infinis, nous mettons $|\rho| = \infty$.

2.1.3 Processus de décision markovien

Le processus de décision markovien (MDP) est une généralisation de DTMC. Un MDP peut décrire à la fois un comportement non déterministe et probabiliste. Il est bien connu que le non-déterminisme est un outil précieux pour la modélisation de la simultanéité, et le MDP permet de décrire le comportement de plusieurs systèmes probabilistes fonctionnant en parallèle. Le non-déterminisme est également utile lorsque la probabilité exacte d'une transition n'est pas connue ou bien lorsqu'elle est connue, mais n'est pas considérée comme pertinente.

Un processus de décision markovien (MDP) est un tuple $\mathcal{M} = (S, \bar{s}, A, \delta, L)$ où :

- S est un ensemble fini d'états ;
- $\bar{s}$ est un état initial ;
- A est un ensemble fini d'actions ;
- $\delta : S \times A \to Dist(S)$ est une fonction de transition probabiliste partielle pour attribuer des paires état-action à des distributions de probabilité sur S ;
- $L : S \to 2^{AP}$ est une fonction d'étiquetage d'état .

Dans un état d'un MDP, il y a d'abord un choix non déterministe entre un ensemble d'actions qui sont *disponible* dans l'état. Cet ensemble est noté pas $A(s)$, inclut les actions pour lesquelles une transition probabiliste est définie : $A(s) = \{a|\delta(s,a) \text{ est défini}\}$.
Nous supposons que l'ensemble $A(s)$ n'est pas vide pour tous les états $s \in S$ et une fois une action disponible une $a \in A(s)$ a été choisi dans s, l'action est effectuée et l'état successeur s' est choisi de manière probabiliste, où la probabilité de passer à l'état s' est noté par $\delta(s,a)(s')$.

2.1.4 Adversaire

Pour raisonner formellement sur le comportement probabiliste d'un MDP, nous avons besoin d'un espace de probabilité sur des chemins infinis. Par définition, un chemin infini est une séquence infinie d'états et d'actions $s_0, a_0, s_1, a_1, ...$, où $s_i \in S$, $a_i \in A$ et $\delta(s_i, a_i)(s_{i+1}) > 0$ pour tout $i \geq 0$. Cependant, un espace de probabilité ne peut être construit que lorsque tout le non-déterminisme actuel a été résolu. Chaque résolution possible du non-déterminisme est représentée par un adversaire, qui est chargé de choisir une action dans chaque état du MDP, sur la base de l'historique de son exécution jusqu'à présent. Selon le contexte, les adversaires sont souvent désignés par divers noms, y compris *stratégies*, *ordonnanceurs* et *politiques*.

2.1.5 DTMC induit :

Un adversaire résout les choix non déterministes en fonction du chemin fini qui a conduit à l'état actuel en attribuant des probabilités aux choix non déterministes disponibles dans le dernier état d'un chemin fini. Il transforme donc le modèle non déterministe en un modèle entièrement probabiliste, alors le MDP résultant est déterministe par rapport au choix des actions, ce qui donne sémantiquement un DTMC induit dont les états sont les chemins finis du MDP qui commencent dans les états initiaux.

Pour un MDP $M = (S, \bar{s}, \alpha_M, \delta_M, L)$ et un adversaire $\sigma \in Adv_M$, la DTMC induit est
$M_\sigma = \{T, \bar{s}, \mathbf{P}, L'\}$ où :

— $T = FPath_M$

— Pour deux $\rho, \rho_0 \in FPath_M$:

$$\mathbf{P}(\rho, \rho_0) = \begin{cases} \sigma(\rho)(a).\delta_M(last(\rho), a)(s) & si\ \rho' = \rho a s, a \in A(last(\rho))\ et\ s \in S \\ 0 & sinou; \end{cases}$$

— $L' = L(last(\rho))$ pour tous $\rho \in FPath_M$.

Adversaire sans mémoire :

Un adversaire σ est sans mémoire si $\sigma(\rho)$ ne dépend que de $last(\rho)$, c'est-à-dire pour tout deux chemins $\rho, \rho' \in FPath_M$ tel que $last(\rho) = last(\rho')$, on a $\sigma(\rho) = \sigma(\rho')$.

C'est un adversaire particulier que choisit toujours le même choix dans un état, si pour tout chemin $\sigma(s_0(a_0, \mu_0)s_1...s_n)$ ne dépend que de l'état s_n , il est écrit comme un mappage d'états, c'est-à-dire $\sigma(s)$ pour chaque DTMC induit par $s \in S$ peut être mappé à $|S|$-états DTMC.

2.1.6 Logique temporelle probabiliste

Les propriétés des modèles probabilistes peuvent être exprimées à l'aide de la logique PCTL (*Probabilistic Computing Tree Logic*) [13], qui étend la logique CTL (*Calculary Tree Logic*) [14] par l'ajout de l'opérateur probabiliste P. Dans PCTL, les formules d'état ϕ sont interprétées sur les états d'un modèle et les formules de chemin ψ sont interprétées sur les chemins.
La syntaxe des formules d'état et des formules de chemins sont presque les mêmes pour les DTMCs et les MDPs, mais avec une quantification implicite sur les adversaires.

La syntaxe de PCTL est la suivante :

$$\phi ::= true \mid a \mid \phi \wedge \phi \mid \neg\phi \mid P_{\bowtie p}[\psi]$$

$$\psi ::= \mathcal{X}\phi \mid \phi \mathcal{U}^{\leq k} \phi \mid \phi \mathcal{U} \phi$$

où a est une proposition atomique, $\bowtie \in \{<, >, \geq, \leq\}$ est un opérateur relationnel, $p \in [0, 1]$, $k \in \mathbb{N}$. Notez que chaque proposition atomique a doit être extrait de l'ensemble utilisé pour étiqueter les états de modèle. Une propriété d'un modèle toujours être exprimée comme une formule d'état. Les formules de chemin se produisent uniquement en tant que paramètre du opérateur de chemin probabiliste. Intuitivement, la formule $P_{\sim p} = [\psi]$ signifie que le chemin ψ est vraie avec la probabilité $\bowtie p$, le résultat de l'analyse de cette propriété qualitative est vrai ou faux. La propriété quantitative s'exprime par $P_{=?}[\psi]$ pour évaluer la probabilité que la formule de chemin ψ soit vraie.

Pour le MDP, l'opérateur $P_{=?}[\psi]$ a deux variants qui sont $P_{min=?}$ et $P_{max=?}$, pour calculer la probabilité maximale ou minimale que la formule de chemin ψ soit satisfaite, en considérant tous les adversaires possibles.

Pour les formules de chemin, nous autorisons les opérateurs $\mathcal{X}$(suivant), $\mathcal{U}^{\leq k}$ (borné jusqu'à) et $\mathcal{U}$ (jusqu'à) qui sont standard dans la logique temporelle. Intuitivement, $\mathcal{X}\phi$ est vrai si ϕ est satisfait dans le prochain état ; $\phi_1 \mathcal{U}^{\leq k} \phi_2$ signifie que ϕ_2 est vraie en k étapes et ϕ_1 est vraie jusqu'à cet état ; et $\phi_1 \mathcal{U} \phi_2$ signifie que ϕ_1 est toujours vraie jusqu'à un état où ϕ_2 est vraie.

2.1.7 Le langage de modélisation PRISM

Dans cette sous-section, nous décrivons le langage de modélisation natif du vérificateur de modèle probabiliste PRISM que est utilisé dans ce mémoire. Nous commençons par définir la syntaxe du langage, puis décrivons la sémantique et concluons en présentant un exemples. Une description complète de PRISM est disponible dans [15].

L'outil nécessite principalement deux entrées, à savoir la description du système à analyser et un ensemble de propriétés à vérifier liées au système. Comme pour la description, un système peut être modélisé soit en tant que chaîne de Markov à temps discret (DTMC) permettant des comportements probabilistes, un processus de décision de Markov (MDP) qui étend la DTMC en permettant des comportements non déterministes,

Les composants fondamentaux du langage PRISM sont des modules et des variables. Un système peut être constitué de plusieurs modules pouvant interagir. Chaque module contient un certain nombre de variables dont les valeurs déterminent l'état du module à un moment donné. De plus, un module peut être spécifié comme suit :

```
module <name> ... endmodule
```

Dans PRISM, il existe deux types de variables, à savoir les variables booléennes et les variables entières. Voici un exemple comment déclarer une variable booléenne : Il déclare une variable booléenne secret, initialisée à la valeur false. Alors que la déclaration suivante,

```
secret : bool init false;
```

déclare une longueur variable entière en nombres entiers allant de 0 à 7 et a la valeur initiale 0.

```
length : [0..7] init 0;
```

Les transitions probabilistes dans PRISM sont spécifiées en utilisant des commandes gardées qui ont la forme :

```
[] <guard> -> <prob1> : <command1> + ... + <probN> : <commandN>;
```

où **guard** est un prédicat sur les variables système, **prob1** une distribution de probabilité discrète et **command** est une transition exécutée de manière probabiliste par le système lorsque **guard** évalue à la valeur **true**. Les transitions effectuées par la commande sont une mise à jour des valeurs des variables locale.

2.2 Pseudométrique

La notion d'adjacente est cruciale dans la définition de la confidentialité différentielle, cela implique que la distance entre les bases de données est mesurée par le nombre de lignes dans lesquelles elles sont différentes, c'est-à-dire par la métrique de Hamming. Néanmoins, la définition peut être adaptée en utilisant d'autres notions de distance pour faire l'analyse formelle des modèles. Une métrique est une fonction sur un ensemble X (appelée aussi une fonction de distance) :

$$d : X \times X \longrightarrow \mathbb{R}^+$$

de telle sorte que, pour chaque $x, y, z \in X$,

1. **Axiome de coïncidence :** $d(x, y) = 0$ ssi $x = y$
2. **Symétrie :** $d(x, y) = d(y, x)$
3. **Inégalité triangulaire :** $d(x, z) \leqslant d(x, y) + d(y, z)$

2.3 La confidentialité différentielle

La confidentialité différentielle est une définition mathématique rigoureuse de la vie privée. Dans le contexte le plus simple, en considérons un algorithme qui analyse une base de données et calcule des statistiques à son sujet telles que la somme, la moyenne, la variance, la médiane, etc. Un tel algorithme est dit différentiellement confidentiel si, en regardant la sortie, on ne peut pas savoir si les données d'un individu ont été incluses dans la base de données d'origine ou non. En d'autres termes, la garantie d'un algorithme différentiellement confidentielle est que son comportement ne change pratiquement pas lorsqu'un individu isolé rejoint ou quitte la base de données, et tout ce que l'algorithme pourrait afficher sur une base de données contenant des informations sur un individu est presque aussi susceptible de provenir d'une base de données et sans les informations de cette personne. Plus particulièrement, cette garantie est valable pour tout individu et tout ensemble de données. Par conséquent, quels que soient les détails d'un individu et de quelqu'un d'autre dans la base de données, la garantie de confidentialité différentielle tient toujours. Cela donne une garantie formelle que des informations individuelles sur les participants à la base de données ne sont pas divulguées.

Les bases de données sont supposées être des vecteurs dans $\mathcal{D}^n$ pour certains domaines $\mathcal{D}$. La distance de Hamming $d_H(x, y)$ sur $\mathcal{D}$ n'est le nombre de positions dans lesquelles les vecteurs x, y diffèrent. On laisse $Pr[\cdot]$ pour dénoter une probabilité. Étant donné un algorithme aléatoire $\mathcal{A}$, on laisse $\mathcal{A}(x)$ être la variable aléatoire (ou distribution de probabilité sur les sorties) correspondant à

l'entrée x. Si X et Y sont des distributions de probabilité (ou des variables aléatoires) sur un espace discret $\mathcal{D}$, la différence statistique entre X et Y est définie comme suit :

$$\Delta(X,Y) = \max_{S \subset D} |Pr[X \in S] - Pr[Y \in S]|$$

La définition de la confidentialité différentielle est issue d'une longue série de travaux appliquant des idées algorithmiques à l'étude de la confidentialité [16], [17], [18] aboutissant au travail de [19].

Définition 4. *(ε-differential privacy [19]) Un algorithme randomisé $\mathcal{A}$ est différentiellement privé si pour toutes les bases de données x, $y \in \mathcal{D}^n$ à une distance de Hamming d'au plus 1, et pour tous les $S \subseteq Range(\mathcal{A})$,*

$$Pr[\mathcal{A}(x) \in S] \leq e^{\varepsilon}.Pr[\mathcal{A}(y) \in S].$$

Cette définition précise que la modification des données d'un individu dans la base de données entraine une légère modification de la répartition des résultats. Contrairement aux mesures plus standards de la distance telles que la différence statistique et la métrique ici est multiplicative et même les évènements très improbables doivent avoir approximativement la même probabilité sous les distributions $\mathcal{A}(x)$ et $\mathcal{A}(y)$.
Pour que $\mathcal{A}$ donne une confidentialité différentielle, nous considérons tout sous-ensemble de sorties $S \subseteq Range(\mathcal{A})$, et que x, y soit une paire quelconque de bases de données différentes dans au plus un élément. Lorsque la base de données est x, la masse de probabilité pour tout $r \in Range(\mathcal{A})$ est proportionnelle à e^{ε}, et de même lorsque la base de données est y.

Définition 5. *((ε,δ)-differential privacy [7]) Un algorithme randomisé $\mathcal{A}$ est différentiellement privé si pour toutes les bases de données x, $y \in \mathcal{D}^n$ qui diffèrent par une entrée et pour tous $S \subseteq Range(\mathcal{A})$,*

$$Pr[\mathcal{A}(x) \in S] \leq e^{\varepsilon}.Pr[\mathcal{A}(y) \in S] + \delta.$$

La confidentialité différentielle garantie qu'un algorithme randomisé $\mathcal{A}$ se comportera de manière similaire sur des bases de données similaires x et y.

2.3.1 Termes de base

Jeu de données (en anglais Datasets) :

Nous modélisons un ensemble de données X comme un tuple de n éléments $(x_1, ..., x_n)$, où chaque $x_i \in U$ telle que U est l'ensemble de l'univers des données.

On considère deux modèles des valeurs :

1. Modèle de valeur non-sensible :
   Dans ce modèle nous mesurons le coût de la confidentialité uniquement par rapport à la manière dont le mécanisme traite le bit sensible, et ignorons la manière dont il traite les valeurs privées, où $U = \{0, 1\}$.

2. Modèle de valeur sensible :
   Dans ce modèle nous mesurons la vie privée par rapport à la façon dont elle traite la paire (b_i, v_i) pour chaque individu ou point d'entrée, où $U = \{0, 1\} \times \mathbb{N}^+$

Il y a deux sortes de jeu de donnés selon leur changement des valeurs :

1. Jeu de données évolutives *(mutable datasets)* : l'ensemble de données qui changent au fil du temps se présente assez souvent dans la pratique (par exemple, l'ensemble de données des dossiers de santé des patients dans un hôpital)
2. Jeu de données non évolutives *(immutable datasets)* : l'ensemble de données qui ne changent pas au fil du temps (par exemple, l'ensemble de données des transactions bancaires de l'année précédente d'une société).

Requête de comptage :

Une requête de comptage (*en anglais Counting query*) est une requête pour compter le nombre d'éléments dans la base de données satisfont un prédicat, c'est une requête primitive et extrêmement puissante, parce qu'il y a des nombreuses tâches dans d'analyse de données *(standard datamining)* et dans les statistiques utilise ce type de requête.
Cette requête a deux formes pure (2.1) et fractionnelle (2.2) :

$$q(D) = \sum_{i=0}^{n} q(x_i) \tag{2.1}$$

$$q(D) = \frac{1}{n} \sum_{i=0}^{n} q(x_i) \tag{2.2}$$

2.3.2 Mécanismes privée

Dans ce projet, nous nous sommes concentrés sur les distributions discrètes. Cependant, les mécanismes randomisés utilise d'échantillonnage à partir de distributions continues sont assez courants dans la littérature [8] utilisent des échantillons de distributions à valeur réelle comme la distribution gaussienne et la distribution standard de Laplace.

Un mécanisme randomisé dans un modèle discret est une application $\mathcal{K} : \mathcal{D} \longrightarrow \mathcal{R}$, où $\mathcal{R} = \{0, .., n\}$ et n est la range du mécanisme. Nous écrivons $Pr[i|j]$ pour la probabilité conditionnelle que la sortie $\mathcal{K}(j) = i$ (sur l'entrée $j \in \mathcal{D}$) soit $i \in R$.

Le mécanisme mappe les entrées dans la plage 0 à n aux sorties dans la même plage : $\mathcal{K} : \{0, .., n\} \longrightarrow \{0, .., n\}$. Il est donc naturel de représenter $\mathcal{K}$ comme une matrice carrée $\mathcal{P}_{n\times n}$, où $\mathcal{P}_{(i,j)} = Pr[\mathcal{K}(j) = i] = Pr[i|j]$. Nous abrègerons cette probabilité en $Pr[i|j]$. Notez donc que P est une matrice stochastique de colonne. [1] Les lignes sont des secrets ou les colonnes sont des observables.

2.3.3 Canal théorique de l'information

Les systèmes peuvent être modélisés comme un canal théorique de l'information, Un *canal* est un triple $(\mathcal{X}, \mathcal{Y}, \mathcal{C})$, où $\mathcal{C} : \mathcal{X} \times \mathcal{Y} \to [0, 1]$ est une fonction dans laquelle $\mathcal{X}$ est un ensemble de valeurs d'entrée secrètes, $\mathcal{Y}$ est un ensemble de valeurs de sortie observables. Tout simplement, $\mathcal{C}$ est une matrice de canal, $|\mathcal{X}| \times |\mathcal{Y}|$ matrice dont les entrées $\mathcal{C}(x, y)$ représente la probabilité conditionnelle du canal produisant la sortie $y \in \mathcal{Y}$ étant donné l'entrée $x \in \mathcal{X}$.

Chaque canal $\mathcal{C}$ satisfait deux conditions (2.3) et (2.4) :

$$\mathcal{C}(x, y) \in [0, 1] \tag{2.3}$$

$$\sum_{y\in\mathcal{Y}} \mathcal{C}(x, y) = 1 \tag{2.4}$$

1. les entrées de chaque colonne peuvent être interprétées comme des probabilités et leur somme égale à 1

2.3.4 Système différentiellement confidentiel

Plusieurs auteurs ont récemment proposé des méthodes pour raisonner la confidentialité différentielle sur la base de langages et de modèles de calcul différents, par exemple, les langages de type SQL [8], les langages fonctionnels d'ordre supérieur [20], le modèle MapReduce [9], la figure suivante 2.1 et et le tableau 2.1 présentent différents systèmes différentiellement confidentiels :

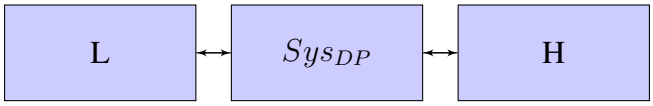


Figure 2.1 Système différentiellement confidentiel

Tableau 2.1 Les systèmes différentiellement confidentiels

H	L	Sys_{DP}
Data provider	Analyste	PINQ
Data providers	Computation provider	Airavat
Data Owner	Data Analyste	GUPT

Catégories :

Les systèmes différentiellement confidentiels peuvent être classés en quatre catégories :

1. **Asynchrone :** La sortie randomisée n'est pas en fonction de la séquence des requêtes, mais en fonction de leur ordre (par exemple, un système utilise un ordonnanceur pour déterminer l'ordre des requêtes afin d'améliorer les performances).
2. **Synchrone :** La sortie randomisée est en fonction de la séquence des requêtes.
3. **Conservative :** ce système utilise une technique de décision que définie en termes du budget de confidentialité pour répondre aux requêtes, par exemple, un système refuse de répondre à d'autres requêtes si son budget est épuisé, parce que s'il répond un autre requête ferait $(\varepsilon > budget)$ plus grand que la borne .
4. **Non-Conservative :** ce système possède une base de données modifiable (en anglais *mutable database*) et peut fournir des limites d'erreurs de confidentialité restreintes en éliminant les données surutilisées.
   Il y a deux types du système non conservatifs :

(a) **Avec interruption :** un système qui permet à d'interrupter la réception des données lorsque sa mémoire est plein par l'envoie d'un message (blocage des entrées ou interruption instantané de l'envoi des données).

(b) **Sans interruption :** c'est une approche pour rendre un système déterministe (non blocage des entrées) et il y a deux versions :
Version 1 : ce système perdre des informations sur les points de données (*en anglais datapoints*) de manière que le système ne peut pas déposer un datapoint lorsque la mémoire de la base de données est pleine.
Version 2 : ce système ne perdre pas les points de données et la probabilité que la mémoire de la base de données soit pleine est quasiment nulle.

Bien qu'il existe différentes catégories de systèmes différentiellement confidentiels, leur architecture reste commune à tous, avec notamment un modèle identique. Ces systèmes représentent un ensemble de trois entités qui s'exécutent de façon asynchrone ou synchrone et qui communiquent à travers l'échange de messages, la modélisation pour ce type de système se singularisent par trois caractéristiques :

1. **Politiques de confidentialité :** est un contrat qui requiert que le fournisseur de donnée (*en anglais DataProvider*) sache comment leurs informations confidentielles seront utilisées. Ainsi, le système doit codifier leurs pratiques des politiques de confidentialité accessibles au publique. Intuitivement, le détenteur des données, ou le système ne permettent jamais à l'analyste d'accéder directement aux données confidentielles.
2. **Mécanismes privés :** est un programme probabiliste qui prend trois paramètres en entrée : une bases de donnée, des bits aléatoires et un ensemble de requêtes, pour produire des réponses randomisées relativement précises aux requêtes.
3. **Gestion des données :** est une discipline de gestion des données pour déterminer la manière dont elles sont traitées et enregistrer, il y a deux types de données :
   (a) Données non- dynamiques auxquelles ils sont rarement modifiées.
   (b) Données dynamiques que changent de manière asynchrone au fil du temps.

2.4 Diagrammes de décision algébriques

Les diagrammes de décision binaire (BDD) [21] sont une représentation compacte des fonctions booléennes $f : \mathbb{B} \longrightarrow \mathcal{D}$, elles sont basées sur la représentation canonique de l'arbre binaire de la fonction sous forme de graphe orienté obtenu par pliage de nœuds internes représentant des sous-fonctions identiques (sous réserve d'un ordre des variables garantissant l'unicité de la représentation) et en utilisant 0 et 1 comme feuilles. Dans [22] [23], ils ont montré comment on peut généraliser des BDD pour représenter de manière cohérente et efficace des matrices en termes de diagrammes de décision binaires multi-terminaux (MTBDD), que sont également appelés diagrammes de décision algébriques (ADD). Un MTBDD est une structure graphique pour représenter des fonctions du type $f : \mathbb{B}^n \longrightarrow \mathcal{D}$, c'est-à-dire des fonctions d'un domaine booléen multidimensionnel à une plage arbitraire $\mathcal{D}$ (notez que pour un n fixe, l'image de la fonction f est toujours finie). Par exemple, $\mathcal{D}$ peut soyez les vrais nombres. Dans le cas spécial $\mathcal{D} = \mathbb{B}$, le MTBDD se réduit en réalité à un BDD, représentant une fonction booléenne. L'idée principale derrière la représentation MTBDD de fonctions à valeurs réelles est l'utilisation d'un graphe acyclique dirigé enraciné en tant que représentation plus compacte de l'arbre de décision binaire résultant de l'expansion de Shannon.

$$f(\mathsf{v}_1, ..., \mathsf{v}_n) = \mathsf{v}_1.f(1, \mathsf{v}_2, ..., \mathsf{v}_n) + (1 - \mathsf{v}_n)f(0, \mathsf{v}_2, ..., \mathsf{v}_n)$$

1où $(\mathsf{v}_1, ..., \mathsf{v}_n)$ sont des variables booléennes et $+$ et $\cdot$ désignent l'addition et la multiplication ordinaires. La figure 2.2 montre un simple MTBDD.

Figure 2.2 MTBDD M, représentant une fonction $f_{\mathsf{M}} : \{0, 1\}^4 \to [0, 1]$

2.5 Preuves d'induction

Une preuve par induction est comme une preuve ordinaire dans laquelle chaque étape doit être justifiée. Cependant, il utilise une astuce qui nous permet de prouver une déclaration sur un nombre arbitraire n en prouvant d'abord qu'il est vrai lorsque n est égale à 1, puis en supposant qu'il est vrai pour $n = k$ et en le montrant pour $n = k + 1$.

2.6 Induction discrètes

Dans cette sous-section nous présentons une structure de preuve inductive :

1. *Définir une propriété $P(n)$:* On prouve qu'une propriété $P(n)$ est vraie pour 0 et que si cette propriété vaut pour n, elle vaut également pour $n+1$, on commence notre preuve en précisant quelle propriété on va prouver par induction.
2. *Énoncer que la preuve est par induction :* On écrit que on va prouver, par induction, que c'est vrai pour tous les nombres qui nous intéressent, si on veut prouver que $P(n)$ est vrai pour tous les nombres naturels pairs supérieurs à sept, on écrit clairement pour donner au lecteur un avertissement sur la façon dont l'induction se déroulera.
3. *Énoncer et prouvez le cas de base :* On doit prouver $P(0)$ est vrai, parce que toutes les preuves inductives nécessitent une sorte de cas de base. on écrit explicitement ce que $P(0)$ indique, par exemple : «Nous prouverons $P(0)$, qui indique que ...».
4. *Énoncer et prouver l'étape inductive :* On montre que pour tout choix de k, si $P(k)$ est vrai, alors $P(k + 1)$ est vrai. Typiquement, on pourra cela en supposant $P(k)$ puis en prouvant $P(k + 1)$. Comme à l'étape (3), pour confirmer que on sait ce que on a supposé et ce que on va prouver.
5. *Conclure la preuve :* Généralement, on écrit quelque chose comme «la preuve par induction est complété » ou QED.

2.7 Induction réel

Le principe d'induction réel introduit dans [24] (voir aussi [25]) est équivalent à l'énoncé suivant :

On pense en termes de prédicats $P(n)$ et ses déclarations indexées par un sous-ensembles de $\mathbb{N}$: Soit $a < b$ des nombres réels. Nous définissons un sous-ensemble $S \subset [a, b]$ comme inductif si :

(RI1) $a \in S$

(RI2) Si $a \leq x < b$, alors $x \in S \Longrightarrow [x, y] \subset S$ pour certains $y > x$

(RI3) Si $a < x \leq b$ et $[a, x) \subset S$, alors $x \in S$

Dans un termes plus général, l'induction réel est pour montrer qu'une déclaration vaut pour tous les nombres réels dans un intervalle en «poussant de gauche à droite».

Cependant, les approches publiées conviennent parfaitement au problème de la confidentialité différentielle que nous traitons ici, car dans l'étape (RI2) de ces approches nécessite qu'il y ait $\Delta > 0$, de sorte que la validité de la propriété étudiée ou ε' implique sa validité pour toutes les valeurs $\varepsilon' \in (\varepsilon - \Delta, \varepsilon + \Delta)$. Ainsi, dans le chapitre 5, nous proposons et nous prouvons un principe similaire par l'induction réel, qui convient à nos besoins.

L'idée de ce principe est simple : si un ensemble de relations est satisfait par un ensemble de valeurs réelles de confidentialité ε pour une valeur $\varepsilon = \varepsilon_0$, et si la validité de ces relations pour toute valeur $\varepsilon_0 < \varepsilon < \varepsilon_n$ implique leur validité pour $\varepsilon + \Delta_0$, alors, ces relations seront satisfaites pour tout ε dans l'intervalle $[\varepsilon_0, \varepsilon_n]$. Intuitivement, sa logique est simple : les relations sont satisfaites pour ε_0, puis les sont satisfaites pour $\varepsilon_1 = \varepsilon_0 + \Delta_1$, puis pour $\varepsilon_2 = \varepsilon_0 + \Delta_2$, puis pour $\varepsilon_3 = \varepsilon_0 + \Delta_2$, et ainsi de suite. Ainsi, au cours de leur évolution, lorsque ε' dépasse de $[\varepsilon_0, \varepsilon_n]$, les valeurs ne s'écarteront jamais de la validité de ces relations.

2.8 Couplage probabiliste

Un couplage probabiliste modélise deux distributions avec une seule distribution conjointe. Généralement, les couplages ne sont pas uniques, et il y a différents technique représentent différentes façons de partager l'aléa entre deux distributions probabilistes.

Pour donner quelques exemples, nous introduisons d'abord quelques distributions standard.

Définition 6. *Soit Q un ensemble fini et non vide. La distribution uniforme sur Q, écrite Unif(Q), attribue la probabilité $\frac{1}{|Q|}$ à chaque élément. Nous écrivons "Lancer" pour la distribution uniforme sur les booléens, la distribution d'un tirage au sort équitable.*

Exemple : .

1. **Cas générale :** Soit une bijection $f : \mathcal{Q} \longrightarrow \mathcal{Q}$ donne un couplage (*Unif(Q)*,*Unif(Q)*)
   — *Couplage de bijection :*

$$\mu_f(q_1, q_2) \triangleq \begin{cases} 1/|Q| & : f(q_1) = q_2 \\ 0 & sinon \end{cases}$$

2. **Cas particulier :** On peut donner deux couplages distincts (*Lancer*, *Lancer*)
   — *Couplage d'identité :*

$$\mu_{id}(q_1, q_2) \triangleq \begin{cases} 1/2 & : q_1 = q_2 \\ 0 & sinon \end{cases}$$

   — *Couplage de négation :*

$$\mu_{\neg}(q_1, q_2) \triangleq \begin{cases} 1/2 & : \neg q_1 = q_2 \\ 0 & sinon \end{cases}$$

Dans ce projet, nous explorons une technique de preuve pour les propriétés relationnelles probabilistes [26], que garantissant la comparaison des exécutions de deux programmes randomisés. De telles propriétés sont courantes en informatique et en théorie des probabilités, à titre d'exemples :

— **Équivalence probabiliste :** Deux programmes probabilistes produisent mêmes distributions.

— **Domination stochastique :** Un programme probabiliste est plus probable qu'un autre pour produire de grandes sorties, par exemple, pour calculer une décision relative au portefeuille d'assurance, de telle sorte chaque portefeuille de sociétés peuvent être représentés par une distribution de probabilité.

2.8.1 Relation de levage :

Les relations de levage (en anglais *lifting*) ont été initialement introduites dans la recherche sur la bisimulation probabiliste, c'est une technique pour vérifier l'équivalence de deux systèmes de transition probabilistes.

Définition 7. *Soit μ_1, μ_2 des sous-distributions sur Q_1 et Q_2, et soit $\mathcal{R} \subseteq Q_1 \times Q_2$ une relation. Une sous-distribution μ sur les paires $Q_1 \times Q_2$ est un témoin du $\mathcal{R}$-lifting de (μ_1, μ_2) si :*

1. *μ est un couplage pour (μ_1, μ_2).*
2. *supp(μ) $\subseteq \mathcal{R}$*

2.8.2 Utilisation du couplage et levage pour la confidentialité diffrentielle :

En considérant la relation du *lifting* comme un type particulier de couplage, nous proposons des technique de vérification pour prouver des propriétés confidentialité en construisant des couplages, tout en tirant des idées du la relaxation *lifting* pour converger vers des conditions que garanties la confidentialité conditionnelles,

Nous développons ensuite dans le chapitre 5 une version approximative du couplage probabiliste, basée sur la relation de levage. On sait que l'existence d'une relation de levage implique une confidentialité différentielle.

2.9 Définitions de sécurité basées sur la trace :

Dans cette section, nous rappelons un certain nombre de définitions de sécurité classiques. Certains d'entre eux sont basés sur l'état et certains ont été définis à l'origine comme une propriété de traces, auquel cas nous donnons une définition correspondante dans notre modèle de système.

2.9.1 Relation de déroulement :

Les théorèmes de déroulement ont d'abord été proposés par Goguen et Meseguer [27]. Ils fournissent des conditions suffisantes de non-interférence qui peuvent être vérifiées car ils reposent sur les conditions locales de paires d'états. L'idée est que s'il existe une relation réflexive $\sim$ des états dans une machine à états d'entrée/sortie [2] $\mathcal{M}$ pour une politique de division des événements en H élevé (*High*) et L faible (*Low*) tels que pour tous les états s, t et action α .

1. *Cohérence de sortie* : si $s \sim_L t$ alors $obs(L, s) = obs(L, t)$
2. *Respecté localement* : si $\alpha \in A_H$ alors $s \sim_L next(s, \alpha)$
3. *Cohérence des étapes* : si $\alpha \in A_L$ alors $next(s, \alpha) \sim_L next(t, \alpha)$

où : $obs : D \times S \to O$ est une fonction représentant l'observation faite dans chaque état par chaque domaine que appartient au $D = \{H, L\}$. Nous avons présentons une légère modification de la définition habituelle, qui aurait une relation d'équivalence $\sim_L$ pour chaque agent L, satisfaisant un ensemble similaire de conditions pour chaque L. Pour la politique $L \leq H$ on peut prendre $\sim_H$ comme la relation universelle, qui satisfait automatiquement les conditions nécessaires.

Cette notion de relation de déroulement est un moyen de faciliter les preuves de non-interférence traditionnelle sur les systèmes déterministes.

Nous proposons dans le chapitre 5 d'une approche pour générer de tels contre-exemples en utilisant les résultats de l'analyse de déroulement. De manière similaire, Nous avons crée une extension que construit une relation entre les chemins et nous vérifions la cohérence des faibles sorties des états cible (cohérence des sorties) en termes des probabilités.

2. Les machines à états sont une méthode de modélisation de systèmes dont la sortie dépend de l'historique complet de leurs entrées, et pas seulement de l'entrée la plus récente.

2.10 Counter-example

L'avantage principal du model checking est qu'il fournit un contre-exemple lorsqu'une une erreur est révélée ou bien la propriété n'est pas satisfaite, il est souhaitable d'obtenir une sorte d'informations de diagnostic afin de tracer ce problème. Cette information est appelée un contre-exemple. Cela pourrait, par exemple, une exécution erronée du système., Edmund Clarke [28], souligne l'importance des contre-exemples :

"Il est impossible de surestimer l'importance de la fonction de contre-exemple. Les contre-exemples sont inestimables pour le débogage de systèmes complexes. Certaines personnes utilisent la vérification des modèles uniquement pour cette fonctionnalité" par Edmund Clarke.

Un contre-exemple montrant une exécution du modèle qui invalide la propriété sous une vérification, cela n'a pas été le cas pour la vérification de modèle probabiliste. Contrairement à d'autres techniques de vérification de modèle, les contre-exemples de ce paramètre ne sont pas fournis par un seul chemin d'exécution. Ce sont plutôt des ensembles d'exécutions du système satisfaisant une certaine propriété indésirable dont la masse de probabilité est supérieure à une borne donnée.

Dans ce projet, nous adoptons une approche différente pour trouver des bogues qui font que les modèles probabiliste violent la confidentialité différentielle.

CHAPITRE 3 REVUE DE LITTÉRATURE

Dans ce chapitre, nous donnons un aperçu des recherches étroitement liées aux travaux de ce mémoire : la conception d'un cadre d'analyse formel pour la vérification automatique des systèmes différentiellement confidentiels. Le chapitre comprend trois sections principales. Nous commençons par discuter les techniques de vérification de modèle probabiliste pour la sécurité dans la section 3.1, ensuite, dans la section 3.2, nous abordons le problème de la vérification de la confidentialité, en mettant un accent particulier sur la bisimulation et leur utilisation dans la vérification. Dans la section 3.3, nous passons en revue les techniques de vérification que sont similaires à ceux étudiés dans ce mémoire.

3.1 Model-checker probabilistie pour la sécurité

L'analyse formelle probabiliste s'est utile dans le domaine de la sécurité. La vérification formelle non probabiliste a été utilisée avec beaucoup de succès dans le passé pour vérifier l'exactitude (ou l'identification des failles) des protocoles de sécurité. Plus récemment, la vérification de modèle probabiliste a été utilisée pour examiner les protocoles de sécurité probabilistes, notamment ceux relatifs à l'anonymat (Crowds) [29], à la signature du contrat [30], à la non-répudiation [31], et le flux d'information quantitative [32]. Nous récapitulons dans le tableau 3.1 les différents travaux pour la vérification de la sécurité par PRISM model checker.

Tableau 3.1 Model-checker probabilistie pour la sécurité

Sécurité	Modèle	Propriété	Étude du cas	Référence
L'anonymat	DTMC	Innocence probable	Crowds protocol	Shmatikov [29]
Signature du contrat	DTMC	Équité	BGMR protocol	Even et al. [30]
Non-Repudiation	PTA	Connaissances sur les gains	N-R protocole	Markowitch et al [31]
QIF	DTMC	Mesure de fuite	DC-Nets et Crowds	Americo et al [32]
DP	MDP	Confidentialité différentielle	Système DP	Dans ce rapport 2020

1. **L'anonymat :** Shmatikov [29] a affirmé le rôle important de la probabilité dans les notions d'anonymat, il a introduit une méthode probabiliste pour analyser le système *Crowds* en vue d'une navigation Web anonyme, parce ce que le protocole *Crowds* constitue un cas intéressant dans lesquels le caractère aléatoire discret est exploité, il est un système pair-à-pair (*en anglais peer-to-peer*) de communication de groupe basé sur le routage aléatoire des messages entre les membres. Il s'agit aussi d'un protocole de sécurité bien connu qui vise à masquer l'identité des stations de navigation sur le Web, par exemple, en appliquant un routage aléatoire pour éviter les fuites d'informations.

   Il a appliqué l'outil PRISM pour modélise un protocole sous-jacent à *Crowds*, plus précisément, il a modélisé le comportement des membres du groupe et l'adversaire sous la forme d'une chaîne de Markov à temps discret, et formalisent les propriétés d'anonymat du système dans la logique temporelle probabiliste PCTL, en montrant, par exemple, comment l'anonymat probabiliste se dégrade lorsque la taille du groupe augmente. Toutefois, ce modèle probabiliste souffre d'un problème d'explosion d'état et n'a jusqu'à présent abouti que pour les systèmes de moins de 20 nœuds.

2. **Signature du contrat :** Une signature du contrat est une forme particulière d'échange équitable, dans lequel les parties échangent des engagements à un contrat(typiquement, les conditions de la transaction). Dans le cas des contrats en ligne, un engagement est souvent identifié par une signature numérique de la partie sur le contrat. Les principales propriétés qu'un protocole de signature de contrat devrait garantir que sont l'équité et la rapidité d'exécution. Par exemple, un protocole entre Alice et Bob est juste pour Alice si et seulement si pour toute situation où Bob a obtenu l'engagement de Alice, Alice peut obtenir l'engagement de Bob indépendamment des actions de Bob. Plus précisément, l'équité serait garantie par l'exécution simultanée des engagements par les parties de protocole.

   Norman et Shmatikov [30] ont analysé un protocole probabiliste de signature de contrat en utilisant l'outil PRISM, il s'agit de la première tentative d'analyse automatique d'un protocole de conversation probabiliste, ils ont étudié l'équité dans trois protocoles probabilistes différents.

   Ils ont pris une version simplifiée du protocole, ils ne tiennent compte que des messages signés de protocole dans leur analyse, et le comportement malhonnête d'un participant permet de retarder et de renvoyer ces messages sur les canaux de communication, ainsi que de contacter un tiers de confiance ou bien une partie honnête de communication chaque fois que cela est possible. Dans leur étude, ils ont constaté que le protocole dans sa description initiale ne respecte pas le délai. Compte tenu de cette description, le PRISM est utilisé pour déterminer la probabilité maximale avec laquelle un état injuste peut être atteint, et pour montrer que cette probabilité peut être arbitrairement petite. Ils ont montré également que l'ajout du

délai n'est pas une tâche triviale, lorsque le temps est mis en œuvre de manière efficace, une méthode simple permettant d'ajouter du délai se transforme en un protocole injuste, c'est-à-dire que l'équité peut être rompue avec une probabilité très élevée. Leur étude de cas montre que l'utilisation de PRISM pour analyser une simple version du protocole, avec un attaquant limité, et cette étude fournit des résultats intéressants.

3. **Non-répudation :** Dans [31], les auteurs ont fait une étude de cas pour analyser le protocole de non-répudiation de Markowitch et Roggeman pour le transfert d'informations, ils ont analysé ces protocoles par des automates temporisés probabilistes dans PRISM. La répudiation est définie comme le signe distinctif de l'une ou l'autre des parties ayant participé à tout ou partie du transfert d'informations de telle sorte l'expéditeur envoie des informations à une seconde partie, le destinataire. Par exemple, dans le commerce électronique, si l'information représente le transfert d'un service, alors la non-répudiation garantit que le client (le destinataire) ne peut pas refuser de recevoir le service en tant que motif de non-paiement.
4. **Flux d'Information quantitative :** Americo et al [32] ont exploré l'utilisation appropriée de la vérification de modèle PRISM pour calculer les limites d'un flux d'informations quantitatif pour différentes études de cas comme le problème dîner des cryptographes (*DC-Net*) et le protocole *Crowds*. Ils ont spécifié les propretés en PCTL pour calculer la capacité de canal, autrement dit ils ont utilisé l'aspect d'accessibilité de certains événements comme des propriétés en PCTL pour calculer les distributions a posteriori, à partir desquelles la capacité de canal est calculée. Ils ont modélisé le *DC-Net* et *Crowds* sous forme des grilles des nœuds et chaque nœud ne peut communiquer qu'avec ses voisins immédiats et ils ont analysons ensuite le *QIF* en utilisant l'état de l'art des métriques qui déjà existe dans littérature comme g-vulnérabilité.

   L'avantage de leur approche est qu'elle permet de modéliser les protocoles de manière directe et exprimer leur abstraction en tant que un canal que peut être facilement quantifié, ils ont calculé ce canal par un vérificateur de modèle probabiliste, plutôt que de dire une fuite spécifique ou une mesure de capacité, ils ont le rendons disponible pour toute utilisation appropriée.
5. **Analyse formelle de la confidentialité diffrentielle par PRISM :** Dans ce rapport, nous faisons un premier pas vers la vérification de la confidentialité diffrentielle en introduisant une approche basée sur la vérification de modèle dans laquelle nous utilisons le vérificateur de modèle probabiliste PRISM pour analyser le comportement des modèles differentiellement confidentiel en tant que Processus de décision Markovien (MDP). Cette approche nous a permis de quantifier de manière précise des fuites d'informations relatives à la confidentialité différentielle, dans un contexte avec une scénario d'interaction entre les composants d'un système concurrent.

3.2 Vérification formelle de la confidentielité diffrentielle

Divers algorithmes de vérification ont été proposés pour les modèles de la confidentialité différentielle pour les systèmes, mais sans implémentations ou d'outils. PRISM [15], sur lequel nous nous sommes fondés ce travail, fournit des modèles et des techniques de vérification pour un large éventail de propriétés des modèles probabilistes, y compris ceux de la logique PCTL*. Nous récapitulons dans le tableau 3.2 les différents travaux de recherche pour la vérification formelle de la confidentialité différentielle.

Tableau 3.2 Travaux de recherche pour la vérification formelle de la confidentialité différentielle

Approche	Modèle	Propriété	Outil	Référence
Approche	Modèle	propriété	✗	Reed et al [20]
Algorithme	PIOAs	Différentielle Non-interférence	✗	Tschantz et al [3]
Approche	Modèle	propriété	CertiPriv	Barth et al [33]
Approche	Modèle	propriété	✗	Lili xu et al [6]
Framework	pLTS	HML	✗	Yang et al [34]
Approche	LMCs	propriété	✗	Castiglioni et al [35]
Approche logique	DTMC	HyperPCTL	✗	Abraham et al [36]
Algorithme	DTMC	dpctl*	✗	Liu et al [37]
Automated verification	MDP	PCTL*	DP_PMC	Dans ce rapport 2020

1. **La distance rend les types plus solides : Calcul pour la confidentialité différentielle** : Reed et al [20] ont développé un système de types capable de vérifier la continuité de *Lipschitz*[1] des programmes fonctionnels, en tant que composant d'un nouveau langage pour la confidentialité différentielle. Ce système de types permettant de capturer la sensibilité d'une fonction, dans laquelle cette mesure peut amplifier les modifications apportées à ses entrées confidentielles, ces définitions étendre la métrique de confidentialité différentielle à l'aide de fonctions de distance génériques sur l'ensemble des secrets.
2. **Raisonnement probabiliste relationnel pour la confidentialité différentielle** :
   Barth et al [33] ont introduit une variante de logique relationnelle de Hoare pour raisonner formellement sur les algorithmes différentiellement confidentiels. Ils ont inclus des théorèmes relatifs à la composition séquentielle et parallèle des requêtes dans le style des énoncés de McSherry [8]. Le groupe de Gilles Barthe est probablement le plus avancé dans ce domaine, par exemple, leur système de type de raffinement relationnel *CertiPriv* a été utilisé

1. *Lipschitz* : est une forme forte de continuité uniforme pour les fonctions .

pour vérifier un certain nombre d'exemples dont l'analyse formelle des calculs différentiellement confidentiel. En particulier, ils fournissent les premières preuves d'exactitude pour les mécanismes de Laplace, gaussiens et exponentiels (que sont trois blocs de construction critiques pour des algorithmes différentiellement confidentiels) et pour certains algorithmes récents randomisés.

3. **Confidentialité différentielle dans les systèmes probabilistes :**

   Yang et al [34] ont proposé un cadre formel pour l'analyse de la confidentialité différentielle dans les modèles probabilistes, plus précisément, les modèles de transition probabiliste étiquetée (en anglais *probabilistic transition systems* (*pLTS*)), que admettant à la fois le comportement non déterministe et probabiliste, ils ont prouvé que la métrique de la borne inférieure (en anglais *metric infimum*) est une instance d'une métrique de la confidentialité différentielle qu'est prédéfinie sur les états, ils ont défini la confidentialité différentielle par la différence des distributions probabilistes des transitions étiquetées. Brièvement, ils ont caractérisé la métrique de la borne inférieure comme une métrique de la confidentialité différentielle via une nouvelle logique appelée (en anglais *privacy logic Hennessy–Milner* (*pHML*)), qui est une variante de la logique *Hennessy–Milner* traditionnelle. De telle sorte, ils ont proposé l'extension de chaque formule, qui est construite avec des paramètres de distance. Ils ont fourni aussi une méthode pour mesurer la distance dans la métrique avec la logique *pHML*.

4. **Hyperpctl : une logique temporelle pour les hyper propriétés probabilistes :**

   Abraham et al [36] ont introduit une approche logique pour exprimer et raisonner sur les hyperpropriétés probabilistes [2]. Ils ont formalisé la confidentialité différentielle à l'aide de cette logique que s'appelle HyperPCTL. Ils ont aussi développé un algorithme de vérification de modèle en utilisant une approche qui calcule de manière itérative la mesure exacte des chemins satisfaisant les sous-formules, mais cet algorithme fonctionne uniquement pour des systèmes spéciaux qui ont certaines propriétés structurelles et nécessitent beaucoup de temps et d'espace, c'est-à-dire le défi majeur dans la vérification des hyperpropriétés, est la complexité de calcul pour une vérification exhaustive croît au moins de manières exponentielles par le nombre d'alternances des quantificateurs de la formule d'entrée.

5. **Vérification de modèle de propriétés différentiellement privées :** Liu et al [37] ont développé une approche probabiliste pour vérifier la notion de la confidentialité différentielle. Cette approche repose sur la modélisation des programmes différentiellement confidentielle

2. Hyper propriétés : étendre les langages basés sur les traces en permettant une quantification explicite et simultanée sur le comportement temporel de plusieurs chemins d'exécution. Ces propriétés peuvent décrire des stratégies de sécurité importantes en matière de flux d'informations, telles que la non-interférence et la confidentialité différentielle, qui ne sont pas définissables par les langages basés sur les traces.

sous forme de chaînes des Markov. Ils ont introduit aussi une nouvelle logique *dpCTL** pour raisonner et exprimer les propriétés dans la confidentialité différentielle, ils ont utilisé les chaînes de Markov pour modéliser les mécanismes non interactifs (typiquement, les mécanismes qui n'interagissent pas avec son environnement). De plus, ils ont affirmé que la vérification de la confidentialité différentielle dans les modèles MDPs est indécidable.

6. **Vérification symbolique de la confidentialité diffrentielle :** Dans ce rapport, nous avons proposé une approche qui intègre les méthodes symboliques de la vérification des propriétés fonctionnelles des systèmes aux approches fondées sur la vérification d'équivalence de systèmes probabilistes. Notre méthode est entièrement automatique qui peut vérifier à la fois la confidentialité différentielle et détecter sa violation en générant des contre-exemples.

3.3 Les techniques de vérification de la confidentialité différentielle

Les approches les plus étroitement liées sont [3,6] ont étudié la confidentialité différentielle en utilisant une notion similaire à la bisimulation probabilste, mais ces approches ne prennent en compte que la (ε,0)-confidentialité différentielle, c'est-à-dire ils considèrent le rapport des probabilités et n'examinent pas la manière dont ils pourraient être calculés. Cependant, ces techniques n'a pas le pouvoir expressif d'offrir un coût de confidentialité limité pour des algorithmes sophistiqués préservant la confidentialité ou bien ne caractérisent pas complètement la bisimilarité probabiliste.

3.3.1 Non-interférence différentielle dans les systèmes interactifs :

Tschantz et al [3] ont donné des expériences détaillées de la confidentialité différentielle dans les systèmes réels et leurs travaux présentent un moyen de modéliser des mécanismes différentiellement privés dans les systèmes interactifs définis par des automates d'I/O probabilistes et de développer des techniques de preuve basée sur la bisimulation pour raisonner sur la confidentialité différentielle de tels systèmes, ils considèrent un système similaire à PINQ et l'utilisent pour démontrer leurs techniques de preuve, ils ont formalisent la confidentialité différentielle dans le cadre des fuites d'informations et ont proposé une méthode de vérification basée sur la preuve de l'existence d'une famille de relations de déroulement (en anglais *Unwinding*) entre les états qui permettent de suivre les fuites de confidentialité qui ne dépassent pas un budget de fuites données. Une grande partie de ce travail est centré sur la vérification de la confidentialité, compte tenu de l'accès à une description de l'algorithme. Ces outils sont destinés à la vérification automatique de la confidentialité du code source.

Cette approche est la deuxième tentative de vérification formelle de la confidentialité différentielle après le travail de [20]. Ils se concentrent sur l'utilisation de mécanismes différentiellement

confidentiel dans les systèmes interactifs en adaptant les travaux antérieurs sur la confidentialité différentielle de fonctions. En effet, ils ont implémenté le mécanisme par un sous-programme modélisé par un automate et un modèle d'implémentation en remplaçant chaque transition pour une fonction par sa chaîne de Markov absorbante correspondante.

Cette définition de la confidentialité différentielle à des systèmes interactifs modélisés par une sorte d'automate probabiliste et elle s'appelle la notion de non-interférence différentielle. Dans leurs paramètres, les actions sont séparées en actions d'entrée et de sortie, les premières étant elles sont séparées en actions de requête et de données, tandis que celles-ci sont divisées en fonction de leur possibilité d'être observées par l'examinateur de données. Un automate probabiliste donne la non-interférence différentielle si le rapport des probabilités des chemins adjacents produisent le même résultat observable est approximativement égal à $exp(\varepsilon)$, où deux chemins sont adjacents si leurs étiquettes d'entrée ne diffèrent que par une étiquette de données.
Ce modèle formel de non-interférence différentielle présente des similitudes avec la non-interférence probabiliste, et basé sur les systèmes de transition probabilistes. Ils développent des méthodes de preuve similaires à la bisimulation et des idées similaires pour reformuler la preuve principale de la confidentialité dans un système. Ces techniques de preuve permettant de raisonner sur la confidentialité différentielle interactive avec des enregistrements (ou points de données) saisissent au fil du temps dans une base de données mutable.
Dans leur système, les requêtes sont fournies par l'analyste qu'est modélisé par une séquence de toutes les requêtes possibles (pour toutes les séquences d'entrée possibles). L'inconvénient de ce modèle est qu'il ne capture pas la stratégie de l'utilisateur pour les systèmes probabilistes, parce qu'il est connu dans la littérature non-interférence que la modélisation de l'utilisateur utilisant un non-déterminisme plutôt qu'une stratégie peut masquer la présence de fuites d'informations. L'aspect de l'analyste de n'est pas capturé dans leur modèle et il est non adaptatif en tant que partie explicite de la modélisation, ils modélisent la base de données par une mémoire limitée, cela entraîne un grossissement inattendu du coût de la confidentialité des calculs, car l'ajout d'un point de données dans un flux d'entrée entraînera une modification de la mémoire totale de deux-points de données (l'enregistrement lui-même et l'enregistrement qu'il déplace), et cette approche est limitée à la confidentialité différentielle pure.
Finallement, la non-interférence différentielle repose sur une notion de *lifting*, et leur méthode de vérification basée sur le déroulement *(Unwinding)*, qu'est une preuve de l'existence d'une famille de bijections indexées entre états, où le paramètre des états de départ, représentant le budget de la confidentialité que détermine le niveau de confidentialité différentielle du système, et qui diminue avec le temps en soustrayant la différence absolue des probabilités à chaque étape de la bisimulation. Une fois que la balance atteinte zéro, les processus doivent se comporter exactement de la même manière.

3.3.2 Bisimulation métrique dans les systèmes concurrent :

Lili xu et al [6] ont proposé des techniques pour assurer la confidentialité différentielle pour les systèmes concurrent en sont utilisées un calcul de processus probabiliste. Ils ont développé une méthode de composition pour prouver la confidentialité différentielle dans un environnement distribué. Cette approche compositionnelle consiste à diviser le système en plusieurs parties, d'analyser séparément le degré de confidentialité et combiner les résultats pour obtenir le degré global de la confidentialité. Techniquement, cette méthode est basée sur la décomposition d'un processus en processus plus simples pour faire le calcul du niveau de la confidentialité de ceux-ci et effectuer leur combinaison pour obtenir le niveau de confidentialité du programme.
La notion de métrique de programme est en train de devenir un outil fondamental dans l'étude de la confidentialité différentielle. L'idée est consiste à prendre en compte les métriques du programme plutôt que les équivalences de programmes. De cette façon, chaque paire de programmes est doublée et se trouve à une certaine distance numérique plutôt qu'elle est simplement équivalente (ou non). L'équivalence de programme correspond à une métrique de programme ou à une distance de programme, c'est-à-dire à une fonction de type métrique définissant une distance entre les programmes en fonction de leur comportement.

En bref, ils ont examiné la confidentialité différentielle pour les systèmes concurrents dans un modèle d'algèbre de processus probabiliste (en anglais *Probabilistic Process Calculus* (PPC)) et d'automates probabilistes. Ils s'intéressaient aux probabilités des traces adjacentes finies dans les automates probabilistes. De plus, ils ont proposé trois métriques différentes pour vérifier leur confidentialité différentielle et ils ont comparé ces métriques, prouvant que les deux dernières métriques sont effectivement plus permissives que la première, mais ils sont comparables. Ils ont été repris la notion d'amortissement pour formuler une notion approximative d'équivalence comportementale probabiliste : la bisimulation probabiliste amortie, où est un nombre réel positif décrivant un degré de similitude entre les processus.

CHAPITRE 4 ANALYSE FORMELLE DE LA CONFIDENTIALITÉ DIFFÉRENTIELLE AVEC PRISM

4.1 Introduction

Dans ce chapitre, nous proposons une approche pour l'analyse formelle de la confidentialité différentielle dans les systèmes concurrents probabilistes caractérisés par la présence de la randomisation et de la concurrence en utilisant la vérification quantitative via un model-checker probabiliste. Il est bien connu que la résolution du non-déterminisme, due aux possibles interconnexions et interactions des composants parallèles comme le composant de l'ordonnanceur et les programmes différentiellement confidentiels qui entrainent une violation de la vie privée et peut provoquer des fuites d'informations. Un moyen de résoudre ce problème consiste à explorer le modèle computationnel de système afin de mieux comprendre l'interaction entre les mécanismes différentiellement confidentiels et les composants parallèles en énonçant la garantie de flux d'informations de bout en bout du système entier.

Nous adoptons une approche de la modélisation à l'aide d'un formalisme issu du réactif module qui permettant de représente les composants synchrones et asynchrones, et de considérer ses comportements probabilistes dans un cadre uniforme prenant en charge la conception hiérarchique et la vérification compositionnelles. Cette démarche permet de construire des modèles de confidentialité dont la sémantique opérationnelle s'exprime à l'aide de processus de décision markovien (MDPs).

Nous démontrons dans un ensemble d'expériences des propriétés quantitatives, telles que les paramètres de la confidentialité : les fuites des données, les probabilités des sorties, l'utilité des mécanismes, les différentes combinaisons des distributions probabilistes des sorties et certains compromis entre ces paramètres. Nous mettons en œuvre notre approche, en nous appuyant sur le vérificateur de modèle PRISM, et nous appliquons à des études de cas.

4.2 Modèle conceptuel des systèmes différentiellement confidentiels

Nous considérons trois types d'entités pour un système différentiellement confidentiel concurrent : les fournisseurs de données, un détenteur de données et les analystes. Un fournisseur de données est responsable de l'envoi des données privé au détenteur de données, et ce dernier il est responsable de la protection des données privées qu'elle détient, et utilise le mécanisme de confidentialité comme un élément constitutif de système pour fournir des réponses randomisées aux analystes. En supposant qu'un ou plusieurs des analystes puissent être des attaquants, plusieurs scénarios possibles peuvent être envisagés. L'un de ces scénarios, l'attaquant peut essayer plusieurs séquences des requêtes pour divulguer les informations confidentielles.

Nous avons besoin du non-déterminisme pour modéliser le parallélisme de ses sous-systèmes et nous avons également besoin de choix probabilistes pour modéliser les réponses possibles d'un mécanisme de la confidentialité. Nous utilisons le processus de décision markovien (MDP) pour modéliser la sémantique opérationnelle du système différentiellement confidentiel. Nous utilisons également les ordonnanceurs probabilistes sans mémoire (*en anglais memoryless probabilistic schedulers*) pour représenter la politique d'ordonnancement de ces systèmes. Nous fournissons aux modèles MDPs un type particulier de notion de la confidentialité qui garantit que le système continuera à produire des sorties randomisées.

Fournisseur de données (en anglais Data Provider) :

Le fournisseur de données ne garantit pas la confidentialité et il est responsable de l'envoi des bases de données ou des enregistrements qui sont appellent des points de données (*en anglais datapoints*) au détenteur de données. On considère deux types deux modèles :

1. Un modèle de base : ce modèle n'a aucune hypothèse sur l'indépendance entre les points de données.
2. Un modèle avancé : ce modèle a besoin des conditions que limitant la caractérisation à des points de données avec une probabilité non nulle.

Détenteur de données (en anglais Data Holder) :

Le détenteur de données est un système différentiellement confidentiel (Sys_{DP}) qui contrôle le système dans son ensemble et qui assure la confidentialité des informations d'un individu qui est bénéficié par le fournisseur des données. On modélise Sys_{DP} par ses trois fonctions principales qui sont :

1. Répondre aux requêtes privées ou publiques d'une manière non déterministe.

2. Mécanismes privés pour assurer la confidentialité différentielle .
3. L'envoi des réponses randomisé à l'examinateur des données .

Analyste (en anglais Data Examiner) :

1. Un modèle de base : l'analyste ne dispose pas des informations précises sur la présence ou l'absence d'un point de données dans la base de donnés, autrement dit, l'analyste a une connaissance nulle et il n'y a aucune hypothèse sur la puissance de l'adversaire. On doit assumer le cas le plus défavorable.
2. Un modèle avancé : en modélisant l'analyste avec une seule variable locale (***cost***) pour déterminer le «cout de confidentialité» d'une requête, c'est-à-dire la valeur qui doit être soustraite du budget de confidentialité pour que le résultat puisse être renvoyé à l'analyste. L'analyste a pour but de calculer la valeur a diminué du budget de la confidentialité.

Modèle de système :

$$M = DataProvider \| DataHolder \| DataExaminer$$

On peut l'écrire plus simplement par :

$$M = M_1 \| M_2 \| M_3$$

Et plus généralement par la structure que capturer la notion de la confidentialité différentielle dans le contexte de concurrence :

$$((M_{11} \| ... \| M_{1m})) \|_H \; M_2 \|_L \; (M_{31} \| ... \| M_{3n})$$

où les m $DataProviders$ $M_{11}||...||M_{1m}$ sont indépendants et communique avec la structure de données via des communications de haut niveau H ($High$), et n $DataExaminers$ envoient des requêtes au $DataHolder$ via des communications de bas niveau L (Low). Supposons que les données secrètes soient stockées dans une structure de données et que l'attaquant peut lire seulement les réponses d'un mécanisme confidentiel, mais ne peut pas accéder aux données secrètes (c'est-à-dire que le contrôle d'accès et la protection de la mémoire fonctionnent correctement). De plus, supposons que l'attaquant puisse essayer de deviner une valeur secrète (en un seul essai ou en plusieurs essais), après avoir envoyé une requête au programme et observé la séquence des réponses.

Le MDP du modèle conceptuel :

Nous présentent maintenant comment modéliser un système destiné à fournir une confidentialité différentielle par les processus de décision markovien (MDP).
Le choix de MDP est motivé par trois raisons : il peut capturer le non-déterminisme et l'incertitude dans un système donné. Ces deux premières raisons sont essentielles dans les systèmes distribués différentiellement confidentiels. Le nombre des entrées dans la base de données peut augmenter, rester identique ou diminuer à chaque instant et l'interaction synchrone et asynchrone avec l'analyste est imprévisible, cela donne lieu à un non-déterminisme. En outre, à tout moment, le système renvoi des réponses randomisées qui sont calculées par un mécanisme de la confidentialité cela nécessite une modélisation de l'incertitude. La troisième raison est qu'on doit calculer la probabilité maximale et minimale pour considérer le meilleur et le pire cas de la confidentialité pour délimiter les requêtes avec la pire erreur, compte tenu les distributions probabiliste. Autrement dit, nous considérons que la confidentialité différentielle limite la perte de confidentialité dans le cas le plus défavorable sur tous les résultats possibles.

Les MDPs des systèmes différentiellement confidentiels sont spécifiées en utilisant plusieurs types d'entités :

1. **États :** Une configuration d'un système différentiellement privé est définie comme un quadruple $S = (X, Q, R, P)$ où :
   - $X = \{x_1, ..., x_n\}$ sont les points d'entrées d'une base de données ;
   - $Q = \{q_1, ..., q_m\}$ sont les résultats privés des requêtes aux bases de données ;
   - $R = \{r_1, ..., r_m\}$ sont les réponses randomisées aux requêtes ;
   - P : sont les paramètres de la confidentialité .

2. **Actions :** Pour définir les actions en associant un signateur qui est un tuple (A, dom, f_D), tel que A sont les actions, $dom = \{H, L\}$ sont des domaines et f_D une fonction ; $f_D : A \to dom$. Nous définissons $A_{dom} = \{\alpha \in A \mid f_D(\alpha) = dom\}$ et nous supposons que :
   - $A_H = \{in_1, ..., in_n, out_1, ..., out_m | n, m \in \mathbb{N}\}$, représentant réspectivement les entrées et les sorties fournies par le fournisseur de donneés, in correspondant aux données transféré et out correspondant ses réponses secrètes .
   - $A_L = \{in_1, ..., in_n, out_1, ..., out_m | n, m \in \mathbb{N}\}$, représentant réspectivement les entrées et les sorties fournies par l'analyste, in_i correspondant à la requête i et l'action out_i correspondant à la réponse randomisée.

3. **Transitions :** Chaque transition dans le modèle est associée à une étiquette d'action, selon le type de modèle que l'on souhaite à analyser et à vérifier, il y a de différents types de transitions se produiront. Nous avons principalement quatre types de transitions :
   — *Entrée ;* $\delta(s, in_h)(s')$: insertion d'un point de données dans un multiset X.
   — *Commande d'abandon ;* $\delta(s, out_H)(s')$: suppression ou l'interruption de l'envoi d'un point de données.
   — *Fonction non-déterministe ;* $\delta(s, in_L)(s')$: c'est une transition qui représente le calcul interne d'une fonction sur un multiset X.
   — *Résultat de sortie ;* $\delta(s, out_L)(s')$: c'est une transition probabiliste qui représente une réponse randomisée.
4. **Probabilités :** les probabilités sont déterminées selon le type des transitions :
   — *Résultat de sortie ;* ces transitions sont étendues pour un certain type d'états observables :

$$\delta(s_j, out_L) = [t_0 \to \mathbf{P}[t_0|s_j], ..., t_n \to \mathbf{P}[t_n|s_j]]$$
avec $i \in [0, n]$ et $n \in \mathbb{N}$ nombre des observables pour le secret s_j.

   Plus précisément, l'état s_j réprésente le secret,$T = \{t_0, ..., ..., t_n\}$ sont les observables. A titre d'exemple, ces transitions représentant l'appel au mécanisme confidentiel et $\mathbf{P}[.][.]$ une probabilité proportionnelle à la valeur ε.
   — *Entrée* , *Commande d'abandon* et *la Fonction non déterministe ;* l'espace probabiliste définie selon le contexte de l'application à modéliser.
5. **Récompenses :** Les récompenses sont utilisées pour raisonner sur un éventail plus large de mesures quantitatives relatives au comportement du modèle de la confidentialité telle que la perte cumulative de la vie privée d'un point de données et l'évaluation de l'utilité de la confidentialité différentielle. Par exemple, nous pouvons étendre un MDP avec des récompenses d'action pour représenter le nombre des résultats imprécis d'une requête jusqu'à une certaine observation donnée.
6. **Étiquettes :** Les modèles de la confidentialité doivent également contenir des étiquettes pour identifier les états que représente un des ensembles de propriétés de système. Les états de modèle peuvent être divisés en états *secret* ou *observables*. Le premier modélise les aspects non déterministes d'un système, d'une part pour le choix d'un point de données pour participiez à une base de données, et d'autre part pour répondre à des requêtes selon d'un politique. Par exemple, un système asynchrone peut à la fois répondre à chaque requête de manière différentiellement confidentielle et ne pas l'être différemment en modifiant l'ordre de réponse aux

requêtes ou en corrélant les choix faits par le ordonnenceur. Les étiquettes sont des moyens d'identifier des ensembles d'états $Secret$ qui présentant un intérêt particulier, par exemple le voisinage des bases de données qui ne diffère que par un seul point de données et les états *observables* codent les réponses randomisées des mécanismes confidentiels.

L'ensembles des propositions atomiques :

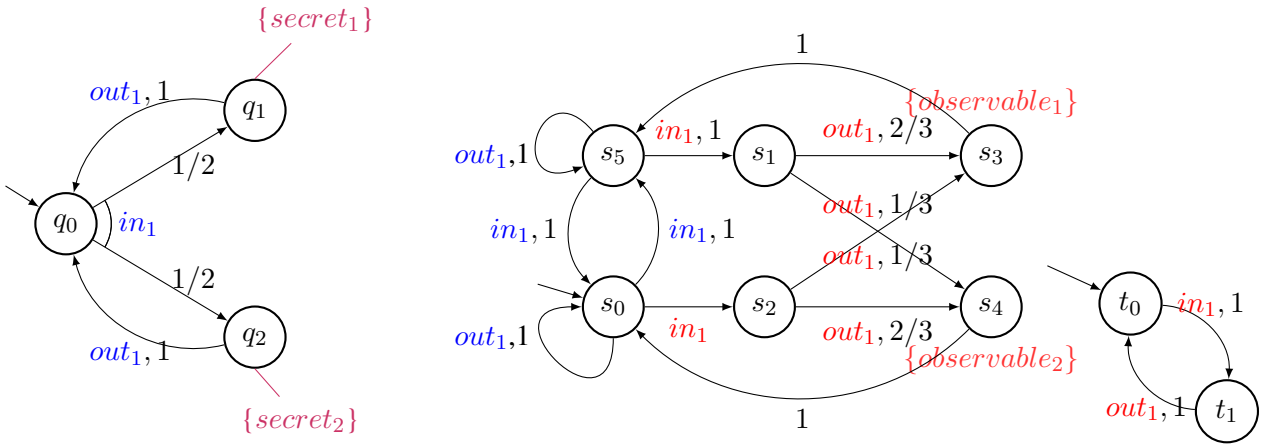
$$AP = \{secret_1, ..., secret_m, observable_1, ..., observable_n\}$$

Notre objectif sera de comparer les états des chaînes de Markov étiquetées du point de vue de la confidentialité différentielle. Notez que deux états (s_1, s_2) peuvent être considérés comme indistinguables si $\mathbf{P}_2(s_1, t) = \mathbf{P}_2(s_2, t)$. S'ils ne sont pas indistinguables, la différence entre eux peut être quantifiée en utilisant la distance variationnelle. Nous allons définir la notion naturelle de l'adjacente par un niveau de distinction entre les secrets et nous allons appliquer le même principe de la confidentialité différentielle, c'est-à-dire que plus le niveau de distinction entre (s_1, s_2) est petit, plus les distributions de probabilité $(\mathbf{P}_2(s_1, t), \mathbf{P}_2(s_2, t))$ doivent être similaires et indistinguables pour obtenir une notion significative de la vie privée.

Pour arriver à une définition formelle de la confidentialité différentielle des systèmes basée sur la similitude de leurs comportements, nous devons d'abord avoir une méthode d'analyse pour les systèmes probabiliste. Notre méthode est basée à la fois sur des propriétés qui sont basées sur des actions (*en anglais action-based properties*) et basées sur des états (*en anglais states-based properties*), autrement dit basé sur des traces de chemin qui sont des variables aléatoires sur des séquences d'actions pour explorer la structure de la similitude des comportements des programmes différentiellement confidentiels et raisonner à leur sujet, en utilisant les propositions atomiques attribuées à chaque état par la fonction d'étiquetage L. Dans notre frameworke, la confidentialité ne peut être analysée qu'entre deux états de départ de deux chaînes de Markov, mais il est également raisonnable de permettre une entrée sous la forme d'une trace ou d'une séquence d'actions, la sortie serait également une trace. Ici, le choix des étiquettes (à un état spécifique : $secret$ ou $observable$) correspondrait aux décisions prises par l'utilisateur. Ce paramètre prendrait en charge un large éventail de scénarios qui pourraient être modélisés et vérifiés comme différentiellement confidentiel.

Exemple : Confidentialité différentielle compositionnelle

Nous décrivons dans cette partie, une modélisation d'un système différentiellement confidentiel par un MDP composé à partir des modèles de ses sous-systèmes. Chacun de ces sous-systèmes est alors modélisé par un $\mathcal{M}_i$, et le modèle du système global correspond à la composition parallèle synchrone de ces différents MDPs. La figure 4.1 réprésente un système globale. $\mathcal{M} = \mathcal{M}_1||\mathcal{M}_2||\mathcal{M}_3$.



(a) $DataProvider$ (b) $DataHolder$ (c) $DataExaminer$

Figure 4.1 Modèle MDP d'un système différentiellement confidentiel

Pour assurer la confidentialité différentielle de bout en bout dans un système construit à partir des composants, il est important de s'assurer que la composition des composants est elle-même diffrentiellemnt confidentielle, en assurant une absence contrôlée [1] de flux d'informations des entités de haut niveau aux entités de bas niveau, alors nous avons deux type d'actions $A \triangleq A_H \cup A_L$:

— $in_1 \in A_H$: L'ajout d'un point de données.

— $out_1 \in A_H$: La suppression d'un point de données.

— $in_1 \in A_L$: Requête appliquer à une base de données.

— $out_1 \in A_L$: Réponse randomisée à une requête.

Nous avons quatre propositions atomiques : $AP = \{secret_1, secret_2, observable_1, observable_2\}$ avec le fonction d'étiquetage :

$$L(q_1) = \{secret_1\}, \quad L(q_2) = \{secret_2\}, \quad L(s_3) = \{observable_1\}, \quad L(s_4) = \{observable_2\}.$$

1. Les probabilités de traces d'une réponse observables par les utilisateurs de bas niveau (*Data examiner*) soient indistinguable pour n'importe quelle paire d'entrées de haut niveau (points de données dans notre contexte)

Techniquement, les composants du système fonctionnent comme suit :

1. ***Fournisseur de données*** **:** son espace d'états est défini comme suit : $S_{\mathcal{M}_1} = \{q_0, q_1, q_2\}$, où l'état q_1 indique qu'un point de données a été envoyé et n'a pas été chargé dans la base de données, l'état q_2 indique qu'un point de données a été envoyé et chargé dans la base de données.
2. ***Détenteur de données*** **:** est un sous-système qui contrôle la confidentialité et il est synchronisé avec d'autre composants de système entier, son espace d'états est défini par :
   $S_{\mathcal{M}_2} = \{s_0, s_1, s_2, s_3, s_4, s_5\}$. Ce sous-système peut être dans l'état initial $\bar{s} = s_0$ où la base de données vide, et dans les états s_1 et s_2 que représentent les résultats posssible d'une requête in_1 appliqué au base de données, et dans les états s_2 et s_3 que représentent les réponses randomisées aux requêtes .
   $\delta(s_0, in_1)(s_5)$: ajouter un point de données.
   $\delta(s_5, out_1)(s_0)$: supprimer un point de données.
3. ***Analyste*** **:** Il a deux tâches principales :
   $\delta(t_0, in_1)(t_1)$: envoyer une requête au système ;
   $\delta(t_1, out_1)(t_0)$: recevoir une réponse randomisée.

ε-confidentialité différentielle :

Nous illustrons la notion de la confidenitalité diffrentielle en MDP. À chaque étape, il faut décider quel composant prendra la transition suivante. Il y a donc un certain nombre de choix d'actions non déterministes à prendre ensuite dans chaque état. Les actions α_i représentent l'exécution du premier composant (*Data Provider*) tandis que les actions β_i représentent l'exécution du second composant (*Data Examine*).

Ces choix non-déterministes sont faits par un ordonnanceur $\sigma \in Adv$ que impose aux trois composants du système un ordre d'exécution. Par exemple dans l'état initial , il tient que $\sigma((s_0, q_0, t_0) \longrightarrow ((s_5, q_1, t_0))) = \alpha_1$. Maintenant, nous considérons les deux chemins suivants $\pi_1 = (s_0, q_0, t_0) \xrightarrow{\alpha_1} (s_5, q_1, t_0) \xrightarrow{\beta_1} (s_1, q_1, t_1) \xrightarrow{\beta_2} (s_3, q_1, t_0)$, $\pi_2 = (s_0, q_0, t_0) \xrightarrow{\alpha_2} (s_5, q_1, t_0) \xrightarrow{\alpha_3} (s_5, q_0, t_0) \xrightarrow{\beta_3} (s_0, q_2, t_1) \xrightarrow{\beta_4} (s_3, q_2, t_0)$. Nous calculons facilement que les probabilités de deux traces ne diffèrent que par un seul variable et qu'ils ont le même observable :
$\frac{Pr^{\sigma}(\pi_1)}{Pr^{\sigma}(\pi_2)} = \frac{\frac{1}{6}}{\frac{1}{3}} = \varepsilon = 0.5$, alors les deux traces sont différentiellement confidentiels par rapport un paramètre privé ε =0.5. Dans la partie suivante, nous allons faire une analyse exhaustive de la confidentialité différentielle.

4.3 Confidentialité différentielle : modèle et sémantique

Dans cette section, nous formalisons la notion de la confidentialité différentielle dans les modèles probabilistes DTMCs et MDPs. Cette notion requiert que les distributions de probabilité sur les traces que sont exécutés sur des entrées adjacentes soient approximativement indistinguables.

4.3.1 Espace de données

L'espace de données d'un modèle est induit par un ensemble fini de variables Var. Chaque variable $v \in Var$ d'un modèle a un type $Type(v)$. Le type $Type(Var)$ est défini formellement comme l'ensemble des valeurs que v peut assumer. Nous limitons explicitement les types de variables autorisés dans les modèles. En particulier, pour la vérification des modèles nous limitons aux types finis et nous admettons également les types mathématiques appropriés tels que $\mathbb{B}$ et $\mathbb{N}$.

Dans le contexte de la confidentialité différentielle, on considère les programmes que fonctionnent sur un flux des données, et on défini des variables aléatoires discrètes pour représenter les points de données : $X = \{x_1^{(t)}, x_2^{(t)}, ..., x_n^{(t)}\}$ telle que $t, n \in \mathbb{N}$ et $X \subset Var$.
Tout au long du fonctionnement de système, nous gardons la trace de plusieurs valeurs des requêtes qui sont dénotées par l'ensemble des variables Q que contiennent des informations relatives à la requête q dans le jeu de requêtes Q, tel que $q \in Q$. Nous donnons pour ces requêtes l'expressivité de l'interaction d'une fonction privée avec les points de données : une requête demandait à chaque point de données s'ils satisfont ou non à un prédicat donné. L'ensemble des variables qui contient toutes les valeurs possibles des requêtes est donné par : $Q = \{q_1^{(t)}, q_2^{(t)}, ..., q_m^{(t)}\}$ telle que $t, m \in \mathbb{N}$ et $Q \subset Var$

$$q_j = \sum_{i=0}^{n} expr_j(x_i) \;\; avec \;\; j \in [0, m]$$

q_j la variable de la requête de comptage j et $expr_j$ sa expression booléen sur les variables x_i.

4.3.2 Mécanisme de confidentialité

Le modèle mathématique que nous utilisons pour modéliser les mécanismes de confidentialité est conçu pour modéliser des programmes probabilistes séquentiels et non récursifs. Un mécanisme de confidentialité est une chaîne de Markov à temps discret. Nous exigeons que le mécanisme soit correct et l'aspect probabiliste caractérise les réponses aléatoires aux analystes dont le secret n'est pas connu. (Voir la figure 4.2 ci-dessous).

$$
\begin{array}{rc}
 & \text{Observables} \\
 & \begin{array}{ccc} t_1 & t_i & t_n \end{array} \\
\begin{array}{r} \\ s_1 \\ \text{Secrets } s_i \\ s_n \end{array} & \left[\begin{array}{ccc}
\mathbf{P}_{\mathcal{K}}[t_1|s_1] & \mathbf{P}_{\mathcal{K}}[t_i|s_1] & \mathbf{P}_{\mathcal{K}}[t_n|s_1] \\
\mathbf{P}_{\mathcal{K}}[t_1|s_i] & \mathbf{P}_{\mathcal{K}}[t_i|s_i] & \mathbf{P}_{\mathcal{K}}[t_n|s_i] \\
\mathbf{P}_{\mathcal{K}}[t_1|s_n] & \mathbf{P}_{\mathcal{K}}[t_i|s_n] & \mathbf{P}_{\mathcal{K}}[t_n|s_n]
\end{array}\right]
\end{array}
$$

Figure 4.2 Mécanisme de confidentialité

L'ensemble des matrices des transitions probabilistes d'une chaîne de Markov est noté par $\mathcal{M}_{n,n}([0,1])$ avec $n = |S|$. Nous définissons la matrice de probabilité d'un mécanisme de confidentialité comme une sous-matrice [2] par :

$$\mathbf{P}_{\mathcal{K}} : Sec \times Obs \longrightarrow [0,1], \text{ telle que } Sec, Obs \subseteq S \text{ et } \mathbf{P}_{\mathcal{K}} \in \mathcal{M}_{n,n}([0,1])$$

Exemple : Préférences des musiques

On illustre cette notion du mécanisme de confidentialité par l'exemple suivant : l'application des préférences des musiques privées. Une base de données de n utilisateurs et k chansons $X = \{x_1, x_2, ..., x_k\}$, chaque utilisateur faire 👍/👎 pour chaque chanson ($x_i = x_i + 1$ ou $x_i = x_i + 0$) et l'objectif de cette application est de choisir une chanson que beaucoup aiment. L'application utilise un mécanisme de confidentialité différentielle pour protéger la vie privée des utilisateurs :
une réponse j avec une probabilité : $\mathbf{P}_{\mathcal{K}}[j] \propto e^{\varepsilon.Card(j)}$
où : $Card((j) =$ nombre des utilisateurs qui aiment la chanson j. et
la probabilité pour observable t_j sachant s_i est égale $\mathbf{P}_{\mathcal{K}}[t_j|s_i] = \dfrac{e^{\varepsilon.Card(j)}}{\sum_i e^{\varepsilon.Card(i)}}$.

Le *Secret* : représente le nombre exact des utilisateurs qui aiment une chanson .
Le *Observable* : représente le nombre randomisé des utilisateurs qui aiment une chanson .

2. Une sous-matrice d'une matrice donnée est une matrice située dans des sous-ensembles spécifiés des lignes et des colonnes d'une matrice donnée.

4.3.3 Raisonnement probabiliste de la confidentialité différentielle .

Les programmes différentiellement confidentiels sont naturellement non déterministes et il est donc difficile d'assurer la notion de la confidentialité différentielle à ces programmes parce que leur sémantique ne spécifie pas l'ordre d'exécution des requêtes ou des programmes simultanés. Bien que le non-déterminisme permette une variété d'implémentations d'ordonnanceur de programme, il conduit également à des attaques via des flux information, des canaux cachés difficiles à contrôler, dans lesquelles des informations sont perdues par la résolution de choix non déterministes d'un d'ordonnanceurs. Intuitivement, la confidentialité différentielle indique qu'un programme est différentiellement confidentiel lorsque ses traces observables publiquement sont indépendantes de ses données confidentielles qui ne sont diffèrent par un seul point de données et de son ordonnanceur, par conséquent, cette section indique que les définitions que nous utilisons dans notre méthode d'analyse .

Notre principe du raisonnement guide naturellement la preuve de la confidentialité différentielle : lors d'une entrée secrète, par exemple, nous pouvons classer les exécutions en fonction du chemin qu'elles empruntent, puis nous considérons chaque comportement séparément pour un observable. De cette façon, nous pouvons raisonner sur un programme complexe en nous concentrant sur des cas plus simples, mais cette image nette est considérablement plus compliquée. Une seule exécution comprend désormais plusieurs traces, chacune avec sa propre probabilité. Les relations entre les probabilités de trace compliquent la réflexion sur les chemins séparément. Lors d'une entrée x_i, par exemple, l'exécution a une certaine probabilité de prendre le premier secret et une certaine probabilité de prendre la deuxième secrète ; en un sens de vérification, le calcul prend les deux branches. Si nous raisonnons sur ces deux cas de manière isolés, nous devons suivre les probabilités de chaque branche afin de rejoindre les cas lorsque les chemins arrivent plus tard aux états observables. À condition que les chemins ne fusionnent pas avant l'atteindront aux états observables, cela est difficile même pour les petits programmes, car la probabilité d'un chemin peut avoir des dépendances complexes sur l'entrée et sur les probabilités d'autres traces possibles. Si nous raisonnons à la place sur les deux comportements séparément, nous devons fournir une analyse unique pour les exécutions qui se comportent assez différemment. Au sens large, alors, un défi central du raisonnement probabiliste de la confidentialité différentielle est d'organiser les divers comportements d'exécution en cas gérables tout en suivant proprement la relation entre les différents secrets. Pour résoudre ce problème, nous avons conçu une riche d'outils conceptuels pour construire leurs preuves, en simplifiant les arguments en abstraction intelligente de détails techniques intéressants adaptés aux notions d'adjacente, sensibilité et l'erreur de privacité ε.

ε-Confidentialité Différentielle pour les chaînes de Markov :

Dans cette section, nous proposons un cadre de travail approprié pour modéliser une grande variété des problèmes de confidentialité dans pour d'autres domaines que les bases de données statistiques. Nous appliquons la confidentialité différentielle dans l'un des deux scénarios où ($\mathcal{D}_1$ et $\mathcal{D}_2$) sont deux chaînes de Markov représentent une base de données, ou (x_1, x_2) sont deux variables appartient à un domaine arbitraire de secrets X.

Identité 1. *Soit s_1 et s_2 deux états réprésentent deux secret adjacents ssi $L_1(s_1) \equiv_1 L_2(s_2)$*
où $\equiv_1$ définie, par :
$d_H(s_1, s_2) = \{\ 1 = |\eta_1(s_1(x)) - \eta_2(s_2(x)) \mid \quad L_1(s_1) = \{a|\ \eta_1 \models a\} \ \wedge\ L_2(s_2) = \{b|\ \eta_2 \models b\}\ \}$
$\eta_1(s_1(x)) \Leftrightarrow \eta_1 \in Eval(x \in s_1)$ *: l'évaluation du variable x dans l'état s_1*
$\eta_1 \models a$ *: l'évaluation du variable satisfait à la proposition atomique a.*
Dans ce cas là, la variable x indique la cardinalité de l'ensemble X.

Définition 8. *Soit $\mathcal{D}_1 = (S_1, \bar{s_1}, \boldsymbol{P}_1, AP_1, L_1)$, $\mathcal{D}_2 = (S_2, \bar{s_2}, \boldsymbol{P}_2, AP_2, L_2)$ deux chaînes de Markov représentent deux secret adjacent. ε-Confidentialité Différentielle est une relation d'équivalence $\mathcal{R}_d^\varepsilon$ sur $\mathcal{S}_1 \times \mathcal{S}_2$ telle que pour tous couple $(s_1, s_2) \in \mathcal{R}_1^\varepsilon$*

1. $L_1(\bar{s_1}) \equiv_1 L_2(\bar{s_2})$
2. $\dfrac{\boldsymbol{P}_1(s_1, t)}{\boldsymbol{P}_2(s_2, t)} < \varepsilon$ *avec* $t \in T$ *et* $\quad T = \{t \in S_1 \cap S_2 \mid observable \in L_1(t) \cap L_2(t)\}$

Condition 1 : Affirme que deux états initiaux $\bar{s_1}$ et $\bar{s_1}$ ne diffère que par un point de données.
Condition 2 : Confirme que les probabilités conditionnelles de passer à une autre classe d'équivalence sont indistinguables, et chaque classe d'équivalence a le même observable. Voir la figure 4.3 :

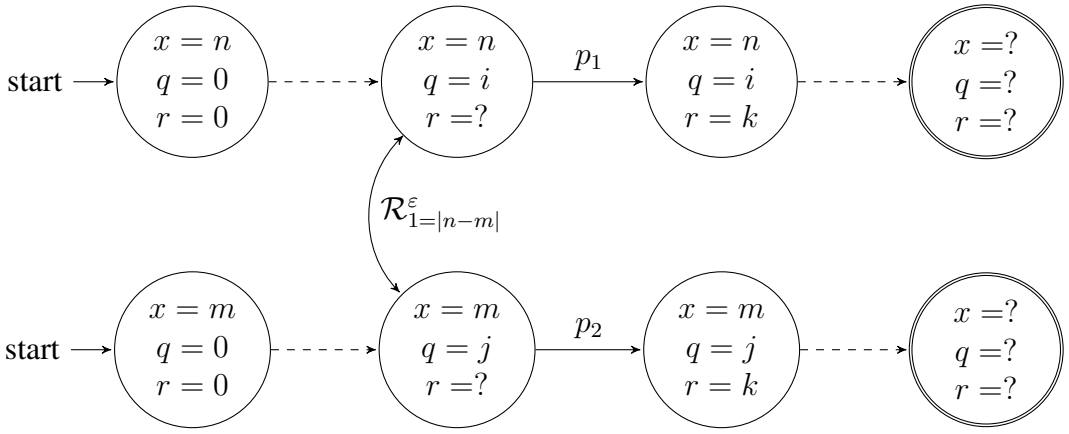


Figure 4.3 ε-Confidentialité Différentielle

Nous considérons deux variantes de la confidentialité différentielle : faible et forte (voir les **définitions 9 et 10**). La différence entre les deux est que la version faible nécessite un seul type de requêtes et la valeur de confidentialité ε soit une constante fixe, tandis que la version forte permet de varier des valeurs ε_i ,et du nombre et des types de requêtes posées, de telle sorte on s'intérésse à la valeur maximal ε_{max} .

Pour les systèmes concurrents et probabiliste, la confidentialité differentielle peut également être écrit sous la forme d'une inéquation des probabilités de transition en trois dimensions de la forme $\mathbf{P}(t|s, out_i))$: chaque case est une probabilité de transition d'un MDP, pour atteindre les états observables lors de l'exécution d'une action out_i à partir d'états secrets .

Définition 9. ***(Confidentialité différentielle faible)****. Étant donné que $\mathcal{M}$ est un MDP donne ϵ-différentielle privé ssi $\delta_{\mathcal{M}}$ induit des distributions proches sur les états associés pour toutes les s_1 et s_2 différant d'au plus un point de données. En particulier pour tout $out_i \in A_L$:*

$$\delta(s_1, out_i)(t) \leq exp(\varepsilon) \times \delta(s_2, out_i)(t)$$

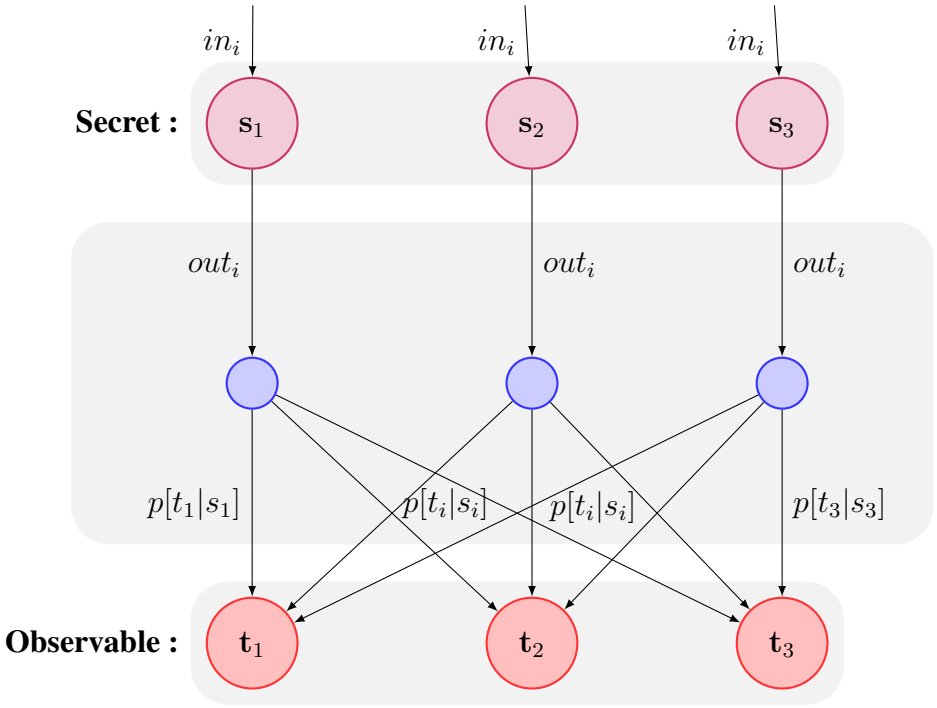


Figure 4.4 Confidentialité diffrentielle faible

En d'autres termes, deux transitions probabilistes sont similaire et indistinguable, s'ils passent par la même classe d'équivalence observable. A partir de chaque état s_i, l'état successeur s_j est choisi en deux étapes : 1) une action disponible $out_i \in A_L$ (par exemple, une réponse à une requête (i)) est choisie de façon non-déterministe s'il y plusieurs types des requêtes). 2) un état successeur est choisi aléatoirement selon la distribution de probabilité. (par exemple, selon l'espace probabiliste d'un mécanisme confidentiel).

Un système différentiellement confidentiel, s'exécutant sous le contrôle d'une politique d'ordonnancement $\sigma \in Adv_{\mathcal{M}}$. En outre, un attaquant puisse observer la sorite randomisé du programme, sélectionne une politique de d'ordonnancement, exécute le programme avec la politique choisie et observe les traces d'exécution. Pour des raisons de simplicité, on suppose que le non-déterministe est au niveau des variables d'entrée secrète adjacente. Les deux secrets ont des valeurs fixe lors de l'exécution du programme. Si l'attaquant peut déduire des informations sur les secrets en observant des séquences de valeurs de sortie publique, on dit alors que le modèle n'est pas différentiellement confidentiel.

La confidentialité différentielle forte pour les paires d'exécutions d'un MDP est défini comme suit :

Définition 10. ***(Confidentialité diffrentielle forte)****. Étant donné que $\mathcal{M}$ est un MDP donne ϵ-différentielle privé ssi pour tous adversaire $\sigma \in Adv_{\mathcal{M}}$ et pour tous chemins $\rho_1 \sim \rho_2$ différentiellement indistinguables .*

$$\boldsymbol{P}(\rho_1, \rho) \leq exp(\varepsilon) \times \boldsymbol{P}(\rho_2, \rho)$$

où :

$$\rho_1 \sim \rho_2 = \begin{cases} L(fstate(\rho_2)) \equiv_d L(fstate(\rho_1)) \\ (\rho_1(i), \rho_2(j)) \in \mathcal{R}_d^{\varepsilon} \\ (lstate(\rho_1), (lstate(\rho_2)) \in \mathcal{R}_d^{\varepsilon} \end{cases}$$

On précise que si un programme différentiellement confidentiel est exécuté à partir de deux états équivalents, alors deux exécutions doivent passer à travers la même séquence de classes d'équivalence d'observable. Ce sont capturé formellement par la définition 10.

Où, ρ_i désigne un chemin fini, modélisant les exécutions de programme qui représente par $\mathcal{M}$ sous l'adversaire σ et $\sim$ est une relation d'équivalence entre les chemins. L'intuition est que les exécutions équivalentes doivent visiter la même séquence de classes d'équivalence d'observable, mais certaines de ces exécutions peuvent être plus rapide que les autres ou bien n'ont pas la même longueur. On pourrait soutenir que cette définition de la confidentialité différentielle est trop forte

parce qu'il capture l'espace probabiliste sous l'adversaire $Pr_s^{\mathcal{M},\sigma}$ qui est défini par une fonction bijective :

$$f(s_1 \xrightarrow{out_1} t_2 \dot{\rightarrow} ...s_k \xrightarrow{in_k} s_{k+1} \xrightarrow{out_k} t_{k+1}) \stackrel{\text{def}}{=} (s_1)(s_1 out_1 t_2)...(s_1 out_1 t_2 ... s_{k+1} out_k t_{k+1})$$

Nous avons pris l'exemple dans la figure 4.1, et nous avons ajouté quelques modification au niveau des transitions comme celui illustré dans la figure ci-dessous :

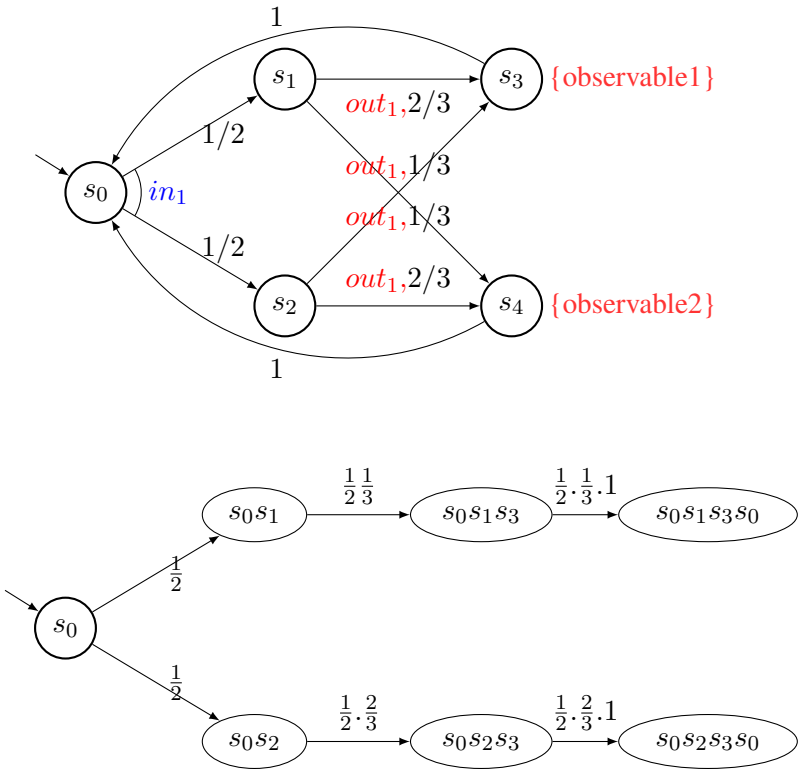


Figure 4.5 Fragment du DTMC induite

Fragment du DTMC induite $\mathcal{M}^\sigma$: où $\rho_0 = s_0$, $\rho_1 = s_1 s_3$ $\rho_2 = s_2 s_3$, telle que :

$L(fstate(\rho_2)) \equiv_1 L(fstate(\rho_1))$

$$\frac{\mathbf{P}(\rho_1, \rho)}{\mathbf{P}(\rho_2, \rho)} = \frac{\frac{1}{6}}{\frac{1}{3}} < exp(\varepsilon)$$

La figure suivante montre la notion de la confidentialité différentielle forte :

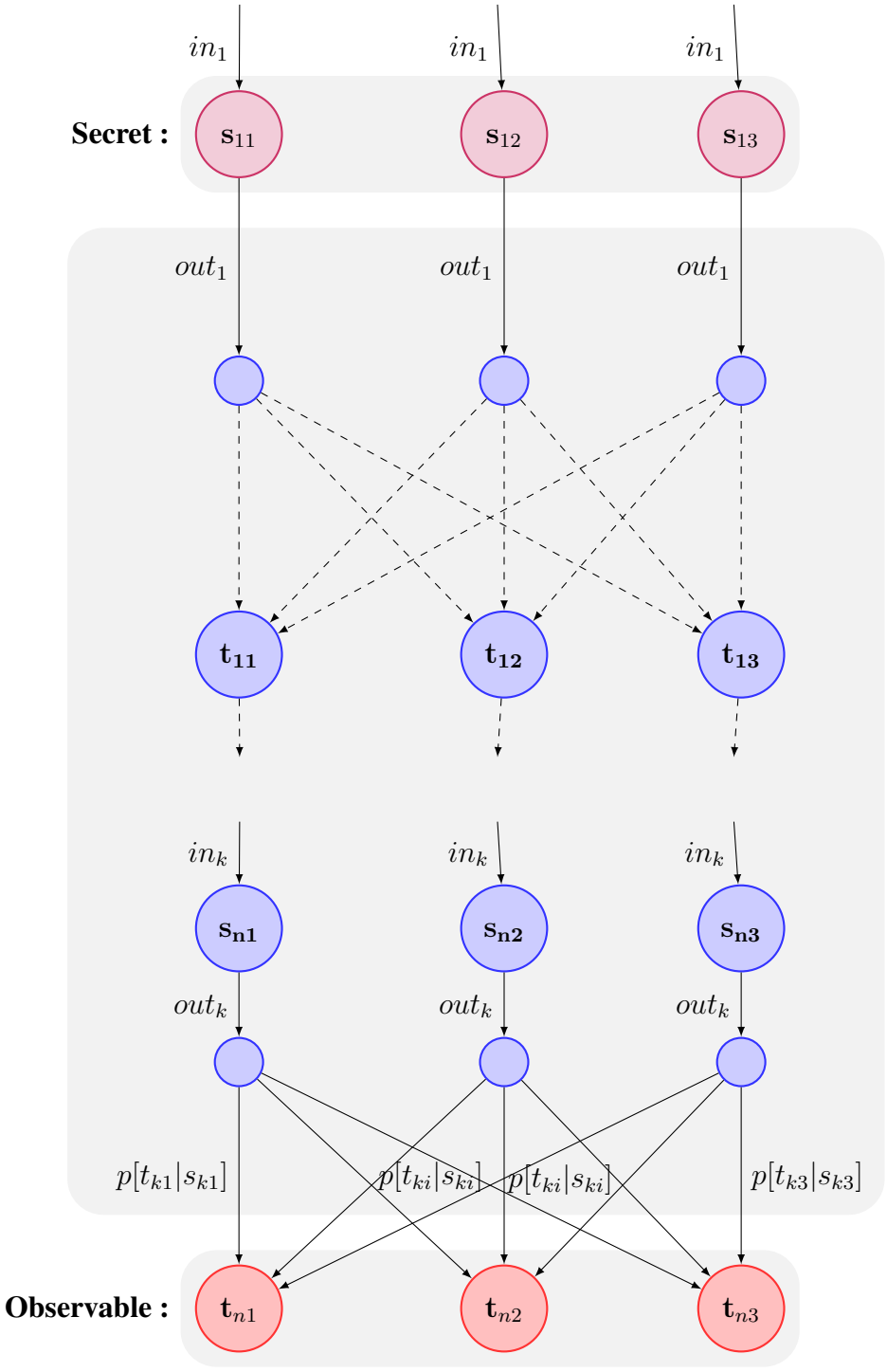


Figure 4.6 Confidentialité diffrentielle forte

4.3.4 Caractérisation de la confidentialité différentielle par PCTL*

La notion de la confidentialité différentielle peut être définie en tant que des propriétés temporelles, car il est formulé en matière de distributions de probabilités sur des chemins, des traces ou des exécutions de systèmes. À un niveau élevé, la définition exige que les distributions de probabilité sur les traces soient approximativement les même lorsque l'implémentation d'une fonction différentiellement confidentielle qui est modélisée par une chaine de Markov et il est exécutée avec des entrées qui diffèrent sur un point de données.

Notez que, puisque nous connaissons les états initiaux du modèle probabiliste $\mathcal{M}$, nous pouvons créer l'espace d'états du modèle de manière à ce qu'il puisse être atteint, de manière à ne considérons que le jeu d'états que pouvant être atteints à partir de l'état initial que représente la base de données vide, mais il est complété par des informations sur la satisfaction de la formule φ que correspondant au résultat d'exécution d'un mécanisme confidentiel. Une fois qu'un chemin ρ de $\mathcal{M}$ atteint un état d'acceptation ou bien un état d'observation d'une réponse randomisée (c'est-à-dire, un état de la forme (s, φ), il constitue un bon préfixe. De plus, $\mathcal{M} \models \varphi$ préserve les probabilités de chemin que représente d'un point de donné qu'il participe, est comparée par la probabilité de chemin d'un point de donné qu'elle ne participe pas et elle n'a aucun contrôle sur le contenu restant de la base de données.

Dans cette sous-section nous verrons comment saisir les propriétés de la confidentialité différentielle dans les modèles probabilistes (voir la section 4.3) à l'aide de la logique probabiliste temporelle $PCTL*$ Afin de construire ces propriétés, nous effectuons une suite de valeurs des probabilités dans d'une matrice de Markov et ces valeurs sont obtenues à partir de l'exécution de model checking probabiliste, et leur est déterminée et utilisée pour marquer chaque état de Markov. En conséquence, on utilise une proposition atomique $observable_i$ pour des états observables dans lesquels les réponses randomisées ont été calculées. On peut alors spécifier les probabilités dans la matrice à l'aide de la formule $Pr(s, \Diamond\phi)$, avec l'opérateur de logique temporelle, qui est souvent appelé «Finalement», on peut aussi le spécifier à l'aide de la formule $Pr(s, \phi_i \mathcal{U} \psi)$ avec l'opérateur «Jusqu'à» pour limitons l'attention aux chemins qui restent dans l'ensemble des états de telle sorte l'état de secret dans la base de données au cours d'exécution de système différentiellement confidentiel.

On reformule la confidentialité différentielle en tant que une propriété multiple de logique temporelle probabiliste : $\Phi = (\varphi_1, ..., \varphi_n)$, où n est le nombre des observables et φ est une propriété de modèle qui sera toujours exprimée par une formule de chemin :

1. $\varphi_i = \Diamond\phi_i$:
   Cette propriété signifier que la probabilité d'avoir éventuellement observé une réponse ran-

domisé i sachant qu' il y a un certain secret dans la base de données : $\mathbb{P}_{=?}[\Diamond \phi_i]$.

2. $\varphi_i = \phi_i \mathcal{U} \psi$:
   Cette propriété signifier que la probabilité pour que le secret ϕ_i doit présent ou absent dans la base de données jusqu'à une observation ψ : $\mathbb{P}_{=?}[\phi_i \mathcal{U} \psi]$.

Les propriétés de la confidentialité différentielle sont des exigences pour les traces finies qui sont formellement énoncées dans les lemmes suivants :

Lemme 1. *Étant donné deux DTMCs $(\mathcal{D}_1, \mathcal{D}_2)$, une propriété φ et un paramètre privé ε si* $\dfrac{\Pr^{\mathcal{D}_1}(\varphi)}{\Pr^{\mathcal{D}_2}(\varphi)} \leqslant exp(\varepsilon)$ *alors $\mathcal{D}_1$ et $\mathcal{D}_2$ sont ε-Confidentialité Différentielle*

Lemme 2. *Étant donné un MDP $\mathcal{M}$, une propriété φ, un paramètre privé ε et pour tout adversaire σ les DTMCs induit D_1^σ et D_2^σ telle que $Pr^{D_1,\sigma}(\varphi) \leqslant exp(\varepsilon) \times Pr^{D_2,\sigma}(\varphi)$ alors $\mathcal{M}$ est ε-Confidentialité Différentielle*

Les preuves sont données à l'Annexe A, et le tableau suivant (4.1) illustre la matrice de markov qui contient les probabilités sur les sorties autorisées qui sont calculer par model-checking probabiliste :

Tableau 4.1 PCTL* pour DP

$$\begin{array}{rc} & \text{Observables} \\ & \begin{array}{ccc} \varphi_1 & \varphi_2 & \varphi_3 \end{array} \\ \begin{array}{r} \mathcal{D}_1 \\ \text{Secrets } \mathcal{D}_2 \\ \mathcal{D}_3 \end{array} & \left[\begin{array}{ccc} P(s_1,\varphi_1) & P(s_1,\varphi_2) & P(s_1,\varphi_3) \\ P(s_2,\varphi_1) & P(s_2,\varphi_2) & P(s_2,\varphi_3) \\ P(s_3,\varphi_1) & P(s_3,\varphi_2) & P(s_3,\varphi_3) \end{array}\right] \end{array}$$

Modèle déterministe : les formules sont interprétées sur les DTMCs

$$\begin{array}{rc} & \text{Observables} \\ & \begin{array}{ccc} \varphi_1 & \varphi_2 & \varphi_3 \end{array} \\ \begin{array}{r} \mathcal{D}^{\sigma_1} \\ \text{Secrets } \mathcal{D}^{\sigma_2} \\ \mathcal{D}^{\sigma_3} \end{array} & \left[\begin{array}{ccc} Pr^{D,\sigma_1}(\varphi_1) & Pr^{D,\sigma_1}(\varphi_2) & Pr^{D,\sigma_1}(\varphi_3) \\ Pr^{D,\sigma_2}(\varphi_1) & Pr^{D,\sigma_2}(\varphi_2) & Pr^{D,\sigma_2}(\varphi_3) \\ Pr^{D,\sigma_3}(\varphi_1) & Pr^{D,\sigma_3}(\varphi_2) & Pr^{D,\sigma_3}(\varphi_3) \end{array}\right] \end{array}$$

Modèle non-déterministe : les formules sont interprétées sur un MDP

1. **Sensibilité :** La sensibilité est l'un des paramètres importants permettant de déterminer avec quelle précision le système peut répondre à des requêtes en calculant le résultat exact de certaines fonctions statistiques f, puis en ajoutant du bruit .

   La sensibilité de f est la quantité que la valeur calculée par f peut changer en ajoutant ou en supprimant un seul point de données de l'ensemble de données. Plus précisément, la sensibilité de f est la valeur maximale que $|f(x) - f(x')|$ où x et x' sont des ensembles de données que différant par un point de données. Formellement la fontion f mappe des bases de données à d nombres entiers [3] .

   Nous formalisons cette intuition sur l'ensemble des séquences d'états extraites par la logique temporelle : Dans notre modèle discret la sensibilité d'une fonction donne une limite supérieure sur le degré de perturbation de sa sortie pour préserver la confidentialité.

   La sensibilité peuvent être exprimées dans deux version l'un pour les états et l'autre pour les traces :

Définition 11. ϕ***-sensibilité***
Étant donné $f : X^n \longrightarrow \mathbb{N}^d$ une fonction sur un domaine arbitraire X. Le sensibilité de f aux états s_1 et s_2 tel que pour tous $x, x' \in X$ est

$$S_f(X) = \max_{\Delta_{(\phi_1,\phi_2)}} \| f(x) - f(x') \|_1,$$

où

$$\Delta_{(\phi_1,\phi_2)} = \{d_H(s_1, s_2) = 1 \mid s_1 \models \phi_1 \wedge s_2 \models \phi_2\}$$

Définition 12. ψ***-sensibilité***
Étant donné $f : X^n \longrightarrow \mathbb{N}^d$ une fonction sur un domaine arbitraire X. Le sensibilité de f aux traces π_1 et π_2 tel que pour tous $x, x' \in X$ est

$$S_f(X) = \max_{\Delta_{(\psi_1,\psi_2)}} \| f(x) - f(x') \|_1,$$

où

$$\Delta_{(\psi_1,\psi_2)} = \{d_H(\pi_1[i], \pi_2[j]) = 1 \mid \forall i, j > 0\ \pi_1[i] \models \psi_1 \wedge \pi_2[j] \models \psi_2\}$$

3. $\mathbb{N}^d = \mathbb{N} \times ... \times \mathbb{N}$, avec d occurrences de $\mathbb{N}$, c'est-à-dire l'ensemble de tous les d-tuples de nombres naturels.

2. **Paramètres de la confidentialité (ε) :**

   (a) **États :** Nous peuvent décrirons ε comme des échantillonnages à partir de nombre continues, mais celles-ci doivent toujours être discrétisées avec une précision finie de manière appropriée, et peut être calculé avec un nombre fini d'états.

   (b) **Récompenses :** Une structure de récompense sur un modèle de confidentialité est utile pour représenter des informations quantitatives comme la perte de la confidentialité sur le système et représentées sur les états et les actions d'un MDP comme suit : $r_{DP} = (r_s, r_\varepsilon)$
   une fonction de récompense d'état $r_s : S \to \mathbb{N}$, où $n \in \mathbb{N}$ la taille de dataset
   une fonction de récompense d'action $r_\varepsilon : S \times out_i \to \mathbb{R}_{\geq 0}$ avec $out_i \in A_L$ telle que $x \in \mathbb{R}_{\geq 0}$ est la perte de la confidentialité

4.4 Analyse quantitative de la confidentialité différentielle

Dans la section précédente, deux nouvelles notions de la confidentialité différentielle sont introduites. Ces notions permettent d'exprimer la confidentialité différentielle dans les systèmes concurrent. Intuitivement, un système est différentiellement confidentiel si, quel que soit le mécanisme, toutes les actions observables A_L peuvent être forcées de se déclencher en choisissant et en effectuant les actions contrôlables [4] appropriées en fonction du comportement observé ou bien les résultats du mécanisme. En d'autres termes, un système est différentiellement confidentiel, si l'analyste a une stratégie, à partir de chaque marquage accessible, pour forcer chaque transition observable A_L à se déclencher à l'aide des transitions contrôlables.

La vérification de la confidentialité différentielle dans les systèmes concurrents peut transformer en problème de la vérification d'une propriété de sécurité sur deux traces qui diffère par un point d'entrée. L'idée présentée dans ce projet est celle de vérifier la confidentialité différentielle sur une trace unique et même sur plusieurs traces en comparent les probabilités d'accessibilité de ces traces vis-à-vis à un paramètre ε. cette vérification est assez complexe et peut être gérée efficacement par les vérificateurs de modèles probabilistes existants, nous avons proposé des techniques automatisé pour vérifier une version non déterministe de la confidentialité différentielle. Notre approche est paramétrée par une politique d'ordonnancement et des propriétés en $PCTL^*$, il est donc possible de vérifier si le programme sera confidentiel en vertu d'une politique d'ordonnancement particulière qui représente un canal. Si la planification changements de politique, le canal doit être rétabli pour cette nouvelle politique de d'ordonnancement. Il est donc donne une garantie pour tous les politiques de d'ordonnancement.

4. les actions contrôlables sont des actions avec une condition que le budget de la confidentialité n'est pas encore épuisé

1. **La confidentialité différentielle :** Étant donné un modèle $\mathcal{M}$ et une proprété φ, est-ce qu'est différentiellement confidentiel ?
2. **Précision :** Étant donné un modèle différentiellement confidentiel $\mathcal{M}$ et une proprété φ, maintient-il ses promesses de précision ?
3. **Perte de la confidentialité :** Pour la quantification de la perte de la confidentialité - Comment et quelle sont les paramètres et les techniques que améliore la confidentialité ?
4. **La confidentialité différentielle changée :** Pour garantit que la modification des données d'un seul individu dans la base de données entraîne une légère modification de la distribution sur les sorties.
5. **La simulation semi honnête :**

   La confidentialité différentielle est formalisée selon le paradigme de la simulation en exigeant l'existence d'un simulateur qui génère la vue d'une partie dans l'exécution, en disant que le point de vue d'une partie dans l'exécution d'un système peut être simulé en compte tenu de ses entrées et de ses sorties.

4.4.1 La confidentialité différentielle :

Nous avons proposé une méthode basée sur le calcul du canal qui vise à vérifier la notion de la confidentialité diffrentielle dans le pire de cas et qui est spécifique à un ordonnanceur.

La méthode général pour calculer le canal que correspondant à un tel système DP est la suivante :

1. Nous identifions les ensembles $secrets$ et $observables$ représentant respectivement les valeurs secrètes et observables du système, Dans notre méthode, il y a deux façons :
   - Spécification des états initiaux, et chaque état initial d'un modèle est défini par une valeur initiale de $secret$, il devrait y avoir un prédicat sur toutes les variables du modèle.
   - Utilisation des étiquettes pour identifier de manière unique des ensembles d'états $secretes$ et $observables$
2. Nous utilisons PRISM pour calculer la probabilité conditionnelle $P(observable_i|secret_j)$ pour chaque entrée et chaque sortie, que correspondant au probabilité de prendre un chemin à partir de l'état initial qui respecte un chemin de propriété propre $P = [path]$.(Voir 4.3.4)
3. Nous comparons systématiquement les probabilités $P(observable_i|secret_j)$ pour chaque paire $observable_i, secret_j$, qui définit notre canal.

Nous avons formulé une conjecture décrivant l'idée pour dériver les canaux que représentant le comportement d'un système différentiellement confidentiel :

Conjecture 1. *Étant donnée un MDP $\mathcal{M}$, une liste des PCTL* $\varphi = (\varphi_1, ..., \varphi_n)$ et un paramètre privée ε, si $\forall \sigma^{max} \in Adv_{\mathcal{M}}\ \forall \varphi_i$ telle que $\Pr^{D_1,\sigma^{max}}(\varphi_i) \leq exp(\varepsilon) \Pr^{D_2,\sigma^{max}}(\varphi_i)$ alors $\mathcal{M}$ est un modèle diffrentiellement confidentiel.*

4.4.2 Précision :

Dans la mesure où la confidentialité différentielle est de plus en plus utilisée dans des contextes pratiques, on peut souvent penser qu'il existe une exigence de précision pour un calcul donné et que l'analyste de données souhaite maximiser la confidentialité du calcul soumis à la contrainte de précision. Cela soulève la question de savoir comment trouver et minimiser le risque empirique privé soumis à une exigence de précision donnée.

Notre approche de vérification de la précision sera la suivante : on spécifier des propriétés par des séquences des réponses possible, chacune avec une précision croissante et une confidentialité réduite et cette vérification de leurs niveaux de précision est effectué de sous forme des contraintes de précision.

Proposition 1. *Nous dirons qu'une chaîne de Markov $\mathcal{D}$ d'un mécanisme privé qui produit une suite des réponses randomisé à une suite de k requêtes est (v, p_1, p_2)-précis par rapport à un seuil T.*

$$\bigwedge_i^k (\bigvee_j^n (R_j + v_i < T))$$

Avec :

- R_j : *Réponse réel à j-ième requête.*
- v_i : *Bruit randomisé ajouter à i-ième réponse réel possible .*
- T : *seuil*
- p_1 : "La probabilité de rester pour toujours dans les états au-dessus du seuil. "**AboveThreshold**"

$$P[\,\Box\, AboveThreshold]$$

- p_2 : "La probabilité possible d'atteindre "**UnderThreshold**"

$$P[\,\Diamond\, UnderThreshold]$$

La propriété **AboveThreshold** prend en compte un jeu de données et une séquence de requêtes et une sortie privée, on vérifier si certain réponse peut dépasser un seuil donné (avec une erreur à cause du bruit). Nous utilisons un algorithme de model-checking pour controler d'exactitude du **AboveThreshold**.

4.4.3 Quantitative de la perte de la confidentialité :

La quantification de la perte de la confidentialité (en anglais *privacy loss*) permet dû l'analyse et le contrôle de la perte cumulative de la vie privée sur plusieurs calculs en séquence ou en parallèle, et pour comprendre le comportement des mécanismes différentiellement confidentiels dans un environnement distribué et permet d'aussi concevoir et d'analyser des modèles complexes différentiellement confidentiel. Nous donnons une méthode d'analyse quantitative à condition quelques-uns soient «significatifs» pour raisonner sur la perte cumulée de vie privée chez un seul individu dont les données pourraient être transmises à plusieurs ensembles de données, chacun pouvant être utilisé de manière indépendante de manière différemment confidentielle.

Le problème est que l'adversaire peut en fait influencer l'organisation de ces ensembles de données par leur interrogation de manière répétée des bases de données dynamique [5].

Nous utilisons des propriétés basées sur les récompenses, qui capturent une variété de mesures quantitatives supplémentaires. Pour assurer dans le cas le plus défavorable (sur toutes les adversaires possibles de processus markovien) le nombre de requêtes pendant l'exécution d'un système.

$$R_{max=?}[F\ observable]$$

Exemple : Une structure de récompense «nombre de réponse avec une précision triviale » est définie par r_s égal à 1 avec une condition "un bruit supppèrieur à l'erreur d'échantillonnage" .

4.4.4 La confidentialité différentielle changée :

La confidentialité différentielle changée est une conséquence intuitive de la confidentialité différentielle que ne rend pas explicitement la notion de changement envisagée. Il compare implicitement la distribution probabiliste sur la sortie sans prendre en considération le paramètre de confidentialité, par exemple, une variable aléatoire Obs, contient les observables avant et après le changement du secret. Si nous nous concentrons sur un seul individu x_i qu'est une variable aléatoire représentant son point de données lorsqu'il change sa valeur, la comparaison s'effectue alors entre $Pr[Obs = o\ lorsque\ x_1 = 1]$ et $Pr[Obs = o\ lorsque\ x_1 = 0]$. La partie de cette caractérisation de la confidentialité différentielle est informelle ce qui rendrait la notion de changement précise est donné par cette équation :

$$DPC = Pr_{max}[\Diamond\phi_1] - Pr_{max}[\Diamond\phi_2]$$

telle que ϕ_1 et ϕ_2 deux observables avant et aprés le changement .

5. il s'agit d'un problème fondamentalement différent de celui qui consiste à interroger de manière répétée une seule base de données fixe.

4.5 Notre méthodologie d'analyse de la confidentialité différentielle

La méthodologie d'analyse de la confidentialité différentielle que nous avons adoptée dans ce projet, illustrée à la figure au-dessous, est classique au probabilistic model cheking avec, toutefois, une nuance liée à la confidentialité diffrentielle.

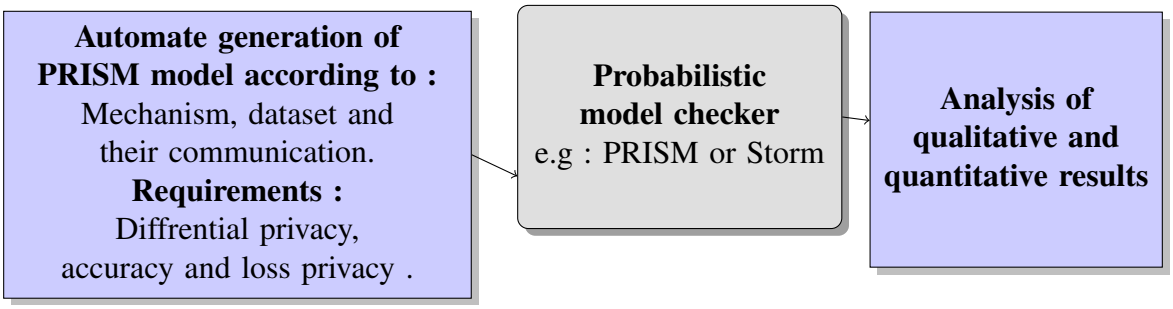


Figure 4.7 Méthodologie d'analyse de la confidentialité différentielle

Cette méthodologie suivre exactement trois étapes : L'étape 1 consiste à modéliser le comportement du système que est décomposé en trois aspects qui sont, le mécanisme de la confidentialité, la communication de ses sous-systèmes et la nature de la base de données. Les modèles correspondants sont spécifiés à l'aide d'un formalisme qu'est un réactif module probabiliste que sont ensuite traduit en un formalisme à états transitions probabiliste, tels que le processus de décision markovien, ces modèles peut être généré automatiquement par PRISM.

Par ailleurs, les exigences de la confidentialité des systèmes sont, quant à-elles spécifiées sous la forme de propriétés de logique temporelle adéquate comme la confidentialité différentielle changée, précision et la perte de la confidentialité (étape 2). Les logiques temporelles utilisées doivent permettre l'expression et l'analyse de plusieurs propriétés probabiliste. Ces propriétés sont ensuite vérifiées et analysées sur le modèle par l'outil PRISM (étape 3).

4.6 Modélisation de système différentiellement privé par PRISM

Nous donnons la sémantique de la confidentialité différentielle d'un module système, telle que construite à l'aide des règles. Nous supposons que :

— $C = \{C_{req}, C_{res}\}$ est un multiensemble des commandes.

— $Var = \{v_1, ..., v_m\}$ est l'ensemble des variables, locales et globales, qui apparaissent dans la description du système.

Nous considérons maintenant la sémantique pour quatres type de commandes du module système différentiellement privé. chaque commande de C prend la forme : $[\alpha]\ g\ \rightarrow p_1 : u_1 + ... + p_n : u_n;$
Avec : $\alpha \in A$ est une action

g : une garde g est un prédicat sur les variables en Var et chaque état du système est une évaluation de ces variables.

p_i : probabilité

u_i : une mise à jour des variables

1. $c \in C_{req}$ pour l'entrée du niveau bas (L) :

$$c = [in_L]\ g\ \rightarrow p_1 : u_1 + ... + p_n : u_n;$$

Avec in_L : Correspondant à une requête ou une fonction appliquée à une base de données.

g : un prédicat pour le choix d'une requête à appliquer.

p_i : probabilité dépendance entre l'un des composants d'un système et d'une requête.

u_i : Évaluation d'une requête par une expression booléenne sur toute les points de données.

2. $c \in C_{res}$ pour la sortie du niveau bas (L) :

$$[out_L]\ g\ \rightarrow p_1 : u_1 + ... + p_n : u_n;$$

Avec out_L : Correspondant à une réponsé randomisée à une requête.

g : un prédicat sur la valeur du requête

p_i : La probabilité d'observer une réponse randomisée, telle que $[p_i, ..., p_n]$ est une ligne d'un de matrice d'un mécanisme de confidentialité.

u_i : bruit ajouté à chacune des sorties

3. $c \in C_{req}$ pour l'entrée du niveau haut (H) :

$$[in_H]\ g\ \rightarrow p_1 : u_1 + ... + p_n : u_n;$$

Avec in_H : Correspondant à l'ajout d'une ou plusieurs points de données à une base de données.

g : un prédicat définit une politique de confidentialité pour une politique de confidentialité selon le contexte de modélisation

p_i : connaissances auxiliaires d'un attaquant sur certaines des points de données .

u_i : Mise à jours d'une ou plusieurs points de données.

4. $c \in C_{res}$ pour la sortie du niveau haut (H) :

$$[out_H]\ g\ \rightarrow p_1 : u_1 + ... + p_n : u_n;$$

Avec out_H : Correspondant à une réponse privée .

g : un prédicat définit une politique de confidentialité.

p_i : probabilité établie selon le contexte du système à modéliser

u_i : Mise à jours d'une ou plusieurs points de données.

Nous illustrons maintenant le comportement de ce système dans le langages de modélisation et de spécification de PRISM à travers des exemples .

1. **Jeux de données (en anglais Data set) :** D'abord, les modules réactifs probabilistes ne supportent pas les structures de données complexes (par exemple, les tableaux et les canaux) et nécessitent un codage de commandes conditionnelles au moyen de pré-condition et post-condition .

   Nous avons deux modèles pour modéliser l'espace des points de données :

   (a) Modèle de valeur insensitive :
   Dans ce modèle nous mesurons le coût de la confidentialité uniquement par rapport à la manière dont le mécanisme traite le bit sensitive, et ignorons la manière dont il traite les valeurs privés, $U = \{0, 1\}$

(b) Modèle de valeur sensitive : Dans ce modèle nous mesurons la vie privée par rapport à la façon dont elle traite la paire (b_i, v_i) pour chaque individu ou point d'entrée.
$U = \{0, 1\} \times \mathbb{N}$

Les listings 4.1 et 4.2 présentent des modèle de l'espace des points de données.

```
1 data : [0..1] init 0 ;
```

Listing 4.1 Modèle de valeur insensitive

```
1 data : [DATA_MIN..DATA_MAX] init 0 ;
```

Listing 4.2 Modèle de valeur sensitive

2. **Requête de comptage (en anglais Counting query) :** nous avons considéré seulement la requête de comptage qu'est une requête standard , primitif et extrêmement puissant.
Une requête de comptage est une fonction :

$$f : X \longrightarrow \{0, 1\}$$

mappe les points de la base de données sur les valeurs booléennes, plutôt que toute valeur dans l'intervalle. Cette requête de comptage est tout simplement une requête linéaire généralisée pour prendre des valeurs des point de données dans l'intervalle [0,1].

Le code du listing 4.3 représente une requête de comptage.

```
1 const int k;
2 formula count=(data0 > k  ?1:0)+(data1 > k  ?1:0)+
3               (data2 > k  ?1:0)+(data3 > k  ?1:0)+
4               (data4 > k  ?1:0)+(data5 > k  ?1:0)+
5               (data6 > k  ?1:0)+(data7 > k  ?1:0);
```

Listing 4.3 Requête de comptage

3. **Mécanisme de la confidentialité :** Nous supposons que le système implement correctement les mécanismes de la confidentialité. Nous identifions un mécanisme $\mathcal{K}$ différentiellement confidentiel par sa matrice de canal de probabilité $\mathcal{P}_{i,j}$, et chaque case représente une probabilité conditionnelle que la sortie est j étant donné que l'entrée est i.
Les listings 4.4 et 4.5 présentent un cas très simple de canal confidentiel.

```
1 const double P1_0_0=0.6666666666666666;
2 const double P1_1_0=0.3333333333333333;
3 const double P1_0_1=0.3333333333333333;
4 const double P1_1_1=0.6666666666666666;
```

Listing 4.4 Les probabilités conditionnelle

```
1 [] (s=3) & (n=1) & (count = 0 )->  P1_0_0 :(RR'=0)  & (s'=4) +
                                     P1_1_0 :(RR'=1)  & (s'=4) ;
2 [] (s=3) & (n=1) & (count = 1 )->  P1_0_1 :(RR'=0)  & (s'=4) +
                                     P1_1_1 :(RR'=1)  & (s'=4) ;
```

Listing 4.5 Les transitions probabilistes dans le modèle présents les sortie du mécanisme

4. **Fournisseur de données (en anglais Data Provider) :** Une des tâches importantes du fournisseur de données est l'envoie des points de données au système. Le pseudocode du listing 4.6 illustré la fonction principale du fournisseur de données.

```
1  module DataProvider
2      s0: [0..1];
3          DATA_x : [DATA_MIN..DATA_MAX];  //datapoint0 , datapoint1
                                              ... and datapointn
4
5          [] (s0=0)  -> (s0'=1) & (DATA_x'=0) ;
6          //.....
7          [] (s0=0)  -> (s0'=1) & (DATA_x'=DATA_MAX) ;
8          [datapoint] (s0=1)  -> (s0'=1-s0);
9          //...
10 endmodule
```

Listing 4.6 Fournisseur de données

5. **Agent et le budget de la confidentialité :** La confidentialité différentielle est une propriété composable et doit être limiter le nombre de requêtes, cet concept donne lieu à la notion du budget de la confidentialité. Chaque calcul différentiellement privé avec un paramètre de confidentialité ε a pour résultat de soustraire epsilon de ce budget. Une fois budget est épuisé, les résultats des requêtes ne peuvent plus être calculé. L'agent de la confidentialité jouer un rôle d'arbitrage en fonction de la valeur du budget pour faire une décision d'accepter ou non les requêtes .
6. **Examinateur de données (en anglais Data Examiner) :** On suppose que l'examinateur de données n'a aucune connaissance de la base de données privée, ne peut pas calculer la valeur de la sortie d'un mécanisme différentiellement privé et doit la rendre moins privée. Autrement

dit, l'examinateur de données ne peut pas augmenter la perte de confidentialité, que ce soit dans la définition formelle ou même dans un sens intuitif. Dans cette partie en réfléchissant au résultat de modèle, quelles que soient les informations auxiliaires disponibles.

Les listings 4.7 et 4.8 présentent les codes correspondant à un agent de confidentialité et un examinateur de données.

```
1 const int alpha;
2 module Agent
3 b : [0..alpha] init alpha  ;   // budget of privacy
4     [datapoint]-> (s=0) & (apply=false)-> (s'=0) & (b'=alpha);
5     [query](s=0) & (apply=true)-> (count'=mod(count,N)) & (s'=1) &
                                    (b'=mod(b-1,alpha)) ;
6     [](s=1) -> (s'=0) ;
7 endmodule
8 formula  apply=((b - epsilon) > 0  ? true:false) ;
```

Listing 4.7 Agent de la protection de la confidentialité

```
1 module DataExaminer
2  s2: [0..1];
3  [] s2=0 -> (s2'=1-s2);
4  [query] s2=1 -> (s2'=1-s2);
5 endmodule
```

Listing 4.8 Examinateur de données

4.7 Spécification de la notion de la confidentialité différentielle

1. **Voisinage des donneés *(en anglais Neighbors data)* :** Nous pensons que $\mathcal{D}^n$ spécifient les bases de données sur n éléments et leurs échantillons nous donne une paire de bases de données voisines : **`secret1`** et **`secret2`** qui diffère par la remplacement du ième élément de $data_i$ pour une certaine valeur x_i. Cette spécification est présentée dans la listings 4.9.

```
1 label  "secret1" = data1=1 & count=c+1 &  n=m  ;
2 label  "secret2" = data1=0 & count=c   &  n=m  ;
3 label  "observable_k" =  RR=k ;
```

Listing 4.9 Voisinage des donneés

2. **Précision (*en anglais Accuracy)* :** Nous formalisions les besoins d'utilité d'un système diffrentiellement privé par trois paramètres : **`T`** décrit l'exigence de précision (être considéré comme un seuil dans une inéquation), **`RR`** est la réponse randomisé et le **`noise`** décrit le bruit ajouté à des réponses réels. Le listings 4.10 présente une solution, associée avec la spécification de la précision.

```
const int T ; // Threshold
formula noise=(RR - count > 0  ? RR - count: count -RR ) ;
formula AboveThresold=(RR - noise < T  ? true: false ) ;
formula UnderThresold=(RR - noise > T  ? true: false ) ;

Pmax=? [ G "AboveThresold" ]
Pmin=? [ F "UnderThresold" ]
```

Listing 4.10 Précision

3. **Perte de la confidentialité (*en anglais Loss privacy)* :** On considère la récompense cumulée obtenue avant qu'un certain nombre d'états cibles dont sont représente une exécution d'un mécanisme privé pour calculer «le coût de confidentialité prévu de l'achèvement d'une observable » ou «la perte de confidentialité attendue pendant la durée de vie d'un système différentiellement privé ». Cette spécification présentée dans le listing 4.11.

```
rewards}  "privacy_loss"
[query] AboveThresold : 1 ;
endrewards

rewards  "privacy_cost"
[query] UnderThresold : 1 ;
endrewards

rewards  "Query"
[query] true : 1;
endrewards
```

Listing 4.11 Perte de la confidentialité

4.7.1 Analyse des modèles de la confidentialité differentielle avec PRISM

Cette section explique comment le système différentiellement confidentiel peut être vérifié avec le model-checker probabiliste. Notre définition de la confidentialité différentielle est formellement énoncée à la section 4.3.3. Nous décrivons dans cette section l'idée principale pour vérifier cette définition et nous prouvons sa validité. Généralement, il ne suffit pas d'étudier seulement les programmes différentiellement confidentiels, parce que parfois, ces programmes sont également testés et après déployer dans environnement distribué dans lequel des informations privées sont divulguées ou bien il y a un risque des fuites d'informations, alors il est important de modéliser et d'étudier le comportement du ces programmes en utilisant un modèle-complet du système. Le domaine de la protection de la vie privée pour ces systèmes regorge de questions de confidentialité et de flot d'information. Pour résoudre ces problèmes, il est nécessaire de construire et d'étudier des modèles d'analyse et de simulation. La confidentialité différentielle n'est pas une notion binaire alors il est nécessaire de développer des modèles pour une compréhension qualitative et quantitative. En plus la nature du trafic entre l'examinateur de données et le système est imprévisible et il est donc typique de développer un modèle indéterministe et probabiliste pour représenter ces systèmes. Dans cette étude, nous avons utilisé PRISM pour construire ces modèles.
Dans ce travail, nous avons proposé un modèle basé sur les différentes relations entre les trois entités (le fournisseur des données, le détenteur de donnée et l'examinateur des données) et les probabilités des réponses randomisées possibles qui sont représentés par une chaîne de Markov. Nous avons présenté la modélisation d'un modèle comportemental via langage PRISM.

De plus, afin d'étudier le comportement quantitatif des mécanismes confidentiels dans un système concurrent, nous allons utiliser deux types de techniques d'analyse : 1) la simulation peut générer différents scénarios de l'examinateur de données. On utilise le mode de simulation pas à pas pour voir les valeurs des points de données et la qualité de la réponse randomisée qu'est mise en évidence par le mécanisme. 2) La vérification de modèle probabiliste pour calculer les différentes probabilités des observables qui sont des chemins d'exécution obtenus à partir d'entrelacement des programmes ont été explorés, autrement dit le model-checker probabiliste examine systématiquement tous les entrelacements des entités d'un système différentiellement confidentiel dans un cadre spécifique pour calculer la matrice probabiliste des canaux de la confidentialité.

4.8 Études de cas et résultats expérimentaux

L'un des objectifs clés de ce projet est de développer des méthodes pouvant être appliquées à des exemples concrets. Dans cette section, nous fournissons une étude de cas : Système différentiellement confidentiel des dossiers médicaux des patients dans un hôpital qui libèrent des informations statistiques et globales préservant la confidentialité. L'hôpital examine les dossiers des maladies pour faire une étude clinique qui explore la relation entre le tabagisme et les maladies pulmonaires, et affiche ces résultats chaque jour de l'étude. Une compagnie d'assurance qui ne comprenait pas a priori cette relation que pourraient modifier radicalement ses convictions comme les primes d'assurance afin de prendre en compte les résultats de l'étude. Dans ce cas, les conclusions tirées par la compagnie sur le risque de quiconque, par exemple, la personne Alice sont fortement affectées par les résultats de l'étude. La figure 4.8 suivante réprésente un système diffrentiellement confidentiel : *DataProvider* : Base de données de l'hopital, *DataHolder* : système similaire à PINQ et *DataExaminer :* compagnie d'assurance.

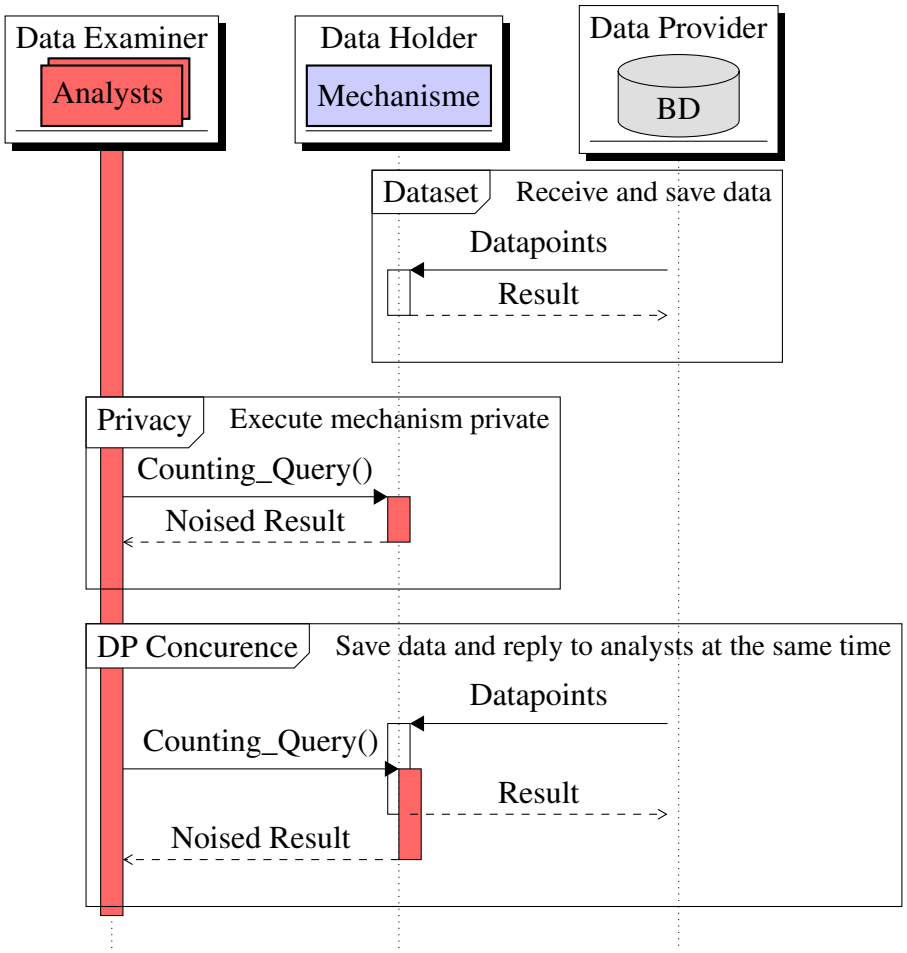


Figure 4.8 Système différentiellement privée : synchrones et mutables

4.8.1 Modèle de la confidentialité différentielle

1. **Politique de confidentialité :**

   Nous avons une base de donnés qui contient 10 dossiers médicaux et leurs niveaux de la confidentialité d'un dossier est associé à chaque $datapoint$ (il n'y a donc que $datapoint$ dans la base de données par un dossier médical), il est maintenant possible d'interpréter les implications des paramètres de confidentialité en termes de sensibilité des dossiers. Par exemple, si ε_i de la requête $COUNT$ est $ln(2)/budget$, une nouvelle $datapoint$ est généré tous les temps et supprimé après i-ième requêtes selon la taille de la base de donnés.

   La politique de confidentialité attachée à des $datapoints$ peut également indiquer un délai d'expiration après lequel le datapoint est supprimé [6]. Étant donné que chaque $Dataprovider$ génère de nouveaux $datapoints$ à un taux inconstant et indéfini.

2. **Méchanime Géométrique Tronqué :** Le méchanime Géométrique Tronqué (*The Truncated Geometric Mechanism (TGM)* [38]) est une adaptation du mécanisme de Laplace, conçu pour produire des sorties sur une plage limitée de valeurs discrètes. (Voir le tableau 4.2)

Tableau 4.2 Méchanime $\frac{1}{2}$-Géométrique Tronqué

Secret/Observable	Obs_0	Obs_1	Obs_2	Obs_3	Obs_4	Obs_5	Obs_6
Sec_0	$2/3$	$1/6$	$1/12$	$1/24$	$1/48$	$1/96$	$1/96$
Sec_1	$1/3$	$1/3$	$1/6$	$1/12$	$1/24$	$1/48$	$1/48$
Sec_2	$1/6$	$1/6$	$1/3$	$1/6$	$1/12$	$1/24$	$1/24$
Sec_3	$1/12$	$1/12$	$1/6$	$1/3$	$1/6$	$1/12$	$1/12$
Sec_4	$1/24$	$1/24$	$1/12$	$1/6$	$1/3$	$1/6$	$1/6$
Sec_5	$1/48$	$1/48$	$1/24$	$1/12$	$1/6$	$1/3$	$1/3$
Sec_6	$1/96$	$1/96$	$1/48$	$1/24$	$1/12$	$1/6$	$1/3$

3. **Jeu de données :** Ce système possède une base de données mutable et peut fournir des limites d'erreur de confidentialité en supprimant les points de données surutilisés.

4. **Requête de comptage** : C'est une requête pour de calculer une estimation du nombre de personnes ont la maladie pulmonaire obstructive chronique (MPOC) dans un échantillon médicale qui change dynamiquement leur volume, l'échantillon contenant au maximum n = 10 patients, dont m ont cette maladie. On note : l'observable est le nombre randomisé des patients ont la maladie respiratoire et le secret est le nombre exact tel que Alice dans la liste ou pas.

6. Nous discuterons de l'impact du délai d'expiration sur la confidentialité au niveau de l'utilisateur dans d'autres modèles comme PTA

4.8.2 Les résultats d'analyse :

Nous présentons le processus de vérification dans un cadre particulièrement représentatif d'un cas général des systèmes différentiellement confidentiel : non-déterministe, conservative (ou base de données évolutive) et non interprétable. Cette recherche est une étude exploratoire sur la manière dont les interactions entre les composants du système et dont les paramètres de la confidentialité ont un effet sur la perte de la confidentialité.

Dans cette expérience, nous analysons l'impact de diverses décisions de modélisation sur la confidentialité et la précision des résultats randomisées et leur utilité. Nous considérons différents paramètres pour le mode de fonctionnement d'un adversaire (non déterministe), et la modélisation de la longueur de la base de données (statique ou dynamique), nous présentons les résultats de la simulation semi-honnête de la vérification obtenus avec l'outil PRISM.

Simulation semi-honnête :

Nous utilisons d'abord la technique de simulation pour étudier le modèle à court terme en générant un chemin randomisé et en suivant les états représentent l'ensemble des possibilités des réponses à une requête bien définie. Nous examinons d'abord l'effet de différentes réponses randomisé et leur réponse réelle. Considérons que la taille du de la base de données variée au cours de temps .Nous étudions ce cas avec différentes valeurs de la confidentialité et en d'autres termes nous étudions un modèle de mémoire reçoivent des données de façon continue comme indiquée dans la figure suivant, nous montrons les résultats expérimentaux pour une requête de comptage .

La figure 4.9 montre le comportement dynamique du modèle proposé avec la mise en œuvre du mécanisme confidentiel. Il montre le comportement de tous les états du système proposé en ce qui concerne l'entrelacement entre les composants du système qui sont les *DataProvider* et *DataExaminer*. Les résultats prédisent que le modèle fourni est asymptotiquement confidentiellement différentiel. En outre, il montre également que la taille de la base de données a un impact puissant sur toutes les réponses. Initialement, on voit clairement dans l'instant Time=96 le secret est size=3,count=2 et l' observable est RR= 3 et nous observons aussi dans Time=104 secret est size=3,count=3 et observable est RR= 3 , ce qui prouve notre hypothèse selon laquelle deux entrées similaires donnent des distributions de sortie similaires .

Path

Step		DataProvider		DataHolder														DataExaminer	Rewards		
Action	#	s0	DATA_x	s	c	v0	v1	v2	n	count	RR	data0	data1	data2	data3	data4	data5	s2	["loss...	["Cost...	["Que...
DataProvider	88	1	1	1	2	1	0	1	5	0	0	1	-1	-1	-1	6	3	1	0	0	0
DataHolder	89	1	1	0	2	1	0	1	5	0	0	1	-1	-1	-1	6	3	1	0	0	0
[datapoint]	90	0	1	1	2	1	0	1	5	0	0	1	-1	-1	-1	6	3	1	0	0	0
DataProvider	91	1	5	1	2	1	0	1	5	0	0	1	-1	-1	-1	6	3	1	0	0	0
DataHolder	92	1	5	0	2	1	0	1	5	0	0	1	-1	-1	-1	6	3	1	0	1	1
[query]	93	1	5	2	2	1	0	1	5	0	0	1	-1	-1	-1	6	3	0	0	0	0
DataExaminer	94	1	5	2	2	1	0	1	5	0	0	1	-1	-1	-1	6	3	1	0	0	0
DataHolder	95	1	5	3	2	1	0	1	3	2	0	1	-1	-1	-1	6	3	1	0	0	0
DataHolder	96	1	5	4	2	1	0	1	3	2	3	1	-1	-1	-1	6	3	1	0	0	0
DataHolder	97	1	5	5	0	1	0	1	3	0	0	1	-1	-1	-1	6	3	1	0	0	0
DataHolder	98	1	5	0	0	0	0	1	3	0	0	-1	-1	-1	-1	6	3	1	0	0	0
[datapoint]	99	0	5	1	0	0	0	1	3	0	0	-1	-1	-1	-1	6	3	1	0	0	0
DataProvider	100	1	8	1	0	0	0	1	3	0	0	-1	-1	-1	-1	6	3	1	0	0	0
DataHolder	101	1	8	0	0	1	0	1	4	0	0	8	-1	-1	-1	6	3	1	0	1	1
[query]	102	1	8	2	0	1	0	1	4	0	0	8	-1	-1	-1	6	3	0	0	0	0
DataHolder	103	1	8	3	0	1	0	1	3	3	0	8	-1	-1	-1	6	3	0	0	0	0
DataHolder	104	1	8	4	0	1	0	1	3	3	3	8	-1	-1	-1	6	3	0	0	0	0

Figure 4.9 L'exploration d'un modèle DP à l'aide du simulateur

Pour trouver la probabilité des différents sortie dans le cas d'un flux de base de données dans le contexte dynamique, on vérifions la propriété : **`Pmax=? [ F<=T "Outputs" ]`**. telle que : **`label "Outputs" = RR=N-2 & count=N-1 & n=N ;`**

Le graphique 4.10 suivant représente les cas où la base de données est de taille **N** = 3,5,7 et 9, la probabilité en tant que valeur de **N** varie, que capture la dépendance à des paramètres spécifiques qui sont l'ordonnanceur non-déterministe et la taille de la base de données qui affectent la précision de la réponse randomisé.

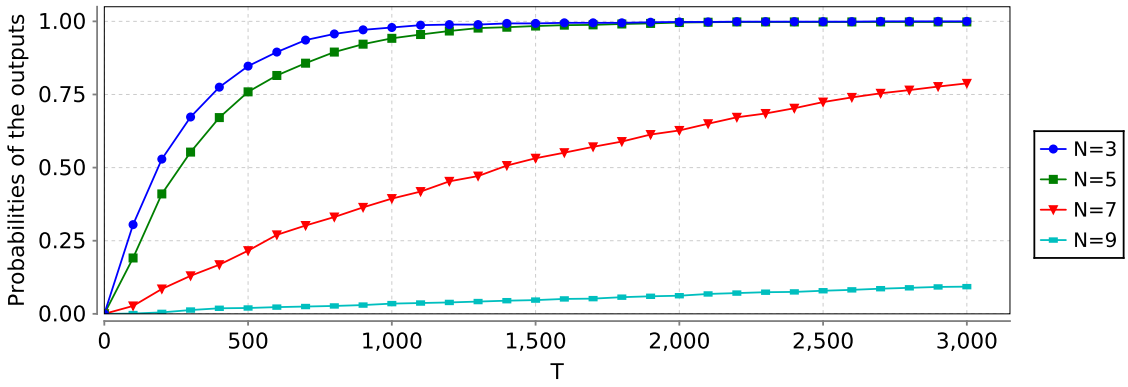


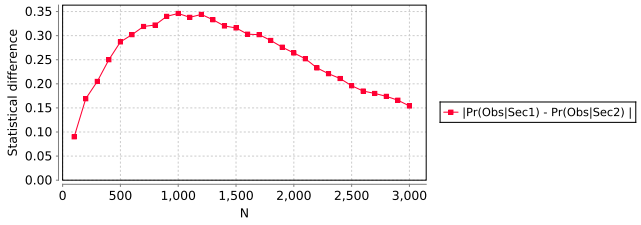
Figure 4.10 Probabilités des sorties dans un context dynamique

Vérification :

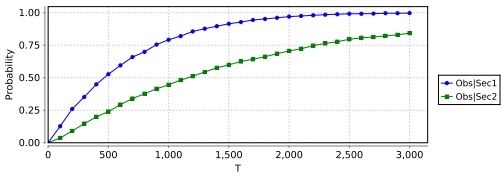
D'abord, nous calculons la probabilité maximal **Pmax**, qu'après un nombre fini d'étapes **T**, de rapporter une réponse randomisé sachant qu'il y a un certain secret, nous exprimons ceci en PRISM comme suit : **Pmax=? [F "Obs_Sec1"]** , **Pmax=? [F "Obs_Sec2"]**

Les figures suivantes 4.11 montre la limite de deux différent probabilités d'avoir la même sortie pour les entrées voisines, dans ce cas, le paramètre de confidentialité est non spécifié **t=?**.

```
label "Obs_Sec1" = RR=4 & count=4 & n=5
label "Obs_Sec2" = RR=4 & count=3 & n=5
```



(a) Différence statistique

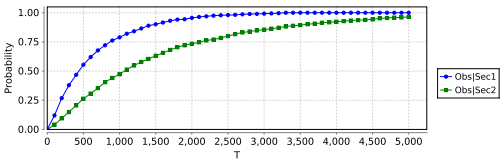


(b) Confidentialité diffrentielle changée

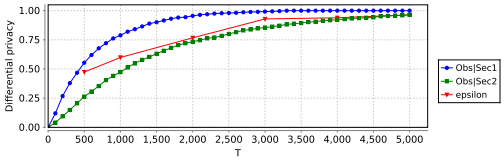
Figure 4.11 Exemple d'exécutions de APMC pour DP changé

Les contrainte en DP est sur les probabilités au pire de cas : $\ln(\frac{Pr[Obs\_sec1]}{Pr[Obs\_sec2]}) \approxeq \varepsilon = ln(2)$ en T=500. Dans les graphiques ci-dessous, nous avons tracé ces valeurs attendues en tant que **epsilon**.

```
label "Obs_Sec1" = RR=4 & count=4 & n=5 & t=4;
label "Obs_Sec2" = RR=4 & count=3 & n=5 & t=4;
```



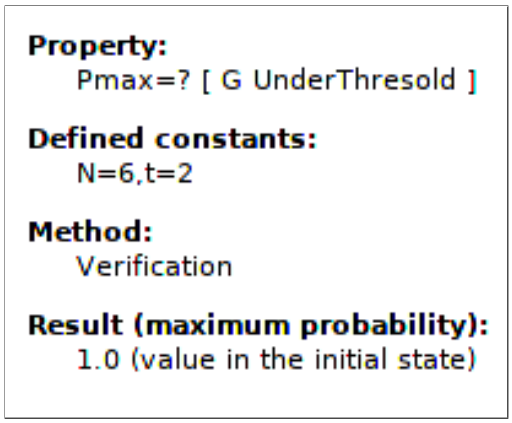
(a) Probabilité d'obtenir des observables



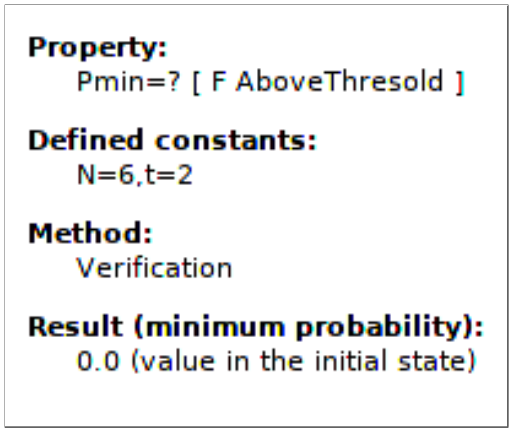
(b) Différence multiplicative dans les distributions probabilistes

Figure 4.12 Exemple d'exécutions de APMC pour DP

Lors de la vérification du modèle, nous avons les exigences suivantes : Sur une base de données de taille n, disons que la précision est non triviale si l'erreur est de racine $\sqrt{n}$, ou bien le bruit ajouté est au maximum $\sqrt{n}$. Les figures suivantes réprésent la résultat de vérification.



(a) Probabilité d'avoir une réponse randomisé sous du seuil de bruit



(b) Probabilité d'avoir une réponse randomisé au-dessus du seuil de bruit

Figure 4.13 Vérificaion de la précision du confidentialité

Dans le graphique ci-dessous 4.14 , nous avons tracé les nombre des requêtes en fonction du taille de la base de données N. Notez que les récompences minimum et maximum attendus sont les mêmes dans chaque cas.

Cost privacy - privacy loss sont des nombres de requêtes qui on peut répondre avec une précision non triviale - trivial respectivement. On peut conclure que le système peut répondre à un très grand nombre de requêtes avec une erreur quasiment nul et confidentialement différentielle.

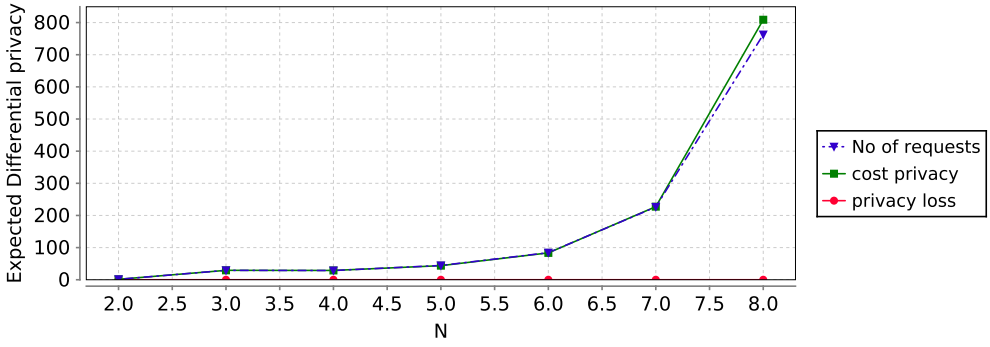


Figure 4.14 Accessibilité attendue de la confidentialité différentielle

4.9 Conclusion

Dans ce chapitre, nous avons proposé un cadre pour vérifier et valider formellement les modèles de la confidentialité différentielle via la vérification et la simulation de modèles probabiliste. L'analyse de la confidentialité différentielle dans les systèmes concurrents est une tâche ardue, le cadre proposé peut être appliqué à n'importe quel modèle de système différentiellement privé. Notre travail est pour démontrer la faisabilité de la vérification probabiliste de modèle (Probabilistic model checker) comme une technique d'analyse de la confidentialité différentielle dans les systèmes concurrents. Nous avons présenté une approche pour la modélisation et la génération automatique des modèles de mécanismes confidentiels et pour des spécifications différents dans l'interaction des composants dans un système différentiellement confidentiel, en montrant comment la technique de vérification de modèle probabiliste peut être utilisée pour vérifier la confidentialité.

Dans ce travail, nous avons développé un modèle de décision markovien à l'aide d'un formalisme PRISM, afin de présenter d'une part l'aspect non déterministe pour modéliser l'interaction entre l'analyste et le système et d'autre part l'aspect probabiliste pour modéliser les réponses randomisées que sont calculé par le mécanisme confidentiel. Nous pensons que les techniques présentées ici s'appliquent plus généralement au système, par exemple pour analyser les différentes approches de gestion budgets de la confidentialité et nous avons aussi démontré l'efficacité du notre frameworke sur étude de cas. Nous montrons la validation de notre framework en prenant l'exemple d'un système que prennent trois composants et en modélisant les mécanismes confidentiels et leurs probabilités de distribution sur des entrées.

Nous avons appliqué la technique de simulation avec succès. Nous avons calculé la probabilité d'accessibilité où il y a toujours des déterministes et des adversaires sans mémoire présentant la récompense cumulative minimale et maximale attendu pour atteindre un coût de la confidentialité. La simulation prend non seulement du temps infini, mais ne garantit pas non plus la qualité des résultats de manière quantitative. Le chapitre suivant se concentre sur la création des algorithmes de vérification de la confidentialité différentielle.

CHAPITRE 5 VERIFICATION SYMBOLIQUE DE LA CONFIDENTIALITÉ DIFFRENTIELLE

5.1 Introduction

Dans le chapitre précédent, nous avons présenté une approche permettant l'analyse formelle de la confidentialité différentielle à l'aide d'un model-checker probabiliste. Nous avons démontré que cette approche fonctionne sur un grand nombre de programmes différentiellement confidentiel, notre framework prend en charge les principes de preuve modulaire, une capacité à décrire et à composer des modules avec hypothèses synchrone ou asynchrone, une capacité à décrire des mécanismes de confidentialité dans les différents niveaux d'abstraction. Cependant, pour obtenir ces résultats, nous avons émis certaines hypothèses sur les paramètres de confidentialité, limitant ainsi la gamme de comportements non déterministes et probabilistes de l'adversaire auxquels nous pouvons faire face dans la pratique. De plus, nos résultats expérimentaux montrent que l'évolutivité de notre approche est en bonne voie.
Dans ce chapitre, nous avons cherché à améliorer encore l'applicabilité et l'évolutivité de notre approche en développant des algorithmes de vérification qui sont basés sur une approche largement utilisée pour lutter contre le problème de l'explosion de l'espace d'état qui est la vérification symbolique, et l'analyse de la confidentialité différentielle devrait être une méthode cohérente et complète.

Nous fournissons la base théorique pour développer des algorithmes et des outils de vérification de la confidentialité différentielle dans les processus probabilistes. Nous présentons un nouvel algorithme symbolique que est basé sur les méthodes de résolution de MTBDD pour les systèmes d'équations linéaires qui représentent les canaux de la confidentialité. Nous définissons aussi la notion de contre-exemple dans le contexte de la confidentialité différentielle. Nous aborderons ensuite une méthode pour générer des contre-exemples utiles pour le débogage du système différentiellement confidentiel.

5.2 Approche pour la vérification de la confidentialité différentielle

5.2.1 Principe

L'approche proposée s'effectue en trois phases :

1. Générations automatique des modèles et la formulation de la propriété de la confidentialité différentielle.
2. Vérification par l'exploration symbolique exhaustive .
3. Analyse des résultats.

Dans un premier temps, le système à étudier est donc modélisé sous forme d'un système probabiliste et la spécification à vérifier sous forme des propriétés *PCTL** et chaque propriété est un observable. Ce système probabiliste peut être représenté par une chaîne de Markov, un processus de décision markovienne ou un automate probabiliste .

Deuxièmement, l'algorithme vérifie si le modèle $\mathcal{M}$ obtenu est différentiellement confidentiel si et seulement si satisfait les propriétés $\Phi = (\varphi_1, ..., \varphi_n)$. Pour cela, il effectue une exploration symbolique exhaustive de l'espace d'états, c'est-à-dire les états d'un système probabiliste accessibles à partir d'un état initial en suivant les deux chemins du modèle simultanément.

Enfin, une analyse des résultats obtenus est effectuée : si $\mathcal{M}$ vérifie $\Phi = (\varphi_1, ..., \varphi_n)$, le modèle du système est déclaré sécurisé (*secure*). Si le modèle ne satisfait pas la propriété ou bien viole la probabilité confidentielle, l'algorithme de vérification retourne (*non-secure*) et des contre-exemples qui est rejoué sur le modèle. Nous avons donc deux possibles : le rejeu du contre-exemple sur le modèle engendre effectivement un trou de sécurité provient de l'abstraction en elle-même, ou bien, la violation de propriété de confidentialité rapportée par l'algorithme n'est présente que sur le modèle si bien qu'aucune erreur n'est soulevée lors du rejeu. Cela signifie alors que le modèle ne préserve pas correctement la confidentialité dans le système, il doit donc être nécessairement faire preuve de raffinement du modèle avant de réaliser une nouvelle vérification.

La figure 5.1 présente notre approche de vérification en détail :

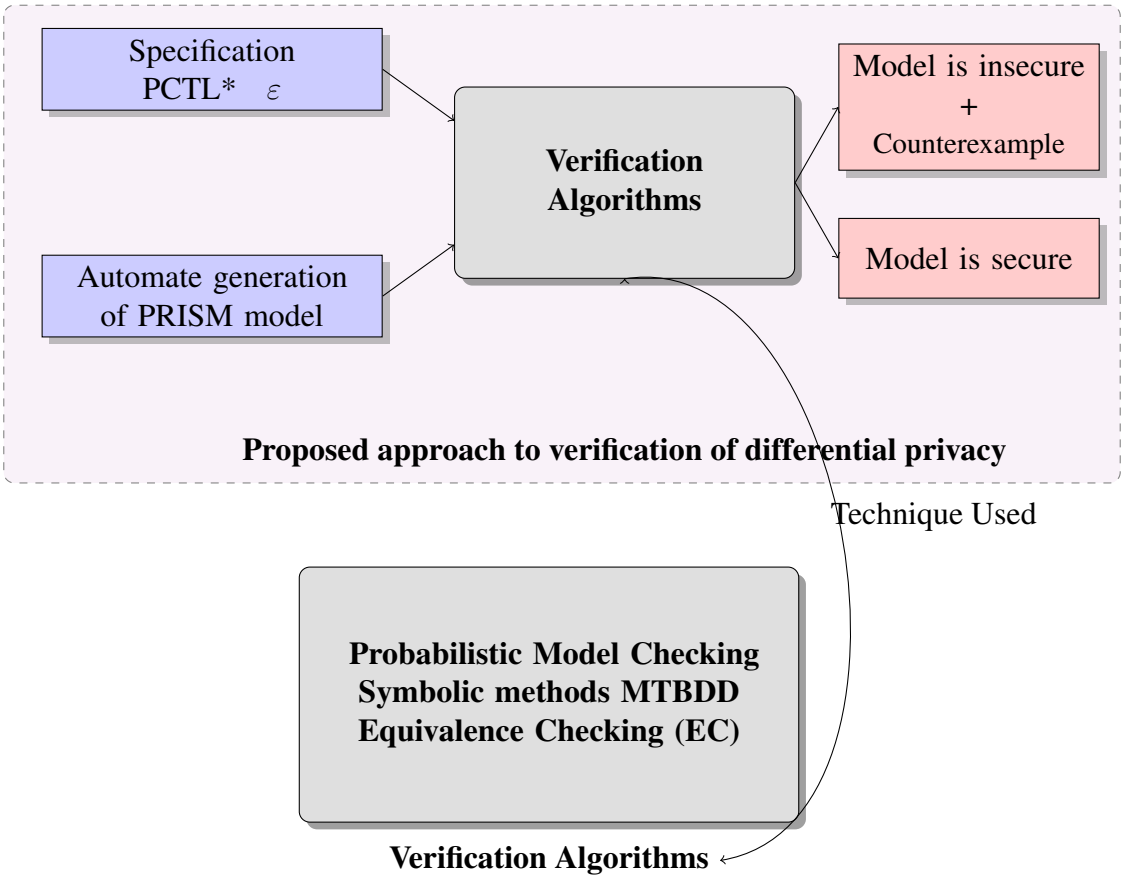


Figure 5.1 Approche proposée pour la vérification de la confidentialité diffrentielle.

5.3 Vérification de la confidentialité confidentielle

Dans cette section, nous décrivons notre approche de vérification de la confidentialité différentielle sur une chaîne de Markov. L'idée principale est que la satisfaction ou la violation de la propriété de la confidentialité peut être démontrée au moyen d'une relation levage *(en anglais lifting)*. Par conséquent, nous introduisons la notion de sémantique opérationnelle de la relaxation cette relation de levage, que nous exploitons des techniques similaires de Model checking probabiliste standard afin de calculer les valeurs de confidentialité.

Matrice des valeurs confidentielles :

L'objectif est de définir une matrice des valeurs privées pour surveiller le budget de confidentialité au niveau des états d'un modèle.
M_ε : est une matrice carrée de taille (n, n) de valeurs confidentielles d'un champ $\mathbb{R}$ avec $n = |S|$. Cette matrice contient principalement trois valeurs : la valeur 0 indique que il n' y pas de perte de budget du confidentialité, $\varepsilon \in \mathbb{R}$ une valeur confidentielle et le symbole ∞ indique que la confidentialité est indéfinie.

$$M_\varepsilon[s,t] = \begin{cases} 0 & si \;\; \mathbf{P}[s,t] > 0 \wedge L(t) \notin Obs \\ \varepsilon & si \;\; \mathbf{P}[s,t] > 0 \wedge L(t) \in Obs \\ \infty & si \;\; \mathbf{P}[s,t] = 0 \end{cases}$$

La relation de lifting :

Nous commençons par définir une opération de *lifting* approximatif qui relève une relation sur des états à une relation sur des distributions probabilistes. Intuitivement, nous utilisons un paramètre pour représenter le budget total de fuite de confidentialité. Ce paramètre se déplaçant sur l'intervalle [0, ε_0] qu'enregistre la quantité actuelle de fuites d'informations au fil du temps à chaque étape de la relaxation.

Nous écrivons pour : $\forall x \in \mathbb{R} \;:\; \mathfrak{L}(x)$ avec la mise à jour de variable x pour contenir une nouvelle valeur Δ, selon le contexte de l'application, il y a plusieurs posibilités des mise à jour des valeurs confidentielles, par exemple en supprimant ou en ajoutant :
La différence absolue des probabilités ($\Delta = |\mathbf{P}[s,t] - \mathbf{P}[s,t']|$), ou un facteur $\Delta = \dfrac{x}{|Sec|}$, dans le cas où, nous avons un problème sur plusieurs itérations dont les distributions probabilistes peuvent évoluer très différemment même sur des secrets voisine.

5.3.1 Relaxation de la relation de lifting

La relaxation de la relation de lifting est une technique efficace pour calculer les valeurs de la confidentialité $\mathcal{E}$ pour un DTMC.
Soit la mesure $\mathcal{E}_i$ d'atteindre les états cibles S^{yes} à partir de l'état $s_i \in S$ est une variable de confidentialité, alors $\mathcal{E}_i$ peut-être calculé de manière récursive comme la solution unique de le système d'équation linéaire suivant :

$$\mathcal{E}(s,\psi) = \begin{cases} \mathfrak{L}(\varepsilon) & si \;\; s \in S^{yes} \\ \mathfrak{L}(0) & si \;\; s \in S^{no} \\ max(\mathfrak{L}(\mathcal{E}(s',\psi))) & sinon \end{cases}$$

Nous prouvons par l'induction réel que $\mathfrak{L}(\varepsilon_i)$ détient tous les nombres réels non négatifs $\varepsilon_i \geq 0$ dans un intervalle de privacité, et chaque ε_i est correspondant à un canal de confidentialité $P^{(i)}$

Corollaire 1. *Soit $\mathfrak{L}$ est vraie on ε_n ssi* $\exists\, n \in \mathbb{N},\ (P^{(n)}[t_j|s_i], P^{(n)}[t_j|s_{i+1}]) \in \mathcal{R}_1^{\mathfrak{L}(\varepsilon_n)}$

(RI1) *$\mathfrak{L}$ est vraie pour $\varepsilon_n = 0$.*

(RI2) *Pour toute $\varepsilon_n > 0$, si $\mathfrak{L}$ est vraie pour tout $\varepsilon_k \in [\varepsilon_n, \varepsilon_0]$ alors $\mathfrak{L}$ est vraie pour tout $\varepsilon_{n+k} \in (0, \varepsilon_n]$ pour $\varepsilon_n < \varepsilon_0$.*

(RI3) *Pour toute $\varepsilon_n > 0$, si $\mathfrak{L}$ est vraie pour tout $\varepsilon_k \in (\varepsilon_n, \varepsilon_0]$ alors $\mathfrak{L}$ est vraie pour ε_n*

Cette corollaire, exprime l'approximation de lifting et nous prouvons leur cohérence et leur complétude par l'induction réel (Voir l'Annexe A.), ce qui nous permet de raisonner sur les comportements différentiellement confidentielle. Le tableau 5.1 montre la matrice de transition d'un DTMC et la matrice des valeurs privées

Tableau 5.1 Les matrices probabilistes pour la confidentialité différentielle

S	s_0	s_1	s_2	s_3	s_4
s_0	0	$1/2$	$1/2$	0	0
s_1	0	0	0	$2/3$	$1/3$
s_2	0	0	0	$1/3$	$2/3$
s_3	1	0	0	0	0
s_4	1	0	0	0	0

(a) $DTMC$

S	s_0	s_1	s_2	s_3	s_4
s_0	∞	0	0	∞	∞
s_1	∞	∞	∞	ε	ε
s_2	∞	∞	∞	ε	ε
s_3	0	∞	∞	∞	∞
s_4	0	∞	∞	∞	∞

(b) ε-DP

5.3.2 Les canaux d'un système confidentiellement privé

A titre d'exemple simple, considérons un fourniseur de données qui choisit toujours d'envoyer à un tel système un point de donnée x_1 ou x_2 avec des probabilités équiprobables $p = q = \frac{1}{2}$, et après avoir reçu un point de donnée $x_i \in X$ par tel système qu'il exécute le mécanisme privé pour afficher la réponse randomisé d'une fonction de comptage dont les valeurs possibles sont $observable1$ et $observable2$. La figure suivante réprésente les canaux d'un système confidentiellement confidentiel.

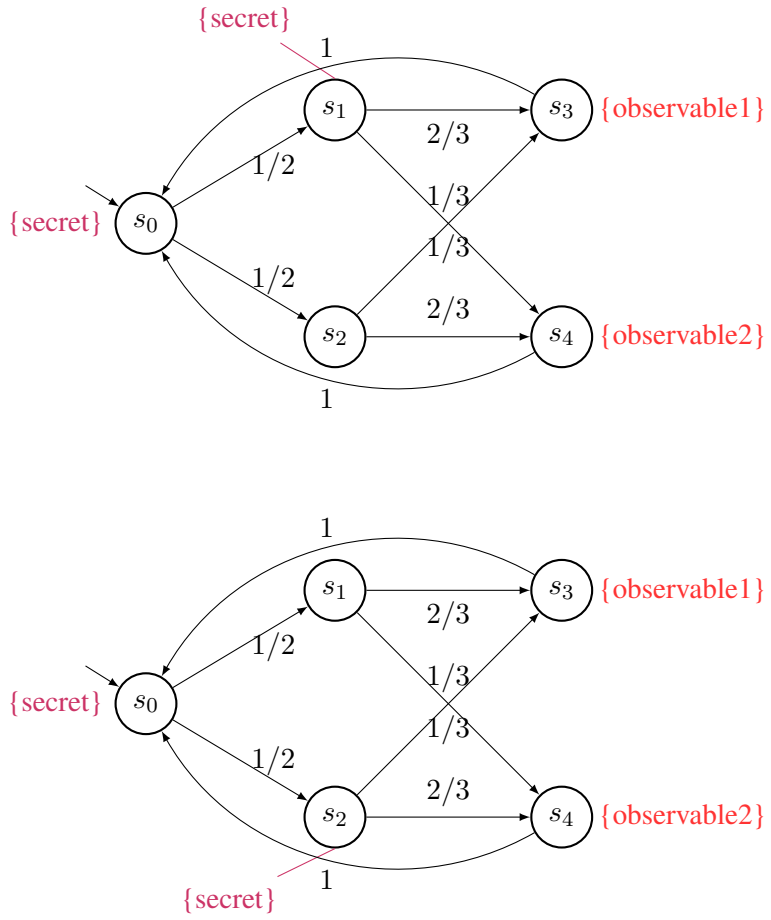


Figure 5.2 Deux chaînes de Markov à temps discret pour la DP

5.3.3 Vérification de la confidentialité différentielle dans les chaînes de Markov :

Nous proposons dans cette section un nouvelle algorithme qui vérifier la confidentialité différentielle dans les chaînes de Markov. Comme nous l'avons vu précédemment dans le chapitre 4, les algorithmes existants ne promettent pas de gérer efficacement cette classe de modèles pour ses valeurs de confidentialité. L'algorithme proposé forme élégamment une fonction par rapport à un paramètre de récurrence pour une propriété particulière d'intérêt comme résultat final. Cet algorithme peut être divisé en trois fonctionnalités principales, comme indiqué dans l'algorithme. Ils sont comme suit :

- **Précalcule :** $PMC(\mathcal{D}_i, \psi)$ est une procédure basée sur le calcul d'un opérateur de point fixe, pour calculer tous les états à partir desquels il est possible, avec une probabilité non nulle atteindre un état satisfaisant ψ. Ce sont les états à partir desquels il existe une probabilité non nulle d'atteindre un état d'observable dans Sat(ψ).
- **Relaxation de la relation du lifting :** $lifting(\mathcal{D}_1, \mathcal{D}_2, \mathsf{D}_\varepsilon, \psi)$ une méthode pour chercher l'existence d'une relation levée approximative de lifting entre deux $DTMCs$
- **Confidentialité différentielle :** Enfin, nous notons que, nous avons deux vecteurs des probabilités où chacun représente des distributions d'une sortie. Ces vecteurs des probabilités doivent toutefois être comparés par rapport la valeur ε adéquate pour garantir que le système est différentiellement confidentiel.

Algorithm 1: Basic formal verification algorithm of differential privacy

Input: $DTMCs\ \mathcal{D}_1$, $\mathcal{D}_2$, property ψ, ε desired privacy

Output: secure or insecure

```
/* Precomputation probabilistic model checking                    */
```

1 $\mathsf{x} \leftarrow PMC(\mathcal{D}_1, \psi)$

2 $\mathsf{y} \leftarrow PMC(\mathcal{D}_2, \psi)$

```
/* Calculating lifting                                            */
```

3 $\mathsf{D}_\varepsilon \leftarrow createDTMC_\varepsilon(\mathcal{D}_1, \mathcal{D}_2, \varepsilon)$

4 $\mathcal{E} \leftarrow lifting(\mathcal{D}_1, \mathcal{D}_2, \mathsf{D}_\varepsilon, \psi)$

```
/* Differential privacy                                           */
```

5 **foreach** $x_i, \in \mathsf{x} \wedge y_i \in \mathsf{y}$ **do**

6 **if** $(x_i, y_i) \in \mathcal{R}_1^{\mathcal{E}_i}$ **then**

7 **if** $|log(x_i) - log(y_i)| \leq \mathcal{E}_i$ **then**

8 **return** secure

9 **else**

10 **return** insecure

5.3.4 Méthode

L'algorithme prend en entrée quatres paramètres qui sont les deux modèles DTMCs à vérifier, la propriété ψ et un paramètre de privacité ε

Étape 1 :

L'algorithme de vérification de modèle probabiliste prend en entrée un DTMC $\mathcal{D}_1$ étiqué et une formule $PCTL*\ \psi$.
$\psi = P =?[secret \bigcup observable1]$

1. Model checker :
   $Sat(secret) = \{s_0, s_1\}, \quad Sat(Observable1) = \{s_3\}$
   $S^{yes} = Sat(P_{\geq 1}[secret \bigcup observable1])) = \{s_3\}$
   $S^{no} = Sat(P_{\leq 0}[secret \bigcup observable1])) = \{s_2, s_4\}$
   $S^{?} = S \setminus (S^{yes} \cup S^{no}) = \{s_0, s_1\}$
2. Solution de systèmes d'équations linéaires :

$$\begin{cases} x_0 = \frac{1}{2}x_1 + \frac{1}{2}x_2 \\ x_1 = \frac{2}{3}x_3 + \frac{1}{3}x_4 \\ x_2 = 0 \\ x_3 = 1 \\ x_4 = 0 \end{cases} \Longrightarrow \begin{cases} x_0 = \frac{1}{2}x_1 \\ x_1 = \frac{2}{3} \\ x_2 = 0 \\ x_3 = 1 \\ x_4 = 0 \end{cases} \Longrightarrow \begin{cases} x_0 = \frac{1}{3} \\ x_1 = \frac{2}{3} \\ x_2 = 0 \\ x_3 = 1 \\ x_4 = 0 \end{cases}$$

3. Résultats :
   $Prob^{\mathcal{D}_1}(secret \bigcup observable1) = \mathsf{x} = [\frac{1}{3}, \frac{2}{3}, 0, 1, 0]$

   Le vecteur x réprésente les probabilités pour chaque $s \in S$, c'est une solution unique d'un programme linéaire. Par exemple x_1 est la probabilité d'attendre S^{yes} à partir s_1

L'algorithme de vérification de modèle probabiliste prend encore une fois en entrée un DTMC $\mathcal{D}_2$ étiqueté et une formule PCTL ψ.
$\psi = P =?[secret \bigcup observable1]$

1. Model checker :
   $Sat(secret) = \{s_0, s_2\}, \quad Sat(Observable1) = \{s_3\}$
   $S^{yes} = Sat(P_{\geq 1}[secret \bigcup observable1])) = \{s_3\}$
   $S^{no} = Sat(P_{\leq 0}[secret \bigcup observable1])) = \{s_1, s_4\}$
   $S^{?} = S \setminus (S^{yes} \cup S^{no}) = \{s_0, s_2\}$
2. Solution de systèmes d'équations linéaires :

$$\begin{cases} y_0 = \frac{1}{2}y_1 + \frac{1}{2}y_2 \\ y_1 = 0 \\ y_2 = \frac{1}{3}y_3 + \frac{2}{3}y_4 \\ y_3 = 1 \\ y_4 = 0 \end{cases} \Longrightarrow \begin{cases} y_0 = \frac{1}{2}y_2 \\ y_1 = 0 \\ y_2 = \frac{1}{3} \\ y_3 = 1 \\ y_4 = 0 \end{cases} \Longrightarrow \begin{cases} y_0 = \frac{1}{6} \\ y_1 = 0 \\ y_2 = \frac{1}{3} \\ y_3 = 1 \\ y_4 = 0 \end{cases}$$

3. Résultats :
   $Prob^{\mathcal{D}_2}(secret \bigcup observable1) = \mathsf{y} = [\frac{1}{6}, 0, \frac{1}{3}, 1, 0]$

Étape 2 :

Création une relation de la confidentialité ε-*DP* sous forme d'une matrice (voir le tableu 5.1b) pour calculer toutes les valeurs possibles du paramètre de privacité $\mathcal{E}$ en fonction du coût de la requête ε et le nombre de secrets est égale n, telle que la matrice D_ε contient trois valeurs : 0 , $\varepsilon \backsimeq \frac{\varepsilon_1}{n}$, et ∞ pour indiquer que la confidentialité différentielle est indéfini.
Par exemple si le nombre de secret est égale $|Sec| = n = 2$ avec $\varepsilon_1 = ln(2)$ et $\varepsilon = \frac{ln(2)}{2}$, toutes les valeurs possibles pour le paramètre de privacité sont :
$\mathcal{E} = (\varepsilon_0, \varepsilon_1, \varepsilon_2, \varepsilon_3) \approx \mathcal{E} = (1.386, 0.693, 0.346, 0)$ et ε_i indique la valeur de privacité dans la $i^{\text{ème}}$ requête. Après on calcule les valeurs $\varepsilon_i \in \mathcal{E}$ par une technique de la relaxation de la relation de lifting qu'est représenté par une équation récursive dans 5.3.1.

Relaxation de la relation de lifting pour le chemin $\rho_1 \models \psi$ dans $\mathcal{D}_1$

$$\begin{cases} \mathcal{E}_0 = max(\mathfrak{L}(\mathcal{E}_1), \mathfrak{L}(\mathcal{E}_2)) \\ \mathcal{E}_1 = max(\mathfrak{L}(\mathcal{E}_3), \mathfrak{L}(\mathcal{E}_4)) \\ \mathcal{E}_2 = \mathfrak{L}(0) \\ \mathcal{E}_3 = \mathfrak{L}(\varepsilon) \\ \mathcal{E}_4 = \mathfrak{L}(0) \end{cases} \Longrightarrow \begin{cases} \mathcal{E}_0 = max(\mathfrak{L}(\mathcal{E}_1), \mathfrak{L}(\varepsilon_0)) \\ \mathcal{E}_1 = max(\mathfrak{L}(\varepsilon_1), \mathfrak{L}(\varepsilon_0)) \\ \mathcal{E}_2 = \varepsilon_0 \\ \mathcal{E}_3 = \varepsilon_1 \\ \mathcal{E}_4 = \varepsilon_0 \end{cases} \Longrightarrow \begin{cases} \mathcal{E}_0 = max(max(\varepsilon_2, \varepsilon_1), \varepsilon_1) \\ \mathcal{E}_1 = max(\varepsilon_2, \varepsilon_1) \\ \mathcal{E}_2 = \varepsilon_0 \\ \mathcal{E}_3 = \varepsilon_1 \\ \mathcal{E}_4 = \varepsilon_0 \end{cases}$$

Résultat : $\mathcal{E}^{\mathcal{D}_1}(s, \psi) = [\varepsilon_1, \varepsilon_1, \varepsilon_0, \varepsilon_1, \varepsilon_0]$

Relaxation de la relation de lifting pour le chemin $\rho_2 \models \psi$ dans $\mathcal{D}_2$

$$\begin{cases} \mathcal{E}_0 = max(\mathfrak{L}(\mathcal{E}_1), \mathfrak{L}(\mathcal{E}_2)) \\ \mathcal{E}_1 = \mathfrak{L}(0) \\ \mathcal{E}_2 = max(\mathfrak{L}(\mathcal{E}_3), \mathfrak{L}(\mathcal{E}_4)) \\ \mathcal{E}_3 = \mathfrak{L}(\varepsilon) \\ \mathcal{E}_4 = \mathfrak{L}(0) \end{cases} \Longrightarrow \begin{cases} \mathcal{E}_0 = max(\mathfrak{L}(\varepsilon_0), \mathfrak{L}(\mathcal{E}_2)) \\ \mathcal{E}_1 = \varepsilon_0 \\ \mathcal{E}_2 = max(\mathfrak{L}(\varepsilon_1), \mathfrak{L}(\varepsilon_0)) \\ \mathcal{E}_3 = \varepsilon_1 \\ \mathcal{E}_4 = \varepsilon_0 \end{cases} \Longrightarrow \begin{cases} \mathcal{E}_0 = max(\varepsilon_1, max(\varepsilon_2, \varepsilon_1)) \\ \mathcal{E}_1 = \varepsilon_0 \\ \mathcal{E}_2 = max(\varepsilon_2, \varepsilon_1) \\ \mathcal{E}_3 = \varepsilon_1 \\ \mathcal{E}_4 = \varepsilon_0 \end{cases}$$

Résultat : $\mathcal{E}^{\mathcal{D}_2}(s, \psi) = [\varepsilon_1, \varepsilon_0, \varepsilon_1, \varepsilon_1, \varepsilon_0]$
Alors le résultat final des valeurs de privacité est obtenue à partir une méthode qui présente une convergence plus rapide en utilisant le principe de la relaxation de la relation de lifting de la confidentialité différentielle que calculent des valeurs de ε disponibles pour chaque état dans chaque itération du calcul ;

$\mathcal{E}(s, \psi) = (\mathcal{E}^{\mathcal{D}_1} \equiv \mathcal{E}^{\mathcal{D}_2})$
$\mathcal{E}(s, \psi) = [\varepsilon_1, inf, inf, \varepsilon_1, \varepsilon_0]$

Étape 3 :

Alors nous avons : $Prob^{\mathcal{D}_1}(\psi) = \mathsf{x} = [\frac{1}{3}, \frac{2}{3}, 0, 1, 0] \quad Prob^{\mathcal{D}_2}(\psi) = \mathsf{y} = [\frac{1}{6}, 0, \frac{1}{3}, 1, 0]$

$\mathcal{E}(s, \psi) = [\varepsilon_1, inf, inf, \varepsilon_1, \varepsilon_0]$ Et comme : $\forall \mathsf{x}_i \in Prob^{\mathcal{D}_1}, \mathsf{y}_i \in Prob^{\mathcal{D}_2} \;\; |log(\mathsf{x}_i) - log(\mathsf{y}_i)| \leq \varepsilon_1$

Conclusion : le système est différentiellement privé.

5.4 Vérification symbolique de la confidentialité différentielle

Dans cette section, nous introduisons un algorithme de vérification de modèle symbolique de la confidentialité différentielle sur des processus de décision Markov étiquetée. La vérification de modèle probabiliste pour les logiques probabilistes implique généralement la résolution de systèmes d'équations linéaires afin de déterminer la probabilité de maintien d'une formule donnée dans un état. Notre algorithme repose sur l'idée de représenter les matrices utilisées dans les systèmes d'équations linéaires par le diagramme de décision binaires multiterminal (MTBDD) pour calculer la matrice de canal de probabilités $\mathsf{X}_{\mathsf{n,m}}$ et chaque élément $\mathsf{X}_{\mathsf{i,j}}$ est un MTBDD représente les probabilités d'attendre l'observable i à partir d'un état du secret dans MDP $\mathcal{M}$ sous l'adversaire σ_j. Notre algorithme basé sur les procédures développé dans ces travaux [39] [40], utilise des BDDs pour représenter des formules Ψ qui sont des observables et des MTBDDs représentant le MDPs et d'autre MTBDDs représentant les valeurs de confidentialité $\mathcal{E}_i$. Cet algorithme est efficace, car il évite la construction d'espace d'état explicite, et représente et manipule efficacement des canaux différentiellement confidentiels.

5.4.1 L'equivalent checking pour DP

La confidentialité différentielle peut être formalisée en considérant des paires de secrets ou des bases de données adjacents et en comparant les distributions probabilistes obtenues après l'exécution du programme probabiliste sur elles, ces distributions soient indiscernables par un facteur ε. Cette définition nous ramène à établir une approche sémantique pour construire une équivalence de vérification probabiliste : l'indiscernabilité de deux d'exécutions d'un programme probabiliste sur deux secrets différés par un seul entrés. Nous avons utilisé la structure MTBDDs et leur propriété pour vérifier l'équivalence des programmes différentiellement confidentiels comme des MTBDDs.

5.4.2 MTBDD pour DP

Dans cette sous-section nous examinons le problème de la représentation des modèles de la confidentialité différentielle en tant que MTBDDs. Nous avons deux types de MTBDDs sont tous les deux décrits par des matrices à valeurs réelles. L'idée de base est qu'une matrice peut être conçue comme une fonction mappant des paires d'indices en nombres réels. Étant donné le codage de ces indices dans des variables booléennes, nous pouvons plutôt voir la matrice comme une fonction mappant des variables booléennes à des nombres réels qui sont des valeurs ε_i, ce qui est exactement ce que représente un MTBDD.
Le premier type de MTBDD est la représentation de MDP $\mathcal{M}$, le deuxième type de MTBDD correspond à l'évaluation des valeurs appropriées du paramètre ε_i en confidentialité différentielle.

Nous devons calculer, pour chaque paire possible de (secret,observable), la probabilité que le programme différentiellement privé produise $observable_j$ lorsque $secret_i$. Habituellement, ces quantités sont stockées sous forme de matrice de canaux, également appelés matrice de canaux différentiellement privés. Nous calculons la matrice de canal dont chaque case où il y a une solution possible à un système d'équations linéaires d'un MDP sous l'adversaire σ_i ce qui peut être fait à l'aide d'un MTBDD.

Représentation le processus de décision de Markov par les MTBDD

Pour représenter la matrice de transition de MDP $M = (S, \bar{s}, A, \delta, L)$ étiqueté par un MTBDD on fait l'abstraction des noms des états et des actions de manière similaire à [40], en utilisant des tuples binaires de propositions atomiques qui sont vraies dans l'état. D'abord, les éléments de tout ensemble fini S peuvent être codés par des vecteurs booléens de longueur $\lceil log_2|S| \rceil$. Par exemple, les états de l'ensemble $S = \{s_0, s_1, s_2, s_3\}$ peuvent être codés de manière évidente par des chaînes de bits de longueur deux .
Pour les actions, on supposons que nous avons un ensemble d'actions donné par
$A = \{in_H, in_L, out_H, out_L\}$. nous pouvons coder l'action $a \in A$ comme chaîne de bits (voir le tableau ce dessous), nous écrivons $\{\xi(in_H) = \{00\}, \xi(in_L) = \{01\}, \xi(out_H) = \{10\}, \xi(out_L) = \{11\}\}$. Si nous utilisons les variables booléennes a_1 et a_2 pour caractériser les deux positions d'une telle chaîne de bits, alors le terme $\bar{a_1}\bar{a_2} = 00$ correspond à l'action in_H, le terme $\bar{a_1}a_2 = 01$ correspond à l'action in_L, le terme $a_1\bar{a_2} = 10$ correspond à l'action out_H et le terme $a_1a_2 = 11$ correspond à l'action out_L.

Nous fixons une énumération $\{a_1, ..., a_n\}$ des propositions atomiques et identifie chaque état $s \in S$ avec le $n-$tuple booléen $\xi(s) = (b_1, ..., b_n)$ où $b_i = 1$ ssi $a_1 \in L(s)$.

Pour représenter une transition, il est nécessaire de coder au moins l'état source et l'état cible d'une transition particulière. Nous allons utiliser les variables booléennes $s_1, ..., s_n$ pour coder l'état source, et $t_1, ..., t_n$ pour encoder l'état cible (en anglais *target*).

Tableau 5.2 Encodage des actions et des états

états	encodage
s_0	00
s_1	01
s_2	10
s_3	11

(a) Les encodages d'état

action	encodage
in_H	00
in_L	01
out_H	10
out_L	11

(b) Les encodages d'action

Pour les choix non déterministe, nous avons des variables supplémentaires c_i pour réprésenter ces choix entre les transitions. Le classement global des variables est ordonnées $\prec$.

$$c_1 \prec ... \prec c_{n_c} \prec a_1 \prec ... \prec a_{n_a} \prec s_1 \prec t_1 \prec ... \prec s_{n_s} \prec t_{n_s}$$

Nous donnons maintenant une définition formelle de la correspondance de la MTBDD pour la confidentialité diffrentielle :

Définition 13. *Étant donnée un MDP $\mathcal{M} = (S, \bar{s}, A, \delta, L)$, et $\mathsf{M} \triangleright \mathcal{M}$, M_ε et $\mathcal{E}^j$ sont ε-MTBDD avec $E \subset S$, $j \in [0, n]$ et n est le nombre des observables possibles.*

M_ε représente ε-DP si et seulement si

- *Une fonction injective $\xi_E : E \mapsto \mathbb{B}^{n_E}$ pour certains $n_E \leq n_S$ telle que $L(e \in E) = \{secret\}$ de telle sorte que pour toutes les $in \in A_L$ les conditions suivantes sont remplies :*

  1. *Si $s \xrightarrow{in} s'$ alors il existe $\underline{c} \in \mathbb{B}^{n_c}$, tel que pour tous $s' \in S$:*
  $$f_{\mathsf{M}_\varepsilon}(\underline{c}, \xi_A(in), \xi_E(e) \bowtie \xi_S(s')) = \varepsilon_c$$
  2. *Si $s \xrightarrow{a} \delta$ alors il existe $\underline{c} \in \mathbb{B}^{n_c}$, tel que $f_{\mathsf{M}}(\underline{c}, \xi_A(a), \xi_S(s) \bowtie \xi_S(s')) \neq 0$ pour tous $s' \in S$ et $\delta \in Dist(S)$:*
  $$f_{\mathsf{M}_\varepsilon}(\underline{c}, \xi_A(a), \xi_S(s) \bowtie \xi_S(s')) = 0$$

$\mathcal{E}^j$ représente la relation de lifting en ε-DP si et seulement si

- *Pour tout $\underline{c} \in \mathbb{B}^{n_c}$, il existe $s \xrightarrow{in} s'$ pour tous $s' \in S$:*
$$f_{\mathcal{E}^j}(\underline{c}, \xi_A(a), \xi_S(s) \bowtie \xi_S(s')) = \mathfrak{L}(\varepsilon_c)$$

Pour illustrer cette définition, la figure ci-dessous 5.3 montre un exemple de la manière dont la confidentialité différentielle peut être représenté par un MTBDD .

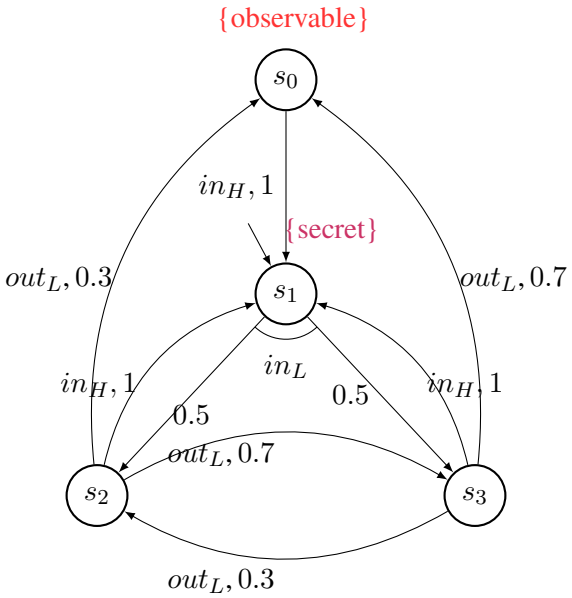


Figure 5.3 Exemple d'un processus de décision markovien pour DP

Exemple de motivation :

La figure 5.3 présente un modèle d'une application diffrentiellement privé similaire au programme $Apple_{DP}$ [1]. Cette application utilise une méthode d'injection de bruit similaire à celle utilisée dans la $Apple_{DP}$, mais avec une différence importante : elle applique des types d'opérations de base que sont la permutation et le tri, avant d'effectuer l'étape de privatisation. Cela garantit que l'analyse des données collectées ne peut pas distinguer les valeurs réelles des valeurs trié,

Nous supposons qu'il y a une vecteur $I = [-1, 0, 1]$ avec $I \subset \mathbb{Z}$ que sert d'un tableau de fréquence des événements d'un utilisateur dans un flux de données.

L'éspace d'états $S = \{s_0, s_1, s_2, s_3\}$ est définie comme suit : $s_0 : I = [-1, 0, 1]$, $s_1 : I = [1, 0, -1]$, $s_2 : I = [0, 1, -1]$ et $s_3 : I = [1, -1, 0]$

$\delta(s, in_L)(t)$: une permutation de deux cases avec une probabilité de p.
$\delta(s, out_L)(t)$: un décalage circulaire de droite à gauche ou gauche à droite selon l'état suivante t et la probabilité p. $\delta(s, in_H)(t)$: Trier dans l'ordre décroissant.

1. Les informations originales d'un utilisateur par l'application $Apple_{DP}$ sont codées à l'aide d'une série de fonctions de hachage et stocké dans une matrice. Chaque coordonnée du matrice est ensuite inversée avec une probabilité de $\frac{1}{1+e^{(\varepsilon/2)}}$, où ε est le paramètre de confidentialité.

Tableau 5.3 Les matrices probabilistes pour la confidentialité différentielle

	00	01	10	11
00	0	1	0	0
01	0	0	0.5	0.5
10	0	1	0	0
10	0.3	0	0	0.7
11	0	1	0	0
11	0.7	0	0.3	0

(a) MDP

	00	01	10	11
00	∞	0	∞	∞
01	∞	∞	ε	ε'
10	∞	0	∞	∞
10	0	∞	∞	0
11	∞	0	∞	∞
11	0	∞	0	∞

(b) ε-MDP

Les fonctions f_{M} et $f_{\mathsf{M}_\varepsilon}$ sont donnés par :

$$f_{\mathsf{M}}(\mathsf{v}_1,\mathsf{v}_2,\mathsf{v}_3,\mathsf{v}_4,\mathsf{v}_5,\mathsf{v}_6,\mathsf{v}_7) = \begin{cases} 1 & : si \;\; \mathsf{v}_1,..,\mathsf{v}_7 \in \{0000001, 0001001, 1001001, 0001011, 1001011\} \\ 0.7 & : si \;\; \mathsf{v}_1,..,\mathsf{v}_7 \in \{0111101, 1111101, 0111010, 1111010\} \\ 0.5 & : si \;\; \mathsf{v}_1,..,\mathsf{v}_7 \in \{0010110, 0010111\} \\ 0.3 & : si \;\; \mathsf{v}_1,..,\mathsf{v}_7 \in \{0111000, 0010111, 0111110, 1111110\} \\ 0 & : sinon \end{cases}$$

$$f_{\mathsf{M}_\varepsilon}(\mathsf{v}_1,\mathsf{v}_2,\mathsf{v}_3,\mathsf{v}_4,\mathsf{v}_5,\mathsf{v}_6,\mathsf{v}_7) = \begin{cases} \varepsilon & : si \;\; \mathsf{v}_1,..,\mathsf{v}_7 = 0010110 \\ \varepsilon' & : si \;\; \mathsf{v}_1,..,\mathsf{v}_7 = 0010111 \\ 0 & : si \;\; \mathsf{v}_1,..,\mathsf{v}_7 \in \{0000001, 0001001, 1001001, 0001011, 1001011, \\ & \qquad\qquad 0111101, 1111101, 0111010, 1111010, \\ & \qquad\qquad 0111000, 0010111, 0111110, 1111110\} \\ \infty & : sinon \end{cases}$$

Les figure suivantes montrent des MTBDDs M et M_ε sur $(\mathsf{v}_1,\mathsf{v}_2,\mathsf{v}_3,\mathsf{v}_4,\mathsf{v}_5,\mathsf{v}_6,\mathsf{v}_7)$ pour les fonctions f_{M} et $f_{\mathsf{M}_\varepsilon}$ qu'ils représentent. Dans la représentation graphique, les sommets non terminaux sont regroupés en sept niveaux, et tous les sommets du même niveau sont supposés être étiquetés avec les variables booléennes codant le choix non-déterminisme, noms des actions, les états source et les états cible, indiquée à gauche.

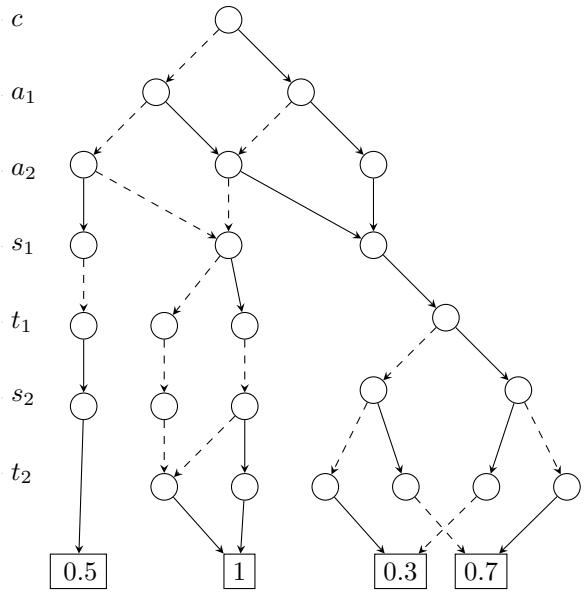


Figure 5.4 MTBDD pour un modèle DP .

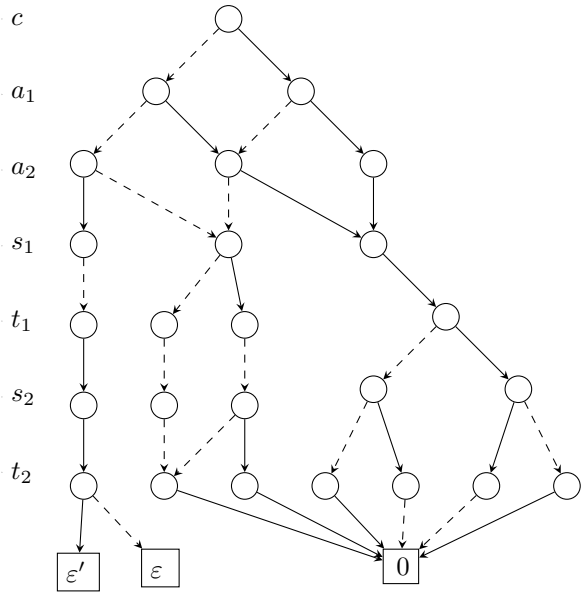


Figure 5.5 ε-MTBDD pour un modèle DP .

La vérification de la confidentialité diffrentielle dans les modèles probabilistes peut-être réduit au problème plus simple de la résolution d'un système d'inéquations linéaires (inéquation matricielle $\mathbf{x_{i,j}x^{-1}_{k,j}} \leq \mathcal{E}_j$), avec $\mathbf{x_{i,j}x^{-1}_{k,j}}$ sont deux secrets et $\mathcal{E}_j$ sont les valeurs de privacité, que sont répresentent par des matrices d'itération.

Notre approche symbolique de vérification repose sur le lemme suivant :

Lemme 3. *Soit $\mathfrak{M}$ une matrice de canaux de dimensions $n \times m$, $\mathfrak{M}$ satisfait à la confidentialité différentielle ssi les systèmes d'inéquations linéaires satisfaisant les conditions suivantes :*

$$\forall \boldsymbol{i} \in [\mathbf{1}, \boldsymbol{m}]\ \forall \boldsymbol{j} \in [\mathbf{1}, \boldsymbol{n}]\ \forall \boldsymbol{k} \in [\mathbf{1}, \boldsymbol{m}]$$
$$\begin{cases} \mathbf{A}_j \mathbf{x_{i,j}} = \mathbf{b}_{i,j} \\ \mathbf{A}_j \mathbf{x_{k,j}} = \mathbf{b}_{k,j} \\ \mathbf{x_{i,j}x^{-1}_{k,j}} \leq \mathcal{E}_j \end{cases}$$

avec n est le nombre des observables possibles et $m = |S|^c$ nombre des DTMCs induit

Ce lemme garantit la confidentialité différentielle pour chaque adversaire $\sigma \in Adv$:

Tableau 5.4 La matrice de canaux d'un système différentiellement confidentiel

$\mathcal{D}^{Adv} \backslash \Psi$	ψ_1	ψ_2	...	...	ψ_n
$\mathcal{D}^{\sigma_1}$	$\mathsf{X}_{1,1}$	$\mathsf{X}_{1,2}$	...	...	$\mathsf{X}_{1,n}$
$\mathcal{D}^{\sigma_2}$	$\mathsf{X}_{2,1}$	$\mathsf{X}_{2,2}$	...	...	$\mathsf{X}_{2,n}$
...	...	...	...	...	...
...	...	...	...	...	...
$\mathcal{D}^{\sigma_m}$	$\mathsf{X}_{m,1}$	$\mathsf{X}_{m,2}$	...	...	$\mathsf{X}_{m,n}$

ε	$\mathcal{E}_1$	$\mathcal{E}_2$	...	...	$\mathcal{E}_n$

Le tableau 5.4 réprésente une matrice de canaux d'un système différentiellement confidentiel, où ψ_i est une proprété pour un observable (i), $\mathcal{D}^{\sigma_j}$ un DTMC induit sous un j-ième-adversaire σ_j et le MTBDD $\mathsf{X}_{i,j}$ est une solution du système d'équations linéaires $\mathbf{A}_j \mathbf{x_{i,j}} = \mathbf{b}_{i,j}$. La vecteur des valeurs de privacité que correspondant à cette matrice réprésente les MTBDDs $\mathcal{E}$ pour chaque case $\mathcal{E}_i$.

5.4.3 Méthode symbolique

L'exigence de notre algorithme de vérification de DP est que les entrées et les sorties puissent être représentées comme des (MT)BDD.

1. **Procédures de pré-calcul pour calculer les MTBDDs et BDDs :**
   La première étape (lignes 1 à 5), nous construire le MTBDD M, qui représente la matrice de probabilité de transition du MDP, puis nous utilisons des procédures de précalcul pour construire les BDDs $\mathsf{B_i}$, $\mathsf{B_{i!}}$, et $\mathsf{B_{i?}}$,qui représentents les ensembles S^{no}, S^{yes} et $S^{?}$ respectivement. En répétant ce processus pour calculer tous les BDDs qui représentents les ensembles d'états qui satisfont ψ_i pour chaque observables i. Ensuite (la ligne 6), est pour calculer le MTBDD représentant A, nous filtrons les états que nous n'avons pas besoin de considérer (ces états qui ne sont pas dans $S^{?}$). Ces procédures sont décrites dans l'annexe.
2. **Calcul du canal de DP par la résolution des systèmes d'inéquations linéaires :**
   Nous avons développé une procédure « *generateAllAdversaries* » : pour générer tous les adversaires (déterministes sans mémoire [2]) du modèle MDP, c'est une méthode polynomial pour énumérer toutes les combinaisons de toutes les actions dans tous les états, en itérant récursivement les états du MDP et leurs choix. Nous modifions la méthode itérative « $IterativeOptimize$ » de calcul de la solution $A_j x_{i,j} \leq b_j$ que sont des probabilités pour chaque propréité ψ_j, de telle sorte que nous utilisons à chaque fois un σ_i parmi toute les adversaires possible, et toutes ces solutions construisent un canal différentiellement privé.

3. **Vérification de la confidentialité différentielle :**
   Premièrement, nous construire un MTBDD M_ε pour DP, puis nous utilisant notre algorithme « $lifting$ » pour construire une solution unique des valeurs de privacités $\mathcal{E}_i$ pour chaque formule ψ_i en utilisant la méthode de la relaxation de la relation de lifting (voir la section 5.3.1)avec une simple variation au niveau des paramètres pour exprimer σ_i. Deuxièmement, (la ligne 14-20) on explore tous les cases de la matrice pour vérifier la conditions DP. $\mathsf{X_{DP}}$ est une MTBDD représentant les rapports entre les probabilités. $\mathsf{M_{DP}}$ est une BDD représentant l'ensemble d'états satisfaisant la confidentialité différentielle est donné par la méthode « $THRESHOLDS$ ». $\mathsf{M_{DP}}$ est une BDD représentant l'ensemble d'états satisfaisant la confidentialité différentielle est donné par la méthode « $THRESHOLDS$ ». La méthode « $Value$ » renvoie la constante BDD avec la valeur $x \in \mathbb{B}$, c'est-à-dire la BDD consistant en un seul sommet terminal v avec la valeur x.

2. Le comportement d'un MDP M avec les états S sous un adversaire déterministe sans mémoire σ peut être représenté par un DTMC avec les états S dans lequel chaque $s \in S$ ne contient que le choix fait par σ dans s.

La méthode « $Value$ » renvoie une constante BDD avec la valeur $x \in \mathbb{B}$, c'est-à-dire la BDD consistant en un seul sommet terminal v avec la valeur x.

Pour calculer le rapport entre les probabilités, nous avons besoin d'une nouvelle définition auxiliaire :

Définition 14. *Étant donné* $\star : \mathbb{R}^+ \times \mathbb{R}^+ \to \mathbb{R}^+$ *un opérateur binaire défini comme* $r_1 \star r_2 = log|r_1| - log|r_2|$

Algorithm 2: Symbolic verification of Differential privacy

Input: MDP $\mathcal{M}$, $\Psi = (\psi_1, ..., \psi_n)$ and ε desired privacy

Output: secure or insecure

```
/* Precomputation procedures to compute MTBDD and BDDs                    */
M ← computeMTBDD(M)
foreach ψ_i ∈ Ψ do
    B_i ← computeBDD(ψ_i)
    B_i! ← computeBDD(¬ψ_i)
    B_i? ← Not(APPLY(∧, B_i, B_i!))
    A_i ← APPLY(M, B_i?, ×)
/* Calculating the canal of DP by resolution the solutions of A_j x_{i,j} ≤ b_j
   */
Adv ← generateAllAdversaries(M)
foreach σ_i ∈ Adv do
    foreach ψ_j ∈ Ψ do
        X_{i,j} ← IterativeOptimize(A_j, B_j, ⩽, σ_i)
/* Differential privacy                                                   */
M_ε ← computeMTBDD(M, ε)
foreach ψ_i ∈ Ψ do
    E_i ← lifting(M_ε, B_i)
for j := 1 to |X_{*,j}| do
    for i := 1 to |X_{i,*}| do
        for k := i to |X_{i,*}| do
            X_DP ← APPLY(X_{i,j}, X_{k,j}, ⋆)
            M_DP ← THRESHOLDS(X_DP, E_j, ≤)
            if CONST(M_DP) = false ∨ Value(M_DP) = 0 then
                return insecure
return secure
```

5.5 Génération de contre-exemples pour la vérification de la confidentialité différentielle

Dans cette section, nous présentons une nouvelle technique pour générer des contre-exemples dans les modèles des chaînes de Markov différentiellement confidentielle. Ces contre-exemples basés sur les chemins et ses relations de déroulement. Les chemins des contre-exemples sont regroupés dans des couples susceptibles de fournir des informations de débogage similaires à l'utilisateur pour détecter les trous de sécurité.

Il semble naturel de fournir des contre-exemples non pas au niveau de l'espace d'état du système, mais au niveau de privacité fixe, pour les garanties de confidentialité (ε). D'une part, cela est dû au fait qu'un concepteur de système aime être signalé aux erreurs à ce niveau. D'un autre côté, bien que les descriptions des modèles probabilistes, elles peuvent entraîner des millions d'états où même des contre-exemples sont incompréhensibles. L'idée de cette technique est de calculer automatiquement des ensembles des chemins qui sont pertinents pour la violation d'une propriété.

Il n'est généralement pas anodin de trouver des traces de cette violation. Dans cette section, nous discutons d'une approche permettant de générer de tels contre-exemples en utilisant les résultats de l'analyse de déroulement par la vérification de modèle basé sur le déroulage et le test de non-interférence [41] [42].

5.5.1 Notion de contre-example en confidentialité différentielle :

Dans cette section, nous définissons ce que sont les contre-exemples en confidentialité différentielle et comment le problème de la recherche de contre-exemples sur des chaînes de Markov.

Définition 15. *Un contre-exemple en confidentialité diffrentielle implique de trouver deux entrées adjacentes DTMCs* $(\mathcal{D}_1, \mathcal{D}_2)$*, un mauvais output* ψ *et un paramètre non privé* ε *telle que* $\frac{\Pr_s^{\mathcal{D}_1}(\psi)}{\Pr_s^{\mathcal{D}_2}(\psi)} > exp(\varepsilon)$.

Pour calculer ces contre-exemples, en utilisent un algorithme que construit une relation entre les chemins et vérifie la cohérence des états observables associés, autrement dit la vérification leur cohérence avec les distributions probabilistes des sorties.

Génération de chemins :

Pour commencer, notons que s'il n'est pas possible de construire des relations de déroulement, parce que ces relations peuvent être très grandes, voire infinies. Selon notre représentation de contre-exemple comme une relation, nous voulions explorer les possibilités de trouver et calculer une paire de chemins qui viole une certaine propriété pour une observable, autrement dit, il existe

au moins une paire de chemins (ρ_1,ρ_2) tels que, pour certains secrets , les observables spécifiées par PCTL* sont incohérentes avec le paramètre ε. Nous voulons profiter de ce fait pour produire deux chemins équivalents en entrée, mais telles qu'elles ne soient pas équivalentes en sortie pour ρ_1 et ρ_2.

Chemin de lifting :

Nous fixons d'abord l'ensemble des chemins qui peuvent faire partie d'un contre-exemple, soi-disant des preuves de la violation d'un état et le valeur ε associée. Ceci nous mène vers la définition suivante.

Définition 16.
Soit une chaîne de Markov $\mathcal{D}$:

- *Un chemin de lifting infini en $\mathcal{D}$ est une suite infinie $(s_0, \varepsilon_0)(s_1, \varepsilon_0)...(s_n, \varepsilon_k)$ des couple tels que : $\forall i > 0$: $\boldsymbol{P}(s_i, s_{i+1}, \varepsilon_j) > 0 \wedge \varepsilon_j > 0$.*
- *Un chemin de lifting fini est un préfixe fini d'un chemin de lifting infini.*

Détection de la violation de la confidentialité différentielle :

Sur la base de la définition décrite ci-dessus 16, nous avons développé une nouvelle méthode pour générer des contre-exemples sur des DTMCs qui violent une propriété pour un observable. Nous avons façonné la notion relation de déroulement entre les DTMCs qui sont des programmes différentiellement confidentiels non sécurisés où la propriété est également violée. De plus, ces relations peuvent être vues comme une représentation d'un ensemble de tous les couples de chemins menant de l'état initial à l'un des états cibles (observables) ont même violation de la confidentialité différentielle, cet ensemble forme un contre-exemple.

Notre méthodologie est la suivante : premièrement, on calcule un ensemble des chemins : comme l'approche de vérification du modèle probabiliste borné elle-même trouve des chemins arbitraires menant d'un état initial à l'un des états cibles, les probabilités sont ignorées. Il se peut donc que de nombreux chemins de faibles probabilités soient trouvés alors que cela conduirait à une terminaison plus précoce. Cela nous a donné l'occasion de calculer la probabilité d'un chemin, mais en fait d'encoder le calcul de probabilité par une formule ψ est satisfaite si et seulement si le chemin correspondant a au moins une certaine probabilité. C'est une façon de vérifier le modèle borné conduit à moins d'itérations de recherche dans la plupart des cas. Deuxiement, on calcule une relation des déroulements des chemins : deux chemins ρ_1 et ρ_2 sont en relation s'ils sont les états cibles de deux chemins sont en cohérence par un facteur ε, et finalement on recherche d'une paire de chemins impliquera une violation de la confidentialité différentielle.

Exemple :

Nous avons pris l'example 5.3.2, pour mettre une erreur dans le modèle afin d'illustrer la notion du conterexemple, cet exemple montre que même si les fonctions de confidentialité fonctionne bien, il peut y avoir une dépendance entre les réponses randomisées qui provoquent une violation de la confidentialité.

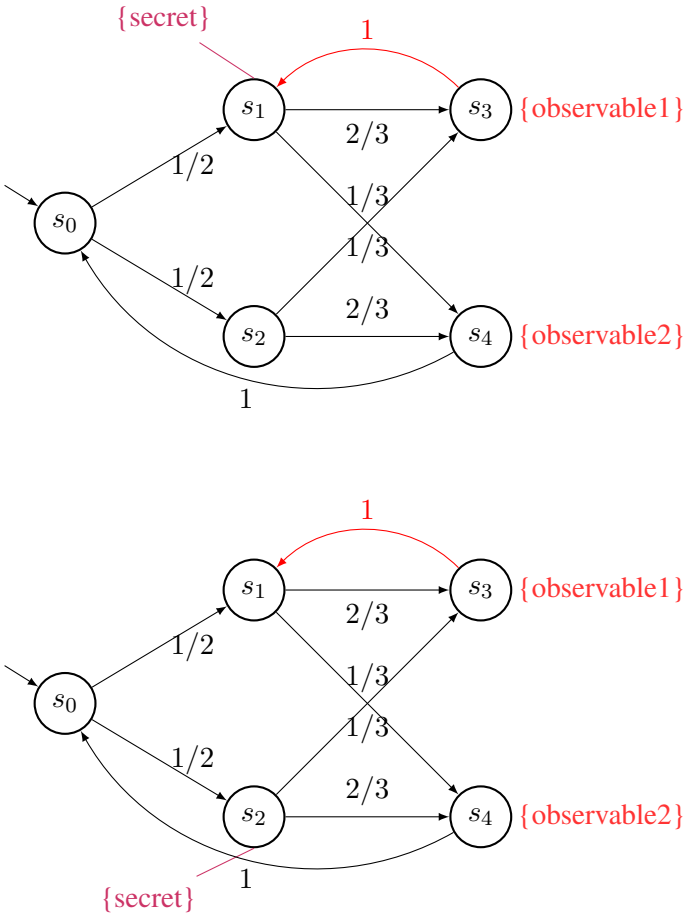


Figure 5.6 Un trou de sécurité dans un Modèle DP.

Nous avons calculé les probabilité et les valeurs de privacité et nous avons obtenu ces résultats :
$Prob^{\mathcal{D}_1}(\neg secret \bigcup observable2) = \mathsf{x} = [1/3, 0, 2/3, 0, 1]$
$Prob^{\mathcal{D}_2}(\neg secret \bigcup observable2) = \mathsf{y} = [1/2, 1, 0, 1, 1]$

$\mathcal{E}(s, \psi) = [\varepsilon_1, inf, inf, \varepsilon_1, \varepsilon_0]$, alors $\exists\, \mathsf{x}_i \in Prob^{\mathcal{D}_1}, \exists \mathsf{y}_i \in Prob^{\mathcal{D}_2}\ \ |log(\mathsf{x}_i) - log(\mathsf{y}_i)| > \varepsilon_1$

Le modèle n'est pas différ1entiellement privé parce que la propriété de la confidentialité est violée, $\psi = (\neg secret \bigcup observable2)$ indiquant que l'exécution atteint un état satisfaisant dans $\mathcal{D}_1$ (c.-à-d, atteint s_4 à partir s_3) avec une probabilité égale à 0 et la probabilité dans état s_3 est 1 dans $\mathcal{D}_2$.

Tableau 5.5 Table des comparaison

Chemins de lifting			
Rang	Chemin	Prob	Valeurs de privacité
0	$s_0 s_2 s_4$	1/3	ε_1
1	$s_0 s_2 s_4$	1/3	ε_1
2	$s_0 s_2 s_4$	1/3	ε_1
3	$s_0 s_2 s_4$	1/3	ε_1
4	$s_0 s_2 s_4$	1/3	ε_1
k	$s_0 s_2 s_4$	1/3	ε_1

(a) Chemins de lifting pour $\mathcal{D}_1$

Chemins de lifting			
Rang	Chemin	Prob	Valeurs de privacité
0	$s_0 s_1 s_4$	1/6	ε_1
1	$s_0 (s_1 s_3 s_1)^1 s_4$	0.277777	ε_1, ε_2
2	$s_0 (s_1 s_3 s_1)^2 s_4$	0.351851	ε_1, ε_2, ε_3
3	$s_0 (s_1 s_3 s_1)^3 s_4$	0.401234	ε_1, ε_2, ε_3
4	...	...	...
k	$s_0 (s_1 s_3 s_1)^k s_4$	0.499999	ε_1, ε_2, ε_3

(b) Chemins de lifting pour $\mathcal{D}_2$

Ce tableau répertorie les chemins finis de lifting classés en fonction de leur probabilité et leur valeur de confidentialité. Notez qu'il n'y a qu'un seul chemin fini dans la partie gauche du tableau que prend la branche au-dessous dans le système $\mathcal{D}_1$, alors que la partie droite de tableaux prennent des chemins ont des probabilités moins élevées et plusieurs valeurs de confidentialité, ce qui illustre le problème de l'équivalence entre les chemins de lifting. Pour ajuster le modèle de sorte qu'il satisfasse la propriété (correction de bogue), il ne suffit pas de modifier la transition du système, mais trouver la valeur maximale de ε_i. De plus, il y a des chemins finis fournissent des informations de diagnostic similaires : ils ne font que des boucles supplémentaires dans s_1 et s_3. En outre, les probabilités de chaque de ces chemins finis sont approximativement indistinguables, ce qui empêche de savoir si un chemin particulier est important.

Enfin, le contre-exemple pour la propriété de l'observable ψ consiste en un nombre infini de chemins finis (à savoir tous les chemins de $\mathcal{D}_1$ et $\mathcal{D}_2$). Pour résoudre ces problèmes, nous partitionnons un contre-exemple représentatif en ensembles de chemins finis qui suivent un modèle indistinguable. Pour garantir que les chemins de lifting fournissent des informations de diagnostic utiles, nous souhaitons que l'ensemble des chemins de lifting formant un contre-exemple réponde à plusieurs propriétés : deux couples de deux chemins différents doivent fournir des informations de diagnostic différentes, les chemins de $lifting$ doivent fournir des informations de diagnostic similaires, ce qui aura pour conséquence une masse de probabilité élevée et le nombre des contreexemples représentatifs doit être limité.

5.5.2 La confidentialité diffrentielle dirigé par les contre-exemples :

L'algorithme prend deux modèles, qui sont des description en DTMCs, $\mathcal{D}_1$ et $\mathcal{D}_2$, la propriété d'un mauvais observable, et le paramètre de privacité ε.

D'abord, nous utilisons l'un des technique de probabilistic modèle checking bourné (SBMC) pour trouver toutes les chemins où la méthode $preprocess(\mathcal{D}_1, \psi)$ renvoie tous les chemins qui satisfaisant ψ dans $\mathcal{D}_1$, puis nous calculons les chemins de lifting dans le produit cartésien de deux ensembles des chemins. Le procédure $EqualsP(\varrho_1, \varrho_2)$ compare deux chemins de lifting et détermine si elles sont équivalentes en ε. Enfin, on cherche à découvert d'une paire de chemins qu'impliquera une violation de la confidentialité différentielle.

Algorithm 3: Counterexample-Driven for Differential Privacy

Input: DTMCs $\mathcal{D}_1, \mathcal{D}_2$, property ψ and ε desired privacy

Output: $\mathcal{C}$ as a counterexample.

```
/* Find paths by SBMC                                              */
```

$R \leftarrow \emptyset$

$FPaths_1 \leftarrow preprocess(\mathcal{D}_1, \psi)$

$FPaths_2 \leftarrow preprocess(\mathcal{D}_2, \psi)$

foreach $\rho_1, \rho_2 \in FPath_1 \times FPath_2$ **do**

  $\varrho_1 \leftarrow liftingPath(\rho_1, \varepsilon)$

  $\varrho_2 \leftarrow liftingPath(\rho_2, \varepsilon)$

  if $EqualsP(\varrho_1, \varrho_2)$ **then**

    $R \leftarrow R \cup (\varrho_1, \varrho_2)$

```
/* Find of pair of path will imply a violation to differential privacy
   */
```

$\mathcal{C} \leftarrow \emptyset$

foreach $(\varrho_1, \varrho_2) \in R$ **do**

  $\mathsf{x}_1 \leftarrow computeProb(\mathcal{D}_1, \varrho_1)$

  $\mathsf{x}_2 \leftarrow computeProb(\mathcal{D}_2, \varrho_2)$

  if $(\mathsf{x}_i, \mathsf{x}_i) \notin \mathcal{R}_1^{\varepsilon}$ **then**

    $\mathcal{C} \leftarrow \mathcal{C} \cup (\varrho_1, \varrho_2, \varepsilon)$

return $\mathcal{C}$

5.6 Conclusion

Dans ce chapitre, nous avons présenté une nouvelle approche automatisée de la vérification quantitative des systèmes différentiellement confidentiels. Nous étudions le problème de construction des preuves de couplages entre les distributions de sortie de deux modèles probabilistes qui sont considérés par deux programmes probabilistes que sont exécutés au niveau deux entrées adjacentes.

Pour rendre compte formellement de cette technique de preuve, nous avons montré que les preuves formalisant des couplages qui sont des relations de lifting approximatives établissent diverses propriétés probabilistes, y compris l'équivalence de distribution par un facteur de confidentialité.

Nous avons d'abord présenté une méthode de la relaxation de la relation de lifting que modélise la propagation des valeurs privées d'arrondis à travers un ensemble de contraintes qui sont résolues par la programmation linéaire.

Nous avons développé un algorithme symbolique pour calculer les canaux de la confidentialité différentielle par les programmes booléens. L'avantage de notre approche est que ces algorithmes peuvent être intégrés à tout modèle de vérification basé sur MTBDD.

Nous avons aussi présenté une nouvelle technique pour représenter et calculer des contre-exemples pour des systèmes différentiellement confidentiels déterministes et probabilistes. Cette technique est basée sur la vérification des modèles bornée (en anglais *bounded probabilistic model-checking*) et sur le principe de déroulement.

CHAPITRE 6 CONCLUSION

Ce projet de recherche visait à développer un cadre probabiliste pour l'analyse formelle automatique des systèmes différentiellement confidentiels.

Dans le chapitre 4, nous avons fourni un algorithme de vérification de la confidentialité différentielle pour les systèmes concurrents à l'aide du vérificateur de modèles probabiliste PRISM. Comme nous l'avons montré dans l'étude de cas, ce résultat permet d'estimer la probabilité qu'un programme satisfasse la confidentialité différentielle. Notre approche à la confidentialité utilise des listes des propriétés en PCTL* qui spécifient toutes les traces de chaque secret des états initiaux modélisant des entrées adjacentes jusqu'à l'état observable. Autrement dit, pour toute trace qui commencent par l' état $\bar{s_i}$, il existe une autre trace que commence par une état $\bar{s_j}$ équivalent à $\bar{s_i}$, c'est-à-dire équivalent à chacune des traces de s_j et dont les probabilités dsont indistinguables à ε près. Nous avons construit automatiquement une matrice de canal pour chaque paire d'états secrets initiaux $\bar{s_i}$ et $\bar{s_i'}$ que sont adjacents et pour chaque observable t_j à l'aide PRISM.
Dans le chapitre 5, nous avons proposé un algorithme basé sur une technique de couplages plus riche permettant une vérification entièrement automatisées de la ε-confidentialité différentielle pour des modèles probabiliste, incluant les DTMCs et MDPs introduits au chapitre 4. En résumé, nous avons codé des preuves de couplage approximatives valides avec une relation de *lifting* sous forme de systèmes d'équation linéaire que nous avons résolu à l'aide de techniques automatisées basées sur le calcul d'un opérateur de point fixe. La relaxation des relations de *lifting* simplifie nos preuves en permettant au coût de la confidentialité d'être randomisé pendant l'évolution du système vis à vis à chaque adversaire, en utilisant un calcul itératif. À partir des états initiaux, nour construisons d'abord ces relations puis nous sélectionnons un couplage approximatif pour chaque paire d'états correspondants de deux DTMCs de manière à ce que le coût total du paramètre de confidentialité soit égal à ε. Ce coût est induit par une matrice des valeurs de confidentialité, et les sorties sont un ensemble d'états cibles sur les deux exécutions DTMCs égales sous le couplage approximatif. On peut établir séparément un couplage approximatif pour chaque d'observable o_i dans une exécution, en s'assurant que si la première sortie est égale à une valeur de confidentialité ε_i, la deuxième sortie le sera également.

Nous avons aussi proposé une méthode symbolique de vérification des modèle du confidentialité basée sur les MTBDD(s), cette méthode calcule la matrice de canal d'un modèle probabiliste pour vérifier l'indistinguabilité de deux MTBDDs selon des valeurs de confidentialité aussi réprésentées par des MTBDDs.

Enfin, dans le cas où les modèles vérifiés ne satisfont pas la conficentialité différentielle, nous générons un contre-exemple. Plus précisément, nous avons proposé une méthode basée sur l'analyse de déroulement et la vérification de modèles probabilistes bornée pour générer des contre-exemples invalidant la confidentialité différentielle.

6.1 Perspectives

Le travail effectué dans cet projet a soulevé de nombreuses questions de recherche intéressantes. Dans cette section, nous en discutons quelques unes.

6.1.1 Vérification formelle de la confidentialité au niveau de l'utilisateur :

Nous pensons utiliser des automates temporisés probabilistes (en anglais *Probabilistic timed automata (PTAs)*) pour modéliser et analyser les frameworks de renforcement de la confidentialité des systèmes temps-réel.

La confidentialité différentielle peut être intégrée dans de tels frameworks. Ces frameworks fournissent une approche holistique de protection de la vie privée, qui met en œuvre des politiques de confidentialité définies par l'utilisateur, dans laquelle la politique de confidentialité attachée à l'enregistrement d'un utilisateur peut également indiquer un délai d'expiration après lequel l'enregistrement est supprimé dela base de donnés. Étant donné que chaque utilisateur génère de nouveaux enregistrements à un débit constant et que le temps d'expiration est essentiel pour garantir qu'un nombre limité d'enregistrements provenant du même utilisateur résident simultanément dans la base de données, il est ainsi possible d'interpréter les implications des paramètres de confidentialité en termes de la sensibilité de l'utilisateur, par exemple, un nouvel enregistrement est généré toutes les 10 minutes et supprimé après 10 minutes, un temps moyen de participation dans la base de données par heure ou par jour peut divulguer des informations, cela signifie que l'attaquant aurait besoin d'un certain temps pour surveiller la base de données avant de pouvoir déduire qu'un utilisateur était déja présent dans la base de données. Il serait intéressant de construire des *PTAs* pour la *DP* et d'étudier comment utiliser des techniques de vérification de modèles probabilistes pour analyser le temps d'expiration d'une enregistrement d'un utilisateur.

Une telle méthode serait applicable a tous les systèmes mobiles que utilisent le paradigme de détection et la collecte de données (en anglais *Mobile Crowdsensing Systems*).

6.1.2 Analyse formelle des mécanismes universellement optimaux :

Les mécanismes universellement optimaux sont des mécanismes qui garantissent une utilité optimale à une grande classe d'applications, tout en préservant la confidentialité différentielle.

Dans le future, nous montrerons comment utiliser la notion des sous-groupes du groupe de symétrie dans une chaine de Markov réversible et paramétrique (en anglais *Parametric reversible Markov chain*), pour modéliser et analyser des notions de métriques et pour la vérification de l'existence d'un mécanismes universellement optimal pour une ou plusieurs requête(s) de comptage. À titre d'exemples, nous montrerons que nous pouvons naturellement exprimer et construire des mécanismes universellement optimaux par le mixage des chaînes de Markov paramétrées et les graphes symétriques avec des poids sur les arcs pour spécifier les contraintes des confidentialité.

Formellement, soit un graphe connecté $G = (S, E)$ avec un ensemble de sommets S et un ensemble d'arête E. Nous autorisons des boucles. Soit $w(e)$ des poids positifs sur les arêtes qui réprésentent des contraintes de voisinages d'un graphe.

$$\mathbf{P}[s, s']=\frac{w(s, s')}{W(s)} \quad W(s) = \textstyle\sum_{s''} w(s, s'')$$

$$\mathbf{P} : S \times S \mapsto [0, 1] \cap \mathbb{Q}$$

où $\mathbf{P}$ est une matrice de probabilités avec des valeurs rationnelles telles que pour tout $s \in S$ $\sum_{s' \in S} = \mathbf{P}[s, s'] = 1$.

Les probabilités sont représentées par des expressions régulières évaluées récursivement pour donner finalement une valeur rationnelle exacte à chaque expression. Lorsque le calcul des probabilités de transition ou bien de la fonction probabiliste d'un mécanisme n'a pas encore convergé, les valeurs intermédiaires de l'expression régulièrequi peut être utilisée pour la vérification d'existence d'un mécanismes universellement optimal.

6.1.3 La confidentialité diffrentielle : temps, espace et l'aléa

La confidentialité différentielle est définie sur un canal bruité c'est-à-dire, d'un mapping des entrées sur les sorties randomisées. Cette façon de définir cause problème. Un adversaire peut divulger les informations confidentielles par l'utilisation de toutes les informations disponibles et ne se limitera pas à l'analyse des réponses des mécanismes et et de leur paramètres de confidentialité. Les attaques dites par les canaux secondaires démontrent que des caractéristiques telles que l'attaque temporelle et l'attaque du budget de la vie privée peuvent être exploitées efficacement par un adversaire, en voici la description.

Le système donne à l'adversaire trois informations [43] :

1. La réponse actuelle à leur requête (un nombre, un histogramme, etc.). i.e *observable* **o** ;
2. Le temps **t** auquel la réponse arrive à l'adversaire ou bien à la fin de la connexion réseau ;
3. La décision **d** du système, soit d'exécuter leur requête ou de refuser, car cela dépasserait le budget de protection de la vie privée disponible.

Voici une démonstration que le temps d'exécution mesurable viole la confidentialité :

$$\frac{1}{exp^{(\varepsilon)}} \leq \frac{Pr(o \ \wedge \ d = 1 \ \wedge \ t \mid s)}{Pr(o \ \wedge \ d = 1 \ \wedge \ t \mid s')} \leq exp^{(\varepsilon)}$$

$$\frac{1}{exp^{(\varepsilon)}} \leq \frac{Pr(o \ \wedge \ t \mid s)}{Pr(o \ \wedge \ t \mid s')} \leq exp^{(\varepsilon)}$$

$$\frac{Pr(o \ \wedge \ t \mid s) \geq 0}{Pr(o \ \wedge \ t \mid s') = 0} = \infty$$

La confidentialité différentielle est donc vulnérable.

Une hypothèse fondamentale qui sous-tend les techniques de la confidentialité différentielle est que les programme exécuteront fidèlement le fonction du désinfection (somme ou comptage). Une piste intéressante de recherche consisterait à chercher comment représenter les attaques qui exploitent le fait que l'exécution de la fonction de désinfection et la nature des structures de données qui nous affranchirait de cette hypothèse. La manipulation infidèle des structures de données, peut permettre d'accéder aux enregistrements en effectuant des opérations de parcours avec effets secondaires mesurables par l'adversaire.

RÉFÉRENCES

[1] C. Dwork, A. Roth *et al.*, "The algorithmic foundations of differential privacy," *Foundations and Trends® in Theoretical Computer Science*, vol. 9, n°. 3–4, p. 211–407, 2014.

[2] A. Narayanan et V. Shmatikov, "How to break anonymity of the netflix prize dataset," 2006.

[3] M. C. Tschantz, D. Kaynar et A. Datta, "Formal verification of differential privacy for interactive systems," *Electronic Notes in Theoretical Computer Science*, vol. 276, p. 61–79, 2011.

[4] M. S. Alvim, M. E. Andrés, K. Chatzikokolakis et C. Palamidessi, "On the relation between differential privacy and quantitative information flow," dans *International Colloquium on Automata, Languages, and Programming*. Springer, 2011, p. 60–76.

[5] G. Barthe, M. Gaboardi, J. Hsu et B. Pierce, "Programming language techniques for differential privacy," *ACM SIGLOG News*, vol. 3, n°. 1, p. 34–53, 2016.

[6] L. Xu, "Formal verification of differential privacy in concurrent systems," Thèse de doctorat, Ecole Polytechnique (Palaiseau, France), 2015.

[7] C. Dwork, K. Kenthapadi, F. McSherry, I. Mironov et M. Naor, "Our data, ourselves : Privacy via distributed noise generation," dans *Annual International Conference on the Theory and Applications of Cryptographic Techniques*. Springer, 2006, p. 486–503.

[8] F. D. McSherry, "Privacy integrated queries : an extensible platform for privacy-preserving data analysis," dans *Proceedings of the 2009 ACM SIGMOD International Conference on Management of data*. ACM, 2009, p. 19–30.

[9] I. Roy, S. T. Setty, A. Kilzer, V. Shmatikov et E. Witchel, "Airavat : Security and privacy for mapreduce." dans *NSDI*, vol. 10, 2010, p. 297–312.

[10] A. Hartmanns, M. Klauck, D. Parker, T. Quatmann et E. Ruijters, "The quantitative verification benchmark set," dans *Proc. 25th International Conference on Tools and Algorithms for the Construction and Analysis of Systems (TACAS'19)*, ser. LNCS, vol. 11427. Springer, 2019, p. 344–350.

[11] M. Kwiatkowska, G. Norman et D. Parker, "Stochastic model checking," dans *Formal Methods for the Design of Computer, Communication and Software Systems : Performance Evaluation (SFM'07)*, ser. LNCS (Tutorial Volume), M. Bernardo et J. Hillston, édit., vol. 4486. Springer, 2007, p. 220–270.

[12] J. Rutten, M. Kwiatkowska, G. Norman et D. Parker, *Mathematical Techniques for Analyzing Concurrent and Probabilistic Systems,* P. Panangaden and F. van Breugel (eds.), ser. CRM Monograph Series. American Mathematical Society, 2004, vol. 23.

[13] H. Hansson et B. Jonsson, “A logic for reasoning about time and reliability,” *Formal aspects of computing*, vol. 6, n°. 5, p. 512–535, 1994.

[14] E. M. Clarke et E. A. Emerson, “Design and synthesis of synchronization skeletons using branching time temporal logic,” dans *Workshop on Logic of Programs*. Springer, 1981, p. 52–71.

[15] M. Kwiatkowska, G. Norman et D. Parker, “PRISM 4.0 : Verification of probabilistic real-time systems,” dans *Proc. 23rd International Conference on Computer Aided Verification (CAV’11)*, ser. LNCS, G. Gopalakrishnan et S. Qadeer, édit., vol. 6806. Springer, 2011, p. 585–591.

[16] I. Dinur et K. Nissim, “Revealing information while preserving privacy,” dans *Proceedings of the twenty-second ACM SIGMOD-SIGACT-SIGART symposium on Principles of database systems*. ACM, 2003, p. 202–210.

[17] C. Dwork et K. Nissim, “Privacy-preserving datamining on vertically partitioned databases,” dans *Annual International Cryptology Conference*. Springer, 2004, p. 528–544.

[18] A. Blum, C. Dwork, F. McSherry et K. Nissim, “Practical privacy : the sulq framework,” dans *Proceedings of the twenty-fourth ACM SIGMOD-SIGACT-SIGART symposium on Principles of database systems*. ACM, 2005, p. 128–138.

[19] C. Dwork, F. McSherry, K. Nissim et A. Smith, “Calibrating noise to sensitivity in private data analysis,” dans *Theory of cryptography conference*. Springer, 2006, p. 265–284.

[20] J. Reed et B. C. Pierce, “Distance makes the types grow stronger : a calculus for differential privacy,” dans *ACM Sigplan Notices*, vol. 45, n°. 9. ACM, 2010, p. 157–168.

[21] S. B. Akers, “Binary decision diagrams,” *IEEE Trans. Comput.*, vol. 27, n°. 6, p. 509–516, juin 1978. [En ligne]. Disponible : https://doi.org/10.1109/TC.1978.1675141

[22] R. I. B. E. A. Frohm et C. M. G. G. D. Hachtel, “Algebraic decision diagrams and their applications,” dans *International Conference on Computer Aided Design*. Citeseer, 1993.

[23] M. Fujita, P. C. McGeer et J.-Y. Yang, “Multi-terminal binary decision diagrams : An efficient data structure for matrix representation,” *Formal methods in system design*, vol. 10, n°. 2-3, p. 149–169, 1997.

[24] Y. R. Chao, “A note on “continuous mathematical induction.”,” *Bull. Amer. Math. Soc.*, vol. 26, n°. 1, p. 17–18, 10 1919. [En ligne]. Disponible : https://projecteuclid.org:443/euclid.bams/1183425067

[25] P. L. Clark, “The instructor’s guide to real induction,” *Mathematics Magazine*, vol. 92, n°. 2, p. 136–150, 2019. [En ligne]. Disponible : https://doi.org/10.1080/0025570X.2019.1549902

[26] J. Hsu, “Probabilistic couplings for probabilistic reasoning,” *CoRR*, vol. abs/1710.09951, 2017. [En ligne]. Disponible : http://arxiv.org/abs/1710.09951

[27] J. A. Goguen et J. Meseguer, “Unwinding and inference control,” dans *Proceedings of the 1984 IEEE Symposium on Security and Privacy, Oakland, California, USA, April 29 - May 2, 1984*. IEEE Computer Society, 1984, p. 75–87. [En ligne]. Disponible : https://doi.org/10.1109/SP.1984.10019

[28] E. M. Clarke, “The birth of model checking,” dans *25 Years of Model Checking*. Springer, 2008, p. 1–26.

[29] V. Shmatikov, “Probabilistic model checking of an anonymity system,” *Journal of Computer Security*, vol. 12, n°. 3/4, p. 355–377, 2004.

[30] G. Norman et V. Shmatikov, “Analysis of probabilistic contract signing,” *Journal of Computer Security*, vol. 14, n°. 6, p. 561–589, 2006.

[31] G. Norman, D. Parker et J. Sproston, “Model checking for probabilistic timed automata,” *Formal Methods in System Design*, vol. 43, n°. 2, p. 164–190, 2013.

[32] A. Américo, A. Vaz, M. S. Alvim, S. V. Campos et A. McIver, “Formal analysis of the information leakage of the dc-nets and crowds anonymity protocols,” dans *Brazilian Symposium on Formal Methods*. Springer, 2017, p. 142–158.

[33] G. Barthe, B. Köpf, F. Olmedo et S. Zanella-Béguelin, “Probabilistic relational reasoning for differential privacy,” *ACM Transactions on Programming Languages and Systems (TOPLAS)*, vol. 35, n°. 3, p. 9, 2013.

[34] J. Yang, Y. Cao et H. Wang, “Differential privacy in probabilistic systems,” *Information and Computation*, vol. 254, p. 84–104, 2017.

[35] V. Castiglioni, K. Chatzikokolakis et C. Palamidessi, “A logical characterization of differential privacy via behavioral metrics,” dans *Formal Aspects of Component Software - 15th International Conference, FACS 2018, Pohang, South Korea, October 10-12, 2018, Proceedings*, 2018, p. 75–96. [En ligne]. Disponible : https://doi.org/10.1007/978-3-030-02146-7_4

[36] E. Ábrahám et B. Bonakdarpour, “Hyperpctl : A temporal logic for probabilistic hyperproperties,” dans *International Conference on Quantitative Evaluation of Systems*. Springer, 2018, p. 20–35.

[37] D. Liu, B.-Y. Wang et L. Zhang, “Model checking differentially private properties,” dans *Asian Symposium on Programming Languages and Systems*. Springer, 2018, p. 394–414.

[38] A. Ghosh, T. Roughgarden et M. Sundararajan, “Universally utility-maximizing privacy mechanisms,” *SIAM Journal on Computing*, vol. 41, n°. 6, p. 1673–1693, 2012.

[39] D. A. Parker, “Implementation of symbolic model checking for probabilistic systems,” Thèse de doctorat, University of Birmingham, 2003.

[40] H. Hermanns, M. Kwiatkowska, G. Norman, D. a. Parker et M. Siegle, “On the use of mtbdds for performability analysis and verification of stochastic systems,” *The Journal of Logic and Algebraic Programming*, vol. 56, n°. 1-2, p. 23–67, 2003.

[41] Z. Ding, Y. Wang, G. Wang, D. Zhang et D. Kifer, “Detecting violations of differential privacy,” dans *Proceedings of the 2018 ACM SIGSAC Conference on Computer and Communications Security*. ACM, 2018, p. 475–489.

[42] M. Ochoa, J. Cuéllar, A. Pretschner et P. Hallgren, “Idea : Unwinding based model-checking and testing for non-interference on efsms,” dans *International Symposium on Engineering Secure Software and Systems*. Springer, 2015, p. 34–42.

[43] A. Haeberlen, B. C. Pierce et A. Narayan, “Differential privacy under fire,” dans *Proceedings of the 20th USENIX Conference on Security*, ser. SEC’11. USA : USENIX Association, 2011, p. 33.

[44] P. L. Clark, “The instructor’s guide to real induction,” *arXiv preprint arXiv :1208.0973*, 2012.

[45] G. Malecha, D. Ricketts, M. M. Alvarez et S. Lerner, “Towards foundational verification of cyber-physical systems,” dans *2016 Science of Security for Cyber-Physical Systems Workshop (SOSCYPS)*. IEEE, 2016, p. 1–5.

[46] L. Lamport, “How to write a 21 st century proof,” *Journal of fixed point theory and applications*, vol. 11, n°. 1, p. 43–63, 2012.

ANNEXE A PREUVES

Preuve du lemme 1.

Étant donné deux DTMCs $(\mathcal{D}_1, \mathcal{D}_2)$, une propriété φ et un paramètre privé ε
si $\dfrac{\Pr^{\mathcal{D}_1}(\varphi)}{\Pr^{\mathcal{D}_2}(\varphi)} \leqslant exp(\varepsilon)$ *alors $\mathcal{D}_1$ et $\mathcal{D}_2$ sont ε-Confidentialité Différentielle.*

Preuve :

Soient $\varphi = [\Diamond Obs]$ et l'ensemble des chemins contribuant à la probabilité d'atteindre Obs de s est donné par $\Diamond Obs(s) = \{\pi \in Path^{\mathcal{D}}_{fin}(s) \mid \exists i \;\; observable \in L(\pi(i)\}$
Lors du calcul de la masse de probabilité de $\Diamond Obs(s)$, en prise en compte les chemins finissent à visiter Obs uniquement dans leur dernier état. La probabilité de cet ensemble peut être calculée par la somme des probabilités de ses éléments : $Pr(\Diamond Obs(s))$*, telle que $\Diamond Obs(s)$ est égal à l'union des ensembles de cylindres de tous les chemins de $Path^{\mathcal{D}}_{fin}(s, Obs)$.*

$$\frac{\Pr^{\mathcal{D}_1}_{s'_1}(\Diamond Obs(s'_1))}{\Pr^{\mathcal{D}_2}_{s'_2}(\Diamond Obs(s'_2))} = \frac{\Pr^{\mathcal{D}_1}_{s'_1}(\bigcup_{\pi_1 \in \Diamond Obs_{fin}(s'_1)} Cyl(\pi_1))}{\Pr^{\mathcal{D}_2}_{s'_2}(\bigcup_{\pi_2 \in \Diamond Obs_{fin}(s'_2)} Cyl(\pi_2))}$$

$$= \frac{\sum_{\pi_1 \in \Diamond Obs_{fin}(s'_1)} \Pr^{\mathcal{D}_1}_{s'_1} Cyl(\pi_1)}{\sum_{\pi_2 \in \Diamond Obs_{fin}(s'_2)} \Pr^{\mathcal{D}_2}_{s'_2} Cyl(\pi_2)}$$

$$= \frac{\sum_{s'_1 \in S \setminus Obs} \boldsymbol{P}(s_1, s'_1).\boldsymbol{Pr}^{\mathcal{D}_1}_{s'_1}(\Diamond Obs(s'_1)) + \sum_{s'_1 \in Obs} \boldsymbol{P}(s_1, s'_1)}{\sum_{s'_2 \in S \setminus Obs} \boldsymbol{P}(s_2, s'_2).\boldsymbol{Pr}^{\mathcal{D}_2}_{s'_2}(\Diamond Obs(s'_2)) + \sum_{s'_2 \in Obs} \boldsymbol{P}(s_2, s'_2)}$$

On montre l'assertion de la confidentialité diffrentielle dans différents parties probabilistes.

Nous distinguons trois cas dans lesquelles les $(\mathcal{D}_1, \mathcal{D}_2)$ sont indistingables :

1. ***Premier cas :*** $\boldsymbol{Pr}^{\mathcal{D}_1}_{s'_1}(\Diamond Obs(s'_1)) = 0 \;\; \wedge \;\; \boldsymbol{Pr}^{\mathcal{D}_2}_{s'_2}(\Diamond Obs(s'_2)) = 0$ *Alors*
   l'observable n'est pas accessible : $\dfrac{\Pr^{\mathcal{D}_1}_{s'_1}(\Diamond Obs(s'_1))}{\Pr^{\mathcal{D}_2}_{s'_2}(\Diamond Obs(s'_2))} = 0 \leqslant exp(\varepsilon)$

2. ***Deuxième cas :*** $\boldsymbol{Pr}^{\mathcal{D}_1}_{s'_1}(\Diamond Obs(s'_1)) = 1 \;\; \wedge \;\; \boldsymbol{Pr}^{\mathcal{D}_2}_{s'_2}(\Diamond Obs(s'_2)) = 1$ *Alors*
   $$\frac{\Pr^{\mathcal{D}_1}_{s'_1}(\Diamond Obs(s'_1))}{\Pr^{\mathcal{D}_2}_{s'_2}(\Diamond Obs(s'_2))} = \frac{\sum_{s'_1 \in S \setminus Obs} \boldsymbol{P}(s_1, s'_1) + \sum_{s'_1 \in Obs} \boldsymbol{P}(s_1, s'_1)}{\sum_{s'_2 \in S \setminus Obs} \boldsymbol{P}(s_2, s'_2) + \sum_{s'_2 \in Obs} \boldsymbol{P}(s_2, s'_2)}$$
   $$= \frac{1}{1} \leqslant exp(\varepsilon)$$

3. ***Troisième cas :*** $\boldsymbol{Pr}_{s'_1}^{\mathcal{D}_1}(\Diamond Obs(s'_1)) \in]0,1[\ \wedge \ \boldsymbol{Pr}_{s'_2}^{\mathcal{D}_2}(\Diamond Obs(s'_2)) \in]0,1[$ *Alors*

$$\frac{\Pr_{s'_1}^{\mathcal{D}_1}(\Diamond Obs(s'_1))}{\Pr_{s'_2}^{\mathcal{D}_2}(\Diamond Obs(s'_2))} = \frac{\sum_{s'_1 \in S\setminus Obs} \boldsymbol{P}(s_1, s'_1).\boldsymbol{Pr}_{s'_1}^{\mathcal{D}_1}(\Diamond Obs(s'_1)) + \sum_{s'_1 \in Obs} \boldsymbol{P}(s_1, s'_1)}{\sum_{s'_2 \in S\setminus Obs} \boldsymbol{P}(s_2, s'_2).\boldsymbol{Pr}_{s'_2}^{\mathcal{D}_2}(\Diamond Obs(s'_2)) + \sum_{s'_2 \in Obs} \boldsymbol{P}(s_2, s'_2)}$$

Comme $\sum_{s'_1 \in S\setminus Obs} \boldsymbol{P}(s_1, s'_1) \ \wedge \sum_{s'_2 \in S\setminus Obs} \boldsymbol{P}(s_2, s'_2)$ *sont des connaissances de l'attaquant :*
$\sum_{s'_1 \in S\setminus Obs} \boldsymbol{P}(s_1, s'_1) = 1 - \sum_{s'_2 \in S\setminus Obs} \boldsymbol{P}(s_2, s'_2)$.

Donc si l'on souhaite montrer l'assertion dans ce cas, on montre en fait que quelles que soient les relations d'ordre total dans le corps $([0,1], +, \times, \leq)$ *est vraie alors l'assertion de la confidentialité diffrentielle est vraie.*
Nous avons des contraintes sur les probabilités du mécanisme privé :

$$\frac{\sum_{s'_1 \in Obs} \boldsymbol{P}(s_1, s'_1)}{\sum_{s'_2 \in Obs} \boldsymbol{P}(s_2, s'_2)} \leqslant exp(\varepsilon)$$

Et nous avons un rapport de deux expressions récursives pour tout état $s'_i \in S\setminus Obs$ *:*

$$\frac{\sum_{s'_1 \in S\setminus Obs} \boldsymbol{P}(s_1, s'_1).\boldsymbol{Pr}_{s'_1}^{\mathcal{D}_1}(\Diamond Obs(s'_1))}{\sum_{s'_2 \in S\setminus Obs} \boldsymbol{P}(s_2, s'_2).\boldsymbol{Pr}_{s'_2}^{\mathcal{D}_2}(\Diamond Obs(s'_2))}$$

telle que : $\dfrac{\sum_{s'_1 \in S\setminus Obs} \boldsymbol{P}(s_1, s'_1)}{\sum_{s'_2 \in S\setminus Obs} \boldsymbol{P}(s_2, s'_2)} \leqslant exp(\varepsilon)$

Ainsi selon l'associativité et la commutativité de l'addition d'une somme de plusieurs termes dans n'importe quel ordre .

Nous obtenons : $\dfrac{\Pr_{s'_1}^{\mathcal{D}_1}(\Diamond Obs(s'_1))}{\Pr_{s'_2}^{\mathcal{D}_2}(\Diamond Obs(s'_2))} \leqslant exp(\varepsilon)$

Conclusion *Dans tous les cas :* $\dfrac{\Pr^{\mathcal{D}_1}(\varphi)}{\Pr^{\mathcal{D}_2}(\varphi)} \leqslant exp(\varepsilon)$

Preuve du lemme 2.

Étant donné un MDP $\mathcal{M}$, une propriété φ, un paramètre privé ε et pour tout adversaire σ les DTMCs induit $\mathcal{D}_1^\sigma$ et $\mathcal{D}_2^\sigma$ telle que $Pr^{\mathcal{D}_1,\sigma}(\varphi) \leqslant exp(\varepsilon) \times Pr^{\mathcal{D}_2,\sigma}(\varphi)$ alors $\mathcal{M}$ est ε-Confidentialité Différentielle.

Preuve :

Nous prouvons par induction sur la longueur de chemin ω, la longueur de chemin $|\omega|$ (nombre de transitions)

1. $|\omega_i| = 0, \quad Pr^{\mathcal{D}_1,\sigma}(\omega_1(0) \models \varphi) = Pr^{\mathcal{D}_2,\sigma}(\omega_2(0) \models \varphi) = 1$
2. *IH : Pour toute les chemins,si on a deux chemins $\sigma(\omega_1)$ et $\sigma(\omega_2)$ telle que :*
   $(last(\sigma(\omega_1)), last(\sigma(\omega_2))) \in \mathcal{R}_1^\varepsilon$ *et* $|\omega_1| = m$ *et* $|\omega_1| = n$ *implique :*

$$\frac{1}{e^{(\varepsilon)}} \leq \frac{\boldsymbol{P}(\omega_1(0), \omega_1(1))...\boldsymbol{P}(\omega_1(m-1), \omega_1(m))}{\boldsymbol{P}(\omega_2(0), \omega_2(1))...\boldsymbol{P}(\omega_2(n-1), \omega_2(n))} \leq e^{(\varepsilon)}$$

3. *Nous devons montrer pour que $|\omega_1| = m + 1$, $|\omega_1| = n + 1$ et $(last(\sigma(\omega_1)), last(\sigma(\omega_2))) \in \mathcal{R}_1^\varepsilon$ implique :*

$$\frac{1}{e^{(\varepsilon)}} \leq \frac{\boldsymbol{P}(\omega_1(0), \omega_1(1))...\boldsymbol{P}(\omega_1(m), \omega_1(m+1))}{\boldsymbol{P}(\omega_2(0), \omega_2(1))...\boldsymbol{P}(\omega_2(n), \omega_2(n+1))} \leq e^{(\varepsilon)}$$

   Nous prendre deux cas en considération :

   (a) $Sat(\neg\varphi)$: *Alors* $\sigma(\omega_1)(i) \not\models \varphi$ *et* $Pr^{\mathcal{D}_1,\sigma}(\varphi) = 0$

   (b) $Sat(\varphi)$:

$$\frac{Pr^{\mathcal{D}_1,\sigma}(s_1, \varphi)}{Pr^{\mathcal{D}_2,\sigma}(s_2, \varphi)} = \frac{\sum\limits_{s_1' \in S} \boldsymbol{P}(s_1, s_1').Pr^{\mathcal{D}_1,\sigma}(s_1', \varphi)}{\sum\limits_{s_2' \in S} \boldsymbol{P}(s_2, s_2').Pr^{\mathcal{D}_2,\sigma}(s_2', \varphi)} = \frac{\sum\limits_{s' \in S} \boldsymbol{P}(s_1, s_1').\pi_{s_1,m}^{\mathcal{D}_1,\sigma}(s_1')}{\sum\limits_{s_2' \in S} \boldsymbol{P}(s_1, s_2').\pi_{s_2,n}^{\mathcal{D}_2,\sigma}(s_2')}$$

   telle que :

$\pi_{s_1,m}^{\mathcal{D}_1,\sigma}(s_1') = Pr\{\omega_1 \in Path^{\mathcal{D}_1,\sigma}(s_1) \mid \omega_1(m) = s_1'\}$
$\pi_{s_2,n}^{\mathcal{D}_2,\sigma}(s_2') = Pr\{\omega_2 \in Path^{\mathcal{D}_2,\sigma}(s_2) \mid \omega_2(n) = s_2'\}$

Alors selon l'hypothèse d'induction et les contraintes sur les probabilités du mécanisme privé, on a :

$$\frac{Pr^{\mathcal{D}_1,\sigma}(s_1,\varphi)}{Pr^{\mathcal{D}_2,\sigma}(s_2,\varphi)} = \frac{\sum_{s_1' \in S} \boldsymbol{P}(s_1,s_1').\boldsymbol{P}(\omega_1(0),\omega_1(1))...\boldsymbol{P}(\omega_1(m-1),\omega_1(m))}{\sum_{s_2' \in S} \boldsymbol{P}(s_2,s_2').\boldsymbol{P}(\omega_2(0),\omega_2(1))...\boldsymbol{P}(\omega_2(n-1),\omega_2(n))}$$

et pour $s_1' \models \varphi$ *et* $s_2' \models \varphi$ *:*

$$\frac{Pr^{\mathcal{D}_1,\sigma}(s_1,\varphi)}{Pr^{\mathcal{D}_2,\sigma}(s_2,\varphi)} = \frac{\boldsymbol{P}(\omega_1(0),\omega_1(1))...\boldsymbol{P}(\omega_1(m-1),\omega_1(m)).\boldsymbol{P}(s_1,s_1')}{\boldsymbol{P}(\omega_2(0),\omega_2(1))...\boldsymbol{P}(\omega_2(n-1),\omega_2(n)).\boldsymbol{P}(s_2,s_2')}$$

alors

$$\frac{1}{e^{(\varepsilon)}} \leq \frac{Pr^{\mathcal{D}_1,\sigma}(s_1,\varphi_1)}{Pr^{\mathcal{D}_2,\sigma}(s_2,\varphi_2)} = \frac{\boldsymbol{P}(\omega_1(0),\omega_1(1))...\boldsymbol{P}(\omega_1(m),\omega_1(m+1))}{\boldsymbol{P}(\omega_2(0),\omega_2(1))...\boldsymbol{P}(\omega_2(n),\omega_2(n+1))} \leq e^{(\varepsilon)}$$

Q.E.D

Preuve du proposition 1.

Nous dirons qu'une chaîne de Markov $\mathcal{D}$ d'un mécanisme privé qui produit une suite des réponses randomisé à une suite de k requêtes est (v, p_1, p_2)-précis par rapport à un seuil T .

$$\bigwedge_i^k (\bigvee_j^n (R_j + v_i < T))$$

Avec :

- R_j : *Réponse réel à j-ième requête.*
- v_i : *Bruit randomisé ajouter à i-ième réponse réel possible .*
- T : *seuil*
- p_1 *: "La probabilité de rester pour toujours dans les états au-dessus du seuil. "**AboveThreshold**"*
- p_2 *: "La probabilité possible d'atteindre "**UnderThreshold**"*

Preuve :
Soient $AboveThreshold, UnderThreshold \in AP$.

Nous raisonnons par l'absurde en supposant que : $UnderThreshold \equiv \neg AboveThreshold$
Comme $\Box AboveThreshold \equiv \neg\Diamond\neg AboveThreshold$ *alors*

$$\begin{aligned} P_{\geq p_1}[\Box AboveThreshold] &\iff Pr^{\mathcal{D}}(s, \Box AboveThreshold) \geq p_1 \\ &\iff 1 - Pr^{\mathcal{D}}(s, \Diamond\neg AboveThreshold) \geq p_1 \\ &\iff Pr^{\mathcal{D}}(s, \Diamond\neg AboveThreshold) \leq 1 - p_1 \\ &\iff P_{\leq 1-p_1}[\Diamond\neg AboveThreshold] \\ &\iff P_{\leq p_2}[\Diamond UnderThreshold] \end{aligned}$$

Conclusion : si $AboveThreshold \equiv (\bigwedge_i^k (\bigvee_j^n (R_j + v_i < T)))$ *alors*
$P_{\geq p_1}[\Box AboveThreshold]$ *et* $P_{\leq p_2}[\Diamond UnderThreshold]$

Preuve du corollaire 1.

$\mathfrak{L}$ *est vraie on* ε_n *ssi* $\exists n \in \mathbb{N},\ (P^{(n)}[t_j|s_i], P^{(n)}[t_j|s_{i+1}]) \in \mathcal{R}_1^{\mathfrak{L}(\varepsilon_n)}$

(RI1) $\mathfrak{L}$ *est vraie pour* $\varepsilon_n = 0$.

(RI2) Pour toute $\varepsilon_n > 0$*, si* $\mathfrak{L}$ *est vraie pour tout* $\varepsilon_k \in [\varepsilon_n, \varepsilon_0]$ *alors* $\mathfrak{L}$ *est vraie pour tout* $\varepsilon_{n+k} \in (0, \varepsilon_n]$ *pour* $\varepsilon_n < \varepsilon_0$.

(RI3) Pour toute $\varepsilon_n > 0$*, si* $\mathfrak{L}$ *est vraie pour tout* $\varepsilon_k \in (\varepsilon_n, \varepsilon_0]$ *alors* $\mathfrak{L}$ *est vraie pour* ε_n

Pour l'utiliser l'induction mathématique, on pense en termes de prédictions, c'est-à-dire des énoncés $\mathfrak{L}(\varepsilon_i)$ *dont les valeurs de la confidentialité* ε_i *indexés par les nombres naturels* $\mathbb{N}$

Nous pourrions considérer les valeurs ε_i *comme un intervalle non vide de nombres réels.* [1] *Autrement dit, nous pourrions ne prouver que* $\mathfrak{L}(\varepsilon_i)$ *pour tous les nombres réels inférieurs à certains nombres.*

Notre façon de résoudre ce problème est similaire à d'autres travaux [44] [45]. L'une consiste à s'assurer que chaque étape est toujours au moins égale à une constante positive dont la valeur ε *est au pire de cas. Cependant, nous adopterons une approche alternative . Dans cette approche, nous veillerons à ce que, si nous prouvons* $\mathfrak{L}(i)$ *pour tous les nombres inférieurs à une limite* ε*, nous le prouvons également. Formellement, cela signifie que nous avons besoin d'une troisième étape inductive (RI3)*

Cette preuve de l'induction réelle pour la confidentialité différentielle procède par une contradiction. Nous construisons l'ensemble de tous les éléments de ε_i *ne satisfaisant pas* $\mathfrak{L}$ *et compter sur le fait que cet ensemble a une borne supérieure. Si la borne supérieure satisfait* $\mathfrak{L}$*, cela contredit la deuxième étape inductive (RI2) et, si elle ne satisfait pas* $\mathfrak{L}$*, cela contredit la troisième étape inductive (RI3)*

Nous proposons une preuve formelle de la relaxation de la relation de lifting $\mathfrak{L}$*, en suivant le style de preuve structurée de Leslie Lamport [46] : How to Write a* 21^{st} *Century Proof.*

1. Ce raisonnement de réel inductive, dont chaque étape inductive pour les nombres réels exigerait de prouver de continuer à faire des progrès positifs sans jamais couvrir tous les chiffres réels.

Soit $\mathsf{A} = \{\mathcal{P}_{i,j} \wedge \varepsilon_k \in \mathbb{R} : \varepsilon_k \geq 0 \wedge (P^{(k)}[t_j|s_i], P^{(k)}[t_j|s_{i'}]) \notin \mathcal{R}_1^{\mathfrak{L}(\varepsilon_k)}\}$
On suppose : $\exists \varepsilon_k \in \mathsf{A}$
On prouve : faux.

1. $\exists\, \varepsilon_i \in \mathbb{R} :$ *telle que* ε_i *une borne supérieure de* A
   Preuve : - La confidentialité différentielle promettre que la perte de la confidentialité est limitée à la borne supérieure par une quantité approximativement linéaire ε*.*
   - Tous les ensembles de réels des bornes supérieurs non vide ont une borne supérieure.
2. *Cas :* $\varepsilon_i \notin \mathsf{A}$
   - 2.1 $\forall\, \varepsilon_j \in [0, \varepsilon_i]\,,\, \mathfrak{L}(\varepsilon_j)$
     Preuve : Par la définition de la confidentialité différentielle et la définition de la borne supérieure
   - 2.2 $\exists\, \varepsilon_j < \varepsilon_i\,,\, \forall\, \varepsilon_j \in (0, \varepsilon_i]\,,\, \mathfrak{L}(\varepsilon_j)$
     Preuve : Par la définition de la symétrie, nous avons deux chemins probabilistes ρ_1 *et* ρ_2 *telle que :*
     $$\frac{P(\rho_1)}{P(\rho_2)} = \frac{\prod_i^j P(t,s)}{\prod_j^i P(t,s')} = \frac{P^{(i)}[t|s].P^{(j)}[t|s]}{P^{(j)}[t|s'].P^{(i)}[t|s']} = \left(\frac{P^{(i)}[t|s]}{P^{(i)}[t|s']}\right)\left(\frac{P^{(j)}[t|s]}{P^{(j)}[t|s']}\right) \leq exp(\varepsilon_i).exp(\varepsilon_j)$$
     et par 2.1 et (RI2)
   - 2.3 ε_j *est un majorant de* A
     Preuve : par 2.1 et 2.2
   - 2.4 *QED*
     Preuve : par 2.3, la définition de la confidentialité différentielle et la définition de la borne supérieure
3. *Cas :* $\varepsilon_i \in \mathsf{A}$
   - 3.1 $\forall\, 0 \leq \varepsilon_j < \varepsilon_i\,,\, \mathfrak{L}(\varepsilon_j)$
     Preuve : Parce que ε_i *est un majorant de* A
   - 3.2 $\mathfrak{L}(\varepsilon_i)$
     Preuve : (RI3)
   - 3.3 *QED*
     Preuve : par 3.2 et 3.
4. *QED*
   Preuve : par 2 et 3.

ANNEXE B MODÈLES PRISM POUR LA CONFIDENTIALITÉ DIFFÉRENTIELLE

Dans cette annexe, nous présentons le code source des modèles PRISM pour l'étude d'un système différentiellement confidentiel, qui a été présenté à la section 4.8.

```
mdp

const int N=3 ;        // size of database
const int alpha = 2;  //security parameter
const int t = 2;       // data structure length
const int k;

// Domain of a variable datapoint
const int DATA_MIN = -1;
const int DATA_MAX = 8;

// The Truncated Geometric Mechanism
//when i is the real value, report j with probability:
//p(j|i) = cj * alpha^-|i-j|  where  :
//cj=alpha/(alpha+1)   if j=0 or j=n
//cj=alpha-1/(alpha+1) if   0 < j < n
//alpha  > 1 is a security parameter (alpha= exp(epsilon))

const double P0_0_0=1 ;

const double P1_0_0=0.6666666666666666;
const double P1_1_0=0.3333333333333333;
const double P1_0_1=0.3333333333333333;
const double P1_1_1=0.6666666666666666;

const double P2_0_0=0.6666666666666666;
const double P2_1_0=0.16666666666666666;
const double P2_2_0=0.16666666666666666;
const double P2_0_1=0.3333333333333333;
const double P2_1_1=0.3333333333333333;
const double P2_2_1=0.3333333333333333;
const double P2_0_2=0.16666666666666666;
const double P2_1_2=0.16666666666666666;
const double P2_2_2=0.6666666666666666;

const double P3_0_0=0.6666666666666666;
const double P3_1_0=0.16666666666666666;
const double P3_2_0=0.08333333333333333;
const double P3_3_0=0.08333333333333333;
const double P3_0_1=0.3333333333333333;
const double P3_1_1=0.3333333333333333;
const double P3_2_1=0.16666666666666666;
const double P3_3_1=0.16666666666666666;
const double P3_0_2=0.16666666666666666;
const double P3_1_2=0.16666666666666666;
const double P3_2_2=0.3333333333333333;
const double P3_3_2=0.3333333333333333;
const double P3_0_3=0.08333333333333333;
const double P3_1_3=0.08333333333333333;
const double P3_2_3=0.16666666666666666;
const double P3_3_3=0.6666666666666666;
```

Listing B.1 Modèle système différentiellement confidentiel

```
module DataProvider
    s0: [0..1];
        DATA_x : [DATA_MIN..DATA_MAX];  //datapoint1 , datapoint2 and datapoint3
        [] (s0=0)  -> (s0'=1) & (DATA_x'=0) ;
        [] (s0=0)  -> (s0'=1) & (DATA_x'=1) ;
        [] (s0=0)  -> (s0'=1) & (DATA_x'=2) ;
        [] (s0=0)  -> (s0'=1) & (DATA_x'=3) ;
        [] (s0=0)  -> (s0'=1) & (DATA_x'=4) ;
        [] (s0=0)  -> (s0'=1) & (DATA_x'=5) ;
        [] (s0=0)  -> (s0'=1) & (DATA_x'=6) ;
        [] (s0=0)  -> (s0'=1) & (DATA_x'=7) ;
        [] (s0=0)  -> (s0'=1) & (DATA_x'=DATA_MAX) ;
    [datapoint] (s0=1)  -> (s0'=1-s0);
endmodule

module DataHolder
        s    : [0..5] init 0 ;
        c    : [0..t] init 0 ;
        RR   : [0..N] init 0 ; //randomized responses
        count :[0..N] init 0;  //  value of counting query
        n    : [0..N]  init 0;
        data1 : [DATA_MIN..DATA_MAX] init -1 ;  //datapoint1 , datapoint1 and datapoint3
        data2 : [DATA_MIN..DATA_MAX] init -1 ;
        data3 : [DATA_MIN..DATA_MAX] init -1 ;

        // input : datapoint
        [datapoint] (s=0)  -> (s'=1);

        []  (s=1) & (c=0)  -> (s'=0) & (data1' =DATA_x)  & (n'=mod(n+1,N+1));
        []  (s=1) & (c=1)  -> (s'=0) & (data2' =DATA_x)  & (n'=mod(n+1,N+1));
        []  (s=1) & (c=2)  -> (s'=0) & (data3' =DATA_x)  & (n'=mod(n+1,N+1));

        []  (s=1) & c=0 & (data1 > -1) -> (s'=0) ;
        []  (s=1) & c=1 & (data2 > -1) -> (s'=0) ;
        []  (s=1) & c=2 & (data3 > -1) -> (s'=0) ;

        //input :query
        [query] (s=0) -> (s'=2);
        [] (s=2) -> (s'=3) & (n'=exp_n) & (count'=mod(exp_count,N));

        [] (s=3) & (n=0)  & (count = 0 )->  P0_0_0 :(RR'=0)  & (s'=4)  ;

        [] (s=3) & (n=1)  & (count = 0 )->  P1_0_0 :(RR'=0)  & (s'=4) +   P1_1_0 :(RR'=1)  & (s'=4) ;
        [] (s=3) & (n=1)  & (count = 1 )->  P1_0_1 :(RR'=0)  & (s'=4) +   P1_1_1 :(RR'=1)  & (s'=4) ;

        [] (s=3) & (n=2)  & (count = 0 )->  P2_0_0 :(RR'=0)  & (s'=4) +  P2_1_0 :(RR'=1)  & (s'=4) +   P2_2_0 :(RR'=2)  & (s'=4) ;
        [] (s=3) & (n=2)  & (count = 1 )->  P2_0_1 :(RR'=0)  & (s'=4) +  P2_1_1 :(RR'=1)  & (s'=4) +   P2_2_1 :(RR'=2)  & (s'=4) ;
        [] (s=3) & (n=2)  & (count = 2 )->  P2_0_2 :(RR'=0)  & (s'=4) +  P2_1_2 :(RR'=1)  & (s'=4) +   P2_2_2 :(RR'=2)  & (s'=4) ;

        [] (s=3) & (n=3)  & (count = 0 )->  P3_0_0 :(RR'=0)  & (s'=4) +  P3_1_0 :(RR'=1)  & (s'=4) +  P3_2_0 :(RR'=2)  & (s'=4) +
              P3_3_0 :(RR'=3)  & (s'=4) ;
        [] (s=3) & (n=3)  & (count = 1 )->  P3_0_1 :(RR'=0)  & (s'=4) +  P3_1_1 :(RR'=1)  & (s'=4) +  P3_2_1 :(RR'=2)  & (s'=4) +
              P3_3_1 :(RR'=3)  & (s'=4) ;
        [] (s=3) & (n=3)  & (count = 2 )->  P3_0_2 :(RR'=0)  & (s'=4) +  P3_1_2 :(RR'=1)  & (s'=4) +  P3_2_2 :(RR'=2)  & (s'=4) +
              P3_3_2 :(RR'=3)  & (s'=4) ;
        [] (s=3) & (n=3)  & (count = 3 )->  P3_0_3 :(RR'=0)  & (s'=4) +  P3_1_3 :(RR'=1)  & (s'=4) +  P3_2_3 :(RR'=2)  & (s'=4) +
              P3_3_3 :(RR'=3)  & (s'=4) ;

        []  (s=4) -> (s'=5) & (c'=mod(c+1,t+1)) & (count'=0) & (RR'=0);
        []  (s=5) & (c=0) -> (s'=0) & (data1' =-1) & (n'=exp_n) ;
        []  (s=5) & (c=1) -> (s'=0) & (data2' =-1) & (n'=exp_n) ;
        []  (s=5) & (c=2) -> (s'=0) & (data3' =-1) & (n'=exp_n) ;
endmodule

module DataExaminer
       s2: [0..1];
       [] s2=0 -> (s2'=1-s2);
       [query] s2=1 -> (s2'=1-s2);
endmodule

// labels and formulae for property specification

formula exp_n= (data1  > -1   ?1:0)+(data2 > -1   ?1:0)+(data3 > -1    ?1:0) ;
formula exp_count=   (data1  > k   ?1:0)+(data2 > k   ?1:0)+(data3 > k    ?1:0) ; // counting query
formula Threshold=floor(pow(n,0.5)); // Threshold
formula noise=(RR - count > 0 ? RR - count : count -RR ) ;
formula AboveThresold=(RR - noise > Threshold ? true :false ) ;
formula UnderThresold=(RR - noise < Threshold ? true :false ) ;

label "secret1" = count=2 & n=3 ;
label "secret2" = count=3 & n=3 ;
label "observable1" = RR=2 & count=2 & n=3 ;
label "observable2" = RR=2 & count=1 & n=3 ;
label "Obs_Sec1" = RR=2 & count=1 & n=3 & t=2  ;
label "Obs_Sec2" = RR=2 & count=2 & n=3 & t=2  ;

// rewards
rewards "loss_privacy"
    [query] AboveThresold : 1 ;
endrewards

rewards "Cost_privacy"
    [query] UnderThresold : 1 ;
endrewards

rewards "Query"
    [query] true : 1;
endrewards
```

Listing B.2 Modèle système différentiellement confidentiel

```
// The Truncated Geometric Mechanism
//when i is the real value, report j with probability:
//p(j|i) = cj * alpha^-|i-j|  where  :
//cj=alpha/(alpha+1)   if j=0 or j=n
//cj=alpha-1/(alpha+1) if   0 < j < n
//alpha  > 1 is a security parameter (alpha= exp(epsilon))
//example (alpha = 2, n = 3):

// [2/3  1/6  1/12 1/12
//  1/3  1/3  1/6  1/6
//  1/6  1/6  1/3  1/3
//  1/12 1/12 1/6  2/3 ]


#const N#
#const alpha #



mdp

const int N = #N#;
const double alpha = #alpha#;

global RR :[0..N] init 0 ; //randomized responses
global n :[0..N] init  0;
global count :[0..N] init 0;



const double P0_0_0=1 ;

#for k=1:N#
#for i=0:k#
#for j=0:k#
const double P#k#_#j#_#i#=#(j = 0 | k = j ? (alpha/(alpha+1))*(1/pow(alpha,max(i,j)-min(i,j))) :
      ((alpha-1)/(alpha+1))*(1/pow(alpha,max(i,j)-min(i,j))) )#;
#end#
#end#

#end#


module DataProvider
    s1: [0..1];
    [] true -> (s1'=1-s1);
endmodule

// DataHolder == SysDP

module DataHolder
    s1: [0..1];

[] (s=0) & (n=0)  & (count = 0 )->  P0_0_0 :(RR'=0)  & (s'=1)  ;

#for k=1:N#
#for i=0:k#
[] (s=3) & (n=#k#)  & (count = #i# )-> #for j=0:k-1# P#k#_#j#_#i# :(RR'=#j#)  & (s'=4) + #end#  P#k#_#k#_#i# :(RR'=#k#)  & (s'=4) ;
#end#

#end#

endmodule


module DataExaminer
    s1: [0..1];
    [] true -> (s1'=1-s1);
endmodule
```

Listing B.3 Génération automatique du modèle différentiellement confidentiel