Titre: Title:	Architecture de sécurité des données basée sur HSM et la blockchain
Auteur: Author:	Marc Dib
Date:	2024
Type:	Mémoire ou thèse / Dissertation or Thesis
Référence: Citation:	Dib, M. (2024). Architecture de sécurité des données basée sur HSM et la blockchain [Thèse de doctorat, Polytechnique Montréal]. PolyPublie. https://publications.polymtl.ca/60994/

POLYTECHNIQUE MONTRÉAL
affiliée à l'Université de Montréal

Architecture de sécurité des données basée sur HSM et la blockchain

MARC DIB
Département de génie informatique et génie logiciel

Thèse présentée en vue de l'obtention du diplôme de *Philosophiæ Doctor*
Génie informatique

Novembre 2024

POLYTECHNIQUE MONTRÉAL
affiliée à l'Université de Montréal

Cette thèse intitulée :

Architecture de sécurité des données basée sur HSM et la blockchain

présentée par **Marc DIB**
en vue de l'obtention du diplôme de *Philosophiæ Doctor*
a été dûment acceptée par le jury d'examen constitué de :

Martine BELLAÏCHE, présidente
Samuel PIERRE, membre et directeur de recherche
Alejandro QUINTERO, membre
Abderrezak RACHEDI, membre externe

DÉDICACE

. . .encore une fois. . .

À ma mère Liliane,

Et à feu mon grand-père Henri,

Qui veillent sur moi, d'ici et de là-haut. . .

REMERCIEMENTS

Je tiens tout d'abord à remercier chaleureusement mon directeur de recherche, **Prof. Samuel Pierre**, professeur à Polytechnique Montréal et directeur du LARIM, pour son encadrement, son soutien et sa confiance lors de cette expérience extraordinaire.

J'aimerais remercier aussi les membres du jury qui ont accepté de lire et d'évaluer cette thèse et de participer à sa défense.

Je tiens ensuite à remercier profondément **Dre. Franjieh El Khoury**, associée de recherche, coordonnatrice du LARIM et superviseure du projet ATISCOM. Son encadrement, ses conseils et son partage d'expérience lors de mon doctorat dépassent ce que j'aurai pu imaginer. Merci de m'avoir soutenu tout au long et pour la lecture critique minutieuse de cette thèse et de tous les articles qui en découlent.

Un grand merci à **Monsieur Sandip Patel** de la compagnie *Entrust* pour m'avoir dédié de son temps pour me présenter en détail leur catalogue de modules HSM et les documents fournis, me permettant de me familiariser davantage avec cette composante primaire de ma recherche.

Mes remerciements vont également à **Monsieur Mostafa Chafi** et toute l'équipe de Flex pour m'avoir donné l'opportunité d'appliquer ma recherche dans l'industrie, m'ayant permis de concrétiser mes travaux et de voir leur impact dans le monde réel.

À l'équipe du LARIM, les moments passés à discuter ensemble autour d'un café à nos bureaux étaient non-seulement divertissants, mais aussi enrichissants.

Je remercie aussi tous mes amis et ma famille, qui m'ont soutenu et supporté pendant toute cette année, qui m'ont redonné confiance en moi lorsque j'en avais besoin.

Enfin, un remerciement plus que spécial à ma mère, mon guide dans ce monde, qui m'a offert tout ce qui est bon et beau dans ma vie. Merci de m'avoir toujours inculqué les bonnes valeurs qui ont fait de moi qui je suis aujourd'hui. Merci de tes constants encouragements, ton soutien, et ton amour.

RÉSUMÉ

L'importance de l'informatique dans la vie quotidienne ne fait qu'augmenter. Entre les téléphones intelligents, les électroménagers intelligents, les automobiles intelligentes et toute autre forme d'objets connectés, le nombre de dispositifs reliés au réseau internet est en forte croissance. Avec cette croissance, la quantité de données générées, transférées et stockées est tout aussi affectée. Ceci crée alors une problématique essentielle liée à la sécurité de ces données.

En effet, les nombreuses expériences connues de vol de données, d'usurpation d'identité ou encore de destruction d'information nous montrent que les données informatiques sont à grand risque. De plus, certains événements récents nous rappellent que parmi ces attaques, un tiers provient de l'intérieur de l'organisation ciblée.

La littérature propose des solutions de sécurité, certaines liées aux attaques internes, d'autres à la distribution des données dans les larges systèmes informatiques. Cependant, ces solutions ont souvent des failles qui les rendent inadéquates. Ainsi, cette thèse propose une architecture de sécurité basée sur les technologies « Hardware Security Module » et Blockchain pour protéger les données sur les serveurs dans des systèmes locaux et distribués.

Ainsi, cette architecture se base sur trois modèles pour atteindre son objectif. Tout d'abord, en harmonie avec la méthodologie de gestion des attaques internes, nous proposons un modèle d'attaques internes qui permet d'identifier les attaques possibles dans une architecture basée sur HSM. La seconde composante de l'architecture est un modèle de détection des attaques internes grâce à des mécanismes de défenses basés sur la cryptographie. Ce modèle se base sur les attaques identifiées par le modèle d'attaques internes afin de proposer quatre mécanismes de défense pour la confidentialité, l'intégrité et la disponibilité des données. Finalement, le dernier modèle relève de la distribution sécurisée des données en proposant un Système de Gestion de Bases de Données (SGBD) distribué basé sur la blockchain et HSM. Ce SGBD distribué fait usage des contrats intelligents pour garantir la distribution sécurisée des données.

L'implémentation de l'architecture proposée sert à mesurer et évaluer les performances selon différents critères tels que le délai de traitement, le temps de détection ou l'achalandage du réseau. Les trois modèles ont été évalués selon des méthodologies bien définies qui permettent de mettre en évidence les caractéristiques de leur implémentation. Pour le premier modèle, la métrique δ-score montre l'impact des attaques sur la sécurité du système, ce qui nous permet de mieux identifier et classer les attaques par leur risque. Pour le second modèle, les

résultats montrent que le délai introduit par les mécanismes de défense est raisonnable. En effet, le mécanisme Nonce-Based Process Authentication introduit un délai moyen de 12%, alors que les mécanismes de Hash-Based Field Integrity et Hash-Based Field Availability introduisent un délai moyen de moins de 50%. Cependant, l'ajout en termes de sécurité est considérable avec un taux de détection d'attaques de 100%. Le temps de détection d'attaques pour le mécanisme Hash-Based Row Availability est raisonnablement faible, soit de l'ordre de 5 minutes pour une base de données avec 1000 entrées. Pour le troisième modèle, les résultats montrent que le temps de traitement des requêtes suit une tendance en escaliers, avec plusieurs requêtes qui se rassemblent au sein d'un même bloc. De plus, le modèle proposé permet la distribution, le chiffrement, la révocation d'accès et la validation des données. Enfin, le protocole d'authentification proposé est robuste et léger sur le réseau.

ABSTRACT

Electronic devices are becoming more and more present with every passing day. Between smartphones, smart appliances, smart cars, and every type of connected device, the number of nodes on the worldwide internet is growing strongly. This growth also affects the amount of data generated, stored, and transferred, bringing new problems related to data and information security. Lately, we have seen many cases of data and identity theft and data destruction, and a third of these events originate from inside the targeted organization.

The literature offers security solutions for insider attacks and data distribution in large cyber systems. However, these solutions are flawed and cannot be used as is. Therefore, we propose in this thesis a security architecture based on the blockchain and Hardware Security Module (HSM) technologies to protect server-side data for local and distributed cyber systems.

This architecture is based on the combination of three models. First, in alignment with the Insider Attack Mitigation Methodology, we propose an insider attack model to identify the possible attacks on an HSM-based architecture. The second component is an insider attack detection model relying on cryptography-based defense mechanisms. This model uses the identified attacks in the first model to propose four defense mechanisms for data confidentiality, integrity, and availability. Finally, the last model focuses on secure data distribution by proposing a Distributed Database Management System (DDBMS) based on blockchain and HSM. This DDBMS uses smart contracts to guarantee secure data distribution.

Implementing the proposed architecture allowed us to measure and evaluate the performances according to different criteria, such as latency, detection time, or network congestion. The three models were evaluated according to specific methodologies to showcase their characteristics. For the first model, the δ-score metric shows the impact of each attack on the system security, which allows us to identify and rank attacks by risk level. For the second model, results show that the introduced latency is manageable. Indeed, the Nonce-Based Process Authentication mechanism introduces a 12% delay, whereas the Hash-Based Field Integrity and Hash-Based Field Availability mechanisms introduce a delay of less than 50%. However, their added value in terms of security is solid, with an attack detection rate of 100%. The attack detection time for the Hash-Based Row Availability mechanism is low, averaging around 5 minutes for a database of 1000 entries. For the last module, results show that the processing time follows a step-like tendency, where several queries are encapsulated in the same block. Moreover, the proposed model allows data distribution, encryption, validation, and access revocation. The proposed authentication protocol is robust and lightweight.

TABLE DES MATIÈRES

LISTE DES TABLEAUX

LISTE DES FIGURES

LISTE DES SIGLES ET ABRÉVIATIONS

3-DES	Triple DES
ACL	Access Control List
AES	Advanced Encryption Standard
ARA	Analyse de Risque avec Adversaire
ASN.1	Abstract Syntax Notation One
AT	Authorization Token
AV	Attack Vector
BaDD	Blockchain as a Distributed Database
BAN	Burrows–Abadi–Needham
BFT	Byzantine Fault Tolerance
CIA	Confidentiality, Integrity, and Availability
CIS	Center for Internet Security
CSIRO	Commonwealth Scientific and Industrial Research Organisation
DBMS	Database Management System
DDB	Distributed Database
DDBMS	Distributed Database Management System
DDoS	Distributed Denial of Service
DES	Data Encryption Standard
DNS	Domain Name System
DNSSEC	Domain Name System Security
ECC	Elliptic Curve Cryptography
ECDH	Ellipti–Curve Diffie–Hellman
EM	Employé Malveillant
ESDB	Enhanced StealthDB
FCFS	First-Come First-Served
HBFA	Hash-Based Field Availability
HBFI	Hash-Based Field Integrity
HBRA	Hash-Based Row Availability
HSM	Hardware Security Module
HSM-BSN	HSM-Based Security Node
HSM-BSN IAP	HSM-BSN Identification and Authentication Protocol
IoT	Internet of Things
IP	Internet Protocol

ITCF	Insider Threat Cybersecurity Framework
JSON	Javascript Object Notation
KSU	Key Storage Unit
LMK	Local Master Key
MD5	Message-Digest 5
MitM	Man-in-the-Middle
MMSE	Minimum Mean Square Error
NBPA	Nonce-Based Process Authentication
NFC	Near Field Communication
NIP	Numéro d'Intentification Personnel
ORM	Object Relational Mapping
OSW	Optimized Sliding Window
P2P	Peer-to-Peer
PBFT	Practical Byzantine Fault Tolerance
PBKDF2	Password-Based Key Derivation Function 2
PKCS	Public Key Cryptographic Standard
PoA	Proof of Authority
PoW	Proof of Work
QRC	Quick Response Code
RC4	Rivest Cipher 4
RC5	Rivest Cipher 5
RE	Rogue Employee
RNG	Random Number Generator
RSA	Rivest–Shamir–Adleman
RSM	Replicated State Machine
SBM	Système Bancaire Mobile
SC	Smart Contract
SCA	Side-Channel Attack
SFM	Système de Finances Mobiles
SGBD	Système de Gestion de Bases de Données
SHA	Secure Hash Algorithm
SIM	Subscriber Identification Module
SPM	Système de Paiement Mobile
SQL	Structured Query Language
SSK	Symmetric Session Key
TAG	Trusted Administrator Group

TCP	Transmission Control Protocol
TFA	Two-Factor Authentication
TLS	Transport Layer Security
VPN	Virtual Private Network
WAP	Wireless Application Protocol
ZKP	Zero Knowledge Proofs

CHAPITRE 1 INTRODUCTION

L'informatique est un domaine qui a grandement révolutionné le traitement et la gestion des données. Initialement conçue pour accélérer les calculs mathématiques, elle est vite devenue, en association avec le domaine de télécommunications, un puits d'information, une source de divertissement et un moyen de communication entre les utilisateurs. Deux des aspects de l'informatique, très utilisés aujourd'hui, sont l'interface humain-machine et la mobilité qui permettent de remplacer un grand nombre d'interactions entre les humains. Le rapport annuel de Cisco pour les années 2018 à 2023 [1] montre que, parmi les 25 milliards de dispositifs informatiques dans le monde en 2021, 11 milliards, soit presque la moitié, sont des dispositifs mobiles personnels tels que les téléphones intelligents, les tablettes et les ordinateurs portables. Afin de soutenir ces dispositifs, plus de 5 milliards d'abonnements à des opérateurs de téléphonie mobile ont été enregistrés durant cette même année.

De nos jours, une personne peut effectuer toutes ses opérations bancaires, qu'elles soient basiques ou avancées, à partir de son ordinateur ou téléphone intelligent [2]. Le commerce mobile, connu sous le nom de e-commerce, permet également aux utilisateurs d'acheter ou de vendre des biens et des services à partir du confort de leur domicile [3]. Tout cela se fait grâce aux nombreuses connexions et applications qui régissent aujourd'hui le monde de l'informatique. De plus, la pandémie mondiale de la COVID-19 a permis à la numérisation de se démarquer davantage et d'affirmer son importance dans la vie quotidienne. En effet, avec les nombreux confinements qui ont été imposés dans différents pays, les domaines tels que la banque informatique et le e-commerce ont grandement gagné en dominance sur le marché.

De telles avancées technologiques demandent une considération de leur sécurité, soit la sécurité des serveurs d'applications de façon générale et en particulier des systèmes financiers, qui sont parmi les systèmes les plus critiques. Ces considérations doivent être étudiées par rapport aux dangers et menaces actuels, mais sans oublier également les menaces d'un futur proche comme la cryptanalyse quantique. De plus, ces considérations doivent prendre en compte les divers types de systèmes comme les systèmes locaux et les systèmes distribués, ainsi que les divers types de menaces, comme les menaces externes et les menaces internes.

1.1 Définitions et concepts de base

Dans cette section, nous définissons les concepts élémentaires qui permettent une meilleure compréhension de la thèse. Les thématiques abordées sont les systèmes de finance mobiles,

la cryptographie, le paradigme de la blockchain et la distribution des données dans les larges systèmes informatiques.

1.1.1 Systèmes de finance mobiles

Afin de comprendre l'enjeu de sécurité, il est essentiel de s'immerger dans le contexte qui justifie ce présent travail. L'avancée technologique donne accès aujourd'hui à des Systèmes de Finance Mobiles (SFM) qui permettent de gérer les fonds et les transactions bancaires sans avoir recours à des méthodes plus traditionnelles, comme l'argent liquide ou le passage à un guichet bancaire. Un SFM peut être divisé en deux sous-catégories : les systèmes de paiement mobiles et les systèmes bancaires mobiles.

Définitions

Un Système de Paiement Mobile (SPM) est un système qui permet l'utilisation d'une carte à puce ou d'un terminal mobile (i.e., un téléphone intelligent) pour émettre une transaction monétaire [4, 5]. Ces terminaux entrent généralement en contact avec un lecteur spécialisé qui permet d'effectuer la transaction. De nombreuses technologies existent aujourd'hui pour assurer ces transactions, comme le « Wireless Application Protocol » (WAP), le « Quick Response Code » (QRC) ou le « Near Field Communication » (NFC) [6].

D'autre part, un Système Bancaire Mobile (SBM) permet l'approvisionnement en fonds ou le retrait d'un compte bancaire, ainsi que toute action financière sur ledit compte à travers une application mobile ou Web [7]. Ces deux systèmes constituent les sous-ensembles des Systèmes de Finance Mobiles.

Enjeux de sécurité dans les SFM

Les SFM sont aujourd'hui le centre d'intérêt de nombreux chercheurs d'un aspect légal (i.e., régulations et standardisation) et d'un aspect technologique (i.e., sécurité et architecture) [8]. En effet, les SFM font face à de nombreuses menaces qui mettent à risque la sécurité des informations des utilisateurs de ces services. Ces menaces sont dispersées sur l'ensemble des nœuds qui composent le système [9] tels que les serveurs du système [10, 11], les canaux et éléments intermédiaires du réseau [12] et les terminaux des utilisateurs [13, 14].

Les problèmes que nous retrouvons s'intègrent au paradigme de la confidentialité, intégrité et disponibilité « Confidentiality, Integrity and Availability » (CIA) et ses extensions [15]. L'authentification est également le centre de nombreuses recherches, notamment les méthodes d'authentification biométriques pour la protection des données critiques [16].

D'autres aspects de la CIA se retrouvent également dans la protection contre les maliciels sur les terminaux mobiles. En effet, ces maliciels ont la capacité d'enfreindre à la confidentialité, l'intégrité ou la disponibilité des données en écoutant, modifiant ou interceptant les informations qui transitent par le nœud terminal [13, 17, 18].

L'intérêt accru que les attaquants portent aux SFM fournit une justification et une motivation pour chercher à améliorer la sécurité des SFM.

1.1.2 Cryptographie

Les applications utilisent aujourd'hui la cryptographie pour assurer la sécurité des informations. Nous distinguons trois familles d'opérations cryptographiques [19] : les opérations symétriques, les opérations asymétriques et les opérations à sens unique.

Définitions et notations

Afin de détailler le fonctionnement et les propriétés des algorithmes cryptographiques, nous définissons les ensembles et concepts suivants [20] :

- Soit $\mathbb{B}$ l'ensemble de valeurs binaires que peut prendre un bit, soit $\{0, 1\}$;
- Soit $\mathbb{B}^n$ l'ensemble de mots binaires de taille n, soit $\{0, 1\}^n$;
- Soit $\mathbb{B}^n$ l'ensemble de mots binaires de taille quelconque ;
- L'impraticabilité se définit comme l'impossibilité d'effectuer une certaine opération dans un temps raisonnable [21].

Afin d'unifier les représentations dans ce manuscrit, nous adoptons les notations suivantes :

- $\{x\}_K$ représente le résultat du chiffrement (symétrique ou asymétrique) de x au moyen de la clé K ;
- $\{x\}_K^{-1}$ représente le résultat du déchiffrement (symétrique ou asymétrique) de x au moyen de la clé K ;
- $x|y$ représente le résultat de la concaténation des vecteurs x et y.

Algorithmes de chiffrement symétriques

Un algorithme symétrique nécessite une clé unique qui sera utilisée à la fois pour le chiffrement et le déchiffrement des données, d'où la notion de symétrie [22]. Nous comptons parmi eux les algorithmes « Rivest Cipher 4 » (RC4), « Rivest Cipher 5 » (RC5), « Data Encryption Standard » (DES), « Triple DES » (3-DES) et « Advanced Encryption Standard » (AES), ce dernier étant le plus récent et le plus sécurisé [23, 24].

Unicité de la clé : Soit K une clé cryptographique utilisée pour le chiffrement symétrique, nous définissions f_K comme la fonction représentant un algorithme de chiffrement symétrique utilisant la clé K dans l'équation 1.1.

$$f_K : \begin{array}{ccc} \mathbb{B}^\infty & \rightarrow & \mathbb{B}^\infty \\ x & \rightarrow & \{x\}_K \end{array} \tag{1.1}$$

Nous définissons également f_K^{-1} comme la fonction représentant un algorithme de déchiffrement symétrique utilisant la clé K dans l'équation 1.2.

$$f_K : \begin{array}{ccc} \mathbb{B}^\infty & \rightarrow & \mathbb{B}^\infty \\ x & \rightarrow & \{x\}_K^{-1} \end{array} \tag{1.2}$$

Un algorithme de chiffrement est dit symétrique si $f_K^{-1}(f_K(m)) = m$ [22]. En d'autres termes, la clé utilisée pour le chiffrement devra être préalablement partagée entre les utilisateurs autorisés pour permettre le déchiffrement.

Problématique des algorithmes symétriques : Les algorithmes de chiffrement symétriques soulèvent la problématique de partage de la clé [25]. En effet, peu importe le niveau de sécurité du chiffrement, si le transfert de la clé est compromis, la confidentialité du message le sera aussi. Les algorithmes de chiffrement symétriques ne prennent pas en compte la sécurité de l'échange de la clé symétrique : il s'agit d'un processus qui devra être établi à part.

Algorithmes de chiffrement asymétriques

La cryptographie asymétrique est définie par la norme « Public Key Cryptographic Standard » (PKCS) [26]. Un algorithme asymétrique nécessite deux clés : une clé privée définie par le huitième groupe de normes PKCS (PKCS #8) et une clé publique. La clé privée devra être fortement gardée par son propriétaire pour éviter les attaques par usurpation. Le principe est de chiffrer le message avec une clé et de le déchiffrer par une autre [27]. Ces algorithmes sont vastement utilisés aujourd'hui pour assurer un échange sécuritaire des clés de sessions dans le protocole « Transport Layer Security » (TLS) [28–31].

Dualité de la clé : Soit (PK, PK^{-1}) le couple de clés respectivement publique et privée utilisé pour le chiffrement asymétrique, nous définissions g_K comme la fonction représentant un algorithme de chiffrement et de déchiffrement symétrique utilisant la clé $K \in \{PK, PK^{-1}\}$ dans l'équation 1.3.

$$g_K : \begin{array}{ccc} \mathbb{B}^\infty & \rightarrow & \mathbb{B}^\infty \\ x & \rightarrow & \{x\}_K \end{array} \tag{1.3}$$

Un algorithme de chiffrement est dit asymétrique si $g_K^{-1}(g_K(m)) = g_K(g_K^{-1}(m)) == m$. En d'autres termes, si un mot binaire est chiffré à l'aide de PK^{-1}, il doit être déchiffré avec PK et inversement.

Exemples d'algorithmes de chiffrement asymétriques : Les deux algorithmes de chiffrement asymétriques les plus couramment utilisés sont Rivest–Shamir–Adleman (RSA) [32] et la cryptographie par courbes elliptiques « Elliptic Curve Cyptography » (ECC) [33].

L'algorithme RSA se base sur l'arithmétique et les puissances entières [32]. En effet, un système cryptographique de type RSA est défini par la sélection de deux nombres premiers p et q. Ensuite, le modulo N correspond au produit $p \times q$, la clé privée S et la clé publique P sont définies telles que :

$$S \times P \equiv 1 \ [mod \ (p-1)(q-1)] \tag{1.4}$$

Un message quelconque m est chiffré par la clé publique donnera comme résultat $c = m^P[N]$. Le déchiffrement de c par la clé privée donnera comme résultat $d = c^S[N] = m^{P \times S}[N] = m$.

Comme P et N sont publiques, la connaissance de S permet à l'attaquant de déchiffrer tous les messages. Une méthode de cryptanalyse de l'algorithme RSA est de déterminer la factorisation de N [34]. En soi, il s'agit de déterminer quels sont les entiers p et q qui constituent N. Ce problème de factorisation est actuellement un problème impraticable à résoudre. Cependant, si un attaquant détermine p et q, il sera capable de trouver S à partir de l'équation 1.4.

Une autre méthode de cryptanalyse se base sur la connaissance d'un couple (m, c) où m représente le message déchiffré et c le message chiffré [34]. En effet, comme nous savons que $c^S = m[N]$ et que c,m et N sont connus, la méthode de logarithme discret a pour but de résoudre cette équation sur S. Ainsi, par une seule fuite d'information, l'attaquant peut déterminer la clé secrète et déchiffrer toute communication passée ou future qui utilise cette clé. Cependant, tout comme la factorisation d'entiers, ce problème mathématique est impraticable à résoudre.

Ainsi, l'algorithme RSA est basé sur deux problèmes réputés en mathématique discrèt [32] : la factorisation d'entier et le logarithme discret. La sécurité de RSA dépend entièrement de l'impraticabilité de ces deux problèmes.

L'algorithme ECC est basé sur le problème de logarithme discret des courbes elliptiques, dont la cryptanalyse s'évalue dans l'ordre exponentiel. L'ECC est considérée comme plus performante que RSA, car son impraticabilité est plus puissante.

Le protocole des courbes elliptiques Diffie-Hellman « Elliptic-curve Diffie-Hellman » (ECDH) est un protocole de partage confidentiel de clés défini par la norme PKCS #3 [31, 35, 36]. Comme tout algorithme asymétrique, ECDH se base sur un couple de clés privée et publique.

Les deux parties, Alice (PK_A, PK_A^{-1}) et Bob (PK_B, PK_B^{-1}), décident d'une courbe elliptique, c'est-à-dire un ensemble de paramètres, en particulier la base de la courbe G et un grand nombre premier p [31]. L'algorithme ECDH permet de générer les clés de la façon suivante [31, 35] :

Étape 1 — Les clés privées sont générées à l'aide de G et p conformément aux équations 1.5 et 1.6 :

$$PK_A^{-1} = G^{PK_A} \ [mod\ p] \tag{1.5}$$

$$PK_B^{-1} = G^{PK_B} \ [mod\ p] \tag{1.6}$$

Étape 2 — La clé de secrète SSK est générée à l'aide de G et des deux clés publiques PK_A et PK_B conformément à l'équation 1.7 :

$$SSK = G^{PK_A \times PK_B} \ [mod\ p] \tag{1.7}$$

Étape 3 — En combinant les équations 1.5, 1.6 et 1.7, nous obtenons le résultat suivant pour SSK :

$$SSK = \begin{array}{rcl} (G^{PK_A})^{PK_B} \ [mod\ p] & = & (PK_A^{-1})^{PK_B} \ [mod\ p] \\ (G^{PK_B})^{PK_A} \ [mod\ p] & = & (PK_B^{-1})^{PK_A} \ [mod\ p] \end{array}$$

Ainsi, Alice et Bob, dont chacun dispose de sa clé privée et de la clé publique de l'autre, peuvent calculer séparément la clé secrète et trouver le même résultat.

Applications des algorithmes asymétriques : Il existe aujourd'hui diverses applications des algorithmes asymétriques comme le partage de clé de session ou la signature électronique.

En considérant que PK est publiquement distribuée et que PK^{-1} reste inconnue de tous sauf de son propriétaire, les algorithmes de chiffrement asymétriques peuvent être utilisés pour partager des clés symétriques de façon confidentielle [30] et garantir l'intégrité d'un échange

grâce à la signature électronique [37].

Partage confidentiel de clés de session : Soit Alice (A) qui souhaite envoyer une clé de session symétrique (SSK) à Bob (B) qui dispose d'un couple de clés publique et privée (PK_B, PK_B^{-1}). Un protocole simple d'échange se résume à la méthode suivante [30] :

1. Alice génère SSK ;
2. Alice chiffre SSK avec la clé publique de Bob PK_B : $m = g_{PK_B}(SSK)$;
3. Alice transmet à Bob le message m ;
4. Bob déchiffre m avec sa clé privée PK_B^{-1} : $g_{PK_B^{-1}}(m) = g_{PK_B^{-1}}(g_{PK_B}(SSK)) = SSK$;
5. Bob dispose maintenant de la même clé SSK qu'Alice.

De cette façon, seul Bob qui dispose de PK_B^{-1} sera capable de déchiffrer m pour retrouver SSK.

Signature électronique de messages : Soit Alice qui dispose d'un couple de clés publique et privée (PK_A, PK_A^{-1}) et qui souhaite envoyer un message m à Bob. Bob souhaite vérifier que le message envoyé par Alice n'a pas été altéré en cours de chemin. Un protocole simple de signature électronique se résume à la méthode suivante [37] :

1. Alice rédige le message m ;
2. Alice génère une signature de m en le chiffrant avec sa clé privée PK_A^{-1} : $s = g_{PK_A^{-1}}(m)$;
3. Alice transmet le couple (m, s) à Bob ;
4. Bob déchiffre la signature s en utilisant la clé publique d'Alice PK_A : $m' = g_{PK_A}(s)$;
5. Bob vérifie que $m' = m$ pour assurer que le message m correspond à la signature s.

De cette façon, seule Alice qui dispose de PK_A^{-1} aurait été capable de générer la signature s du message m. Si m a été altéré en cours de chemin, l'attaquant n'aurait pas pu générer la signature s' correspondant au nouveau message.

Fonctions de hachage

Les fonctions de hachage représentent des fonctions à sens unique [38]. Une des propriétés essentielles de ces fonctions est leur aspect chaotique. En effet, une légère perturbation en entrée cause une grande modification de la sortie, rendant la marche arrière par déduction très difficile.

— Soit $h_n : \mathbb{B}^\infty \to \mathbb{B}^n$ une fonction de hachage qui réduit un mot binaire de taille quelconque à un mot binaire de taille n.

Parmi les fonctions de hachage, nous retrouvons l'algorithme « Secure Hash Algorithm »

(SHA) et ses différentes versions SHA0, SHA1 et SHA2 [39] et l'algorithme « Message-Digest 5 » (MD5) [40].

Afin d'être considérée comme sécurisée, une fonction de hachage doit vérifier trois propriétés [41–43] : la résistance préimage, la seconde résistance préimage et la résistance aux collisions.

La résistance préimage correspond au caractère à sens unique de la fonction de hachage [41, 44]. En effet, en choisissant une valeur $y \in \mathbb{B}^n$, elle assure qu'il est impraticable de trouver une valeur $x \in \mathbb{B}^\infty$ telle que $h_n(x) = y$. Dans la pratique, il faudrait que n soit suffisamment grand pour satisfaire cette propriété. Une valeur minimale était $n = 80$ en 2004 [41]. Aujourd'hui, les fonctions de hachage utilisent des valeurs de $n = 256$ ou plus [45].

La seconde résistance préimage [42] stipule qu'en choisissant une valeur de $x \in \mathbb{B}^\infty$ et $y \in \mathbb{B}^n$, $y = h_n(x)$, il est impraticable de trouver une valeur $x' \in \mathbb{B}^\infty$ de sorte que $h_n(x') = y$. Cette propriété est aussi connue sous le nom de « résistance faible aux collisions ».

La résistance aux collisions [43] stipule qu'il est impraticable de trouver $x \in \mathbb{B}^\infty$ et $x' \in \mathbb{B}^\infty$ de sorte que $h_n(x) = h_n(x')$.

Applications des fonctions de hachage : Les fonctions de hachage disposent de plusieurs applications diverses dans le domaine de la cryptographie. Parmi ces applications, nous pouvons citer [46, 47] : la signature électronique améliorée, la randomisation et preuves de travail.

Signature électronique améliorée : Un majeur inconvénient des algorithmes asymétriques est la complexité en temps et en ressources de l'opération avec la clé privée [46].

Les fonctions de hachages peuvent être utilisées pour réduire la taille des messages à signer [38, 46]. Ainsi, dans le protocole de signature électronique, Alice ne signera pas m, mais plutôt $h_n(m)$. Ceci permet de réduire l'entrée de f à $\mathbb{B}^n$ au lieu de $\mathbb{B}^\infty$.

Cette modification diminue grandement la complexité de l'étape 3. Bob, quant à lui devra hacher le message reçu et le comparer au hachage signé.

Randomisation : Des technologies nouvelles comme la blockchain [48] se basent en grande partie sur les fonctions de hachage, en particulier leur aspect chaotique [47]. Les détails de ces applications seront exposés à la Section 1.1.3.

Attaques connexes sur la cryptographie

Les systèmes cryptographiques sont sujets à des attaques par cryptanalyse, c'est-à-dire des attaques qui visent les vulnérabilités des algorithmes cryptographiques. Cependant, ces systèmes sont également vulnérables à des attaques qui ne sont pas reliées directement aux algorithmes cryptographiques, comme le vol de clé ou les attaques « side-channel » [49, 50].

Les algorithmes de chiffrement symétrique et asymétrique se basent sur la possession d'une clé secrète ou d'une clé privée respectivement qui doivent être protégées contre le vol et la fuite [25]. Les clés des algorithmes symétriques peuvent être compromises lorsque les dérivations de clés sont dites faibles selon Pudovkina [51]. Les clés dérivées pourraient être prédites par l'attaquant. Quant aux algorithmes asymétriques tels que RSA, certains dispositifs à capacité limitée sont contraints d'utiliser de petites valeurs pour les exposants RSA, ce qui les rend vulnérables à des vols de clés [34, 52].

Les attaques « Side-Channel Attack » (SCA) sont des attaques basées sur les informations sondées sur le dispositif matériel [53]. En effet, les caractéristiques électroniques des dispositifs peuvent être analysées pour soutirer de nombreuses informations [49, 50]. Parmi les caractéristiques cibles, nous retrouvons la consommation énergétique et la tension électrique du dispositif [54], les ondes électromagnétiques émises par le dispositif [50, 53], l'injection de fautes [55] et le temps de calcul [56].

1.1.3 Blockchain

La technologie « Blockchain » est définie comme un registre décentralisé qui sert à enregistrer des transactions sur différents terminaux [48]. Les propriétés de la blockchain font qu'elle est immutable, résistante à la falsification et aux attaques par rejeu, résiliente et vérifiable [48,57].

Grâce à ses caractéristiques, la blockchain a été réutilisée pour créer le concept de cryptomonnaie [58] qui est une monnaie décentralisée et purement informatique. En opposition aux transferts traditionnels, généralement validés par des entités telles que Visa inc. ou Mastercard inc., les transferts en cryptomonnaie sont validés par les pairs, à travers un ou plusieurs algorithmes de consensus [59].

Bien que les cryptomonnaies représentent la première application de la blockchain (appelée Blockchain 1.0), des applications nouvelles et diverses ont été implémentées grâce à cette technologie. En effet, les contrats intelligents « Smart Contracts » [60] ont été également intégrés à la blockchain, donnant naissance à la Blockchain 2.0. Ces contrats servent à exécuter une transaction spécifique entre deux entités sans avoir besoin d'une entité centralisée de vérification. L'application la plus récente de la blockchain, appelée Blockchain 3.0, touche à

tous les autres domaines possibles d'applications tels que les jeux vidéo, l'internet des objets « Internet of Things » (IoT) ou les chaînes d'approvisionnement [57].

Modèle de référence pour la blockchain

La blockchain est un paradigme très large qui implique plusieurs niveaux. Typiquement, une blockchain peut être décrite en superposant quatre couches [61] :

- La couche réseau qui représente les communications entre les différents nœuds ;
- La couche du consensus qui représente les protocoles utilisés pour atteindre le consensus parmi les nœuds de la blockchain ;
- La couche de réplication « Replicated State Machine » (RSM) qui interprète les transactions et les inscrit dans la blockchain ;
- La couche applicative qui contient les fonctionnalités haut-niveau.

Catégories de Blockchain

Il existe plusieurs catégories de blockchain selon leurs droits d'accès [57]. Les blockchains publiques sont ouvertes à tous les participants qui sont sur le réseau Internet. Ce sont des plateformes du grand public. Les blockchains privées et à consortium ne sont accessibles que par certains participants qui ont les autorisations nécessaires. D'une part, les blockchains privées sont réservées à un usage interne à une organisation. D'autre part, les blockchains à consortium sont légèrement plus ouvertes, dans la mesure où plusieurs organisations peuvent partager la blockchain, mais toujours en faisant usage de permissions d'accès restreintes [57].

Architecture d'une blockchain

Les transactions d'une blockchain sont enregistrées dans des blocs qui sont reliés les uns aux autres de façon linéaire par des méthodes cryptographiques [48, 57]. Les blocs sont créés et entretenus par les différents terminaux, appelés nœuds, qui constituent le réseau blockchain. Afin d'inscrire un bloc dans la chaîne, les nœuds doivent respecter des règles, appelées algorithmes de consensus, qui assurent la synchronisation des différentes réplications de la chaîne [59].

Les blocs utilisent la notion d'arbres de Merkel « Merkle Tree » pour se lier entre eux dans la blockchain [57]. Cependant, au lieu d'avoir une forme d'arborescence, les blocs sont liés par une forme linéaire. En effet, les ramifications causent des effets indésirables et sont donc à éviter [62]. La Figure 1.1 montre un exemple de 4 blocs liés par des hachages cryptographiques.

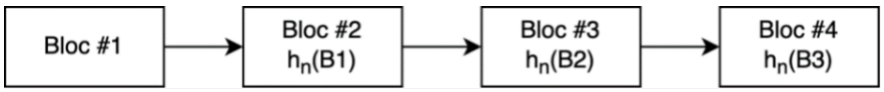

FIGURE 1.1 Chaîne de blocs liés par des hachages cryptographiques

Dans cette figure, $h_n(BX)$ représente le hachage du bloc X, pour X allant de 1 à 4. Ainsi, le bloc #3 contiendra le hachage du bloc #2, qui lui-même contient le hachage du bloc #1. Le seul bloc n'ayant pas d'antécédents est le tout premier bloc de la chaîne, appelé le bloc de genèse.

Blocs de transactions : Un bloc est généralement constitué de deux parties majeures [57, 63] : son entête « block header » et son corps « block body » qui contient l'ensemble des transactions.

L'entête du bloc est responsable de son identification. Ceci inclut différentes informations telles que la version du bloc, la racine de l'arbre de Merkle, un tampon d'horodatage, un nonce et le hachage du bloc parent. Ce dernier permet de lier les blocs entre eux selon la logique des arbres de Merkle [63].

Le corps du bloc contient l'ensemble des transactions [57, 63, 64]. En effet, un bloc peut contenir plus qu'une seule transaction pour avancer plus rapidement dans la validation des transactions sur une blockchain. Ainsi, le corps du bloc indique le nombre de transactions traitées en son sein, la récompense gagnée par le mineur pour ce bloc et les frais des transactions. Certaines applications d'analyse de blockchain indiqueront également la taille du bloc en termes d'octets et la valeur totale des transactions, mais ces champs peuvent simplement être déduits de la lecture des transactions [64].

Nœuds de la blockchain : Un nœud dans la blockchain est un terminal informatique qui peut lire ou écrire des blocs de la chaîne [57, 63]. Ces nœuds se divisent en plusieurs catégories : les nœuds complets, les nœuds miniers et les nœuds à fonctionnalités légères. Chaque catégorie de nœuds dispose de performances différentes pour effectuer des tâches différentes.

Les nœuds complets sont capables d'assurer toutes les fonctionnalités relatives à la blockchain [57, 63]. Ils peuvent stocker l'historique de la chaîne, maintenir le consensus dans le réseau et vérifier les transactions [57].

Les nœuds miniers valident les transactions pour insérer les blocs dans la chaîne [57, 63]. Leurs tâches se résument à suivre les étapes de préparation du bloc telles que les preuves de

travail, les preuves de biens, ou tout autre protocole de consensus choisi pour la chaîne [57].

Algorithmes de consensus : Il existe plusieurs mécanismes pour atteindre le consensus dans une blockchain [59]. Les plus utilisés sont les mécanismes basés sur la « Proof of Work », les mécanismes basés sur la « Proof of Authority » et les mécanismes basés sur la « Byzantine Fault Tolerance ».

Le mécanisme de Proof of Work (PoW) se base sur la résolution d'un problème cryptographiquement complexe pour mériter d'inscrire le bloc voulu dans la chaîne [47]. Ce mécanisme est recommandé dans les blockchain publiques où le nombre de nœuds est élevé. Cependant, il est particulièrement gourmant en ressources.

Le mécanisme de Proof of Authority (PoA) se base sur la désignation d'un ou de plusieurs nœuds autorisés à inscrire des blocs dans la chaîne [ref]. Tout bloc proposé par un autre nœud que les nœuds désigné est refusé par les pairs. La liste des nœuds d'autorité est dynamique et les nœuds d'autorité existants peuvent voter pour conférer ce privilège à un nouveau nœud ou révoquer le privilège d'un nœud qui a été corrompu. Ce mécanisme est plus approprié pour des blockchains privées où l'accès est contrôlé et les ressources de calcul sont limitées.

Le mécanisme de Byzantine Fault Tolerance (BFT) se base sur un vote de masse pour inscrire le bloc [59]. Cependant, l'évolutivité de ce mécanisme est faible car il se base sur l'envoi d'un grand nombre de messages sur le réseau.

Défis connus de la blockchain

Les défis de la blockchain sont répartis sur les différentes couches qui la constituent, soient la couche réseau, la couche de consensus, et la couche de réplication d'état [61].

Défis relatifs à la couche réseau : La blockchain se base sur les communications de type pair à pair « Peer-to-Peer » (P2P) [61]. Ces communications sont établies dans un réseau privé pour les blockchains privées ou semi-privées, et dans un réseau public pour les blockchains publiques.

Les blockchains privées sont sujettes à des attaques internes, car un attaquant interne au réseau peut introduire des nœuds malicieux ou retirer des nœuds honnêtes pour déséquilibrer le consensus [61]. Les blockchains publiques sont sujettes à une large plage d'attaques comme les attaques sur les systèmes de noms de domaines « Domain Name System » (DNS), les attaques de routage ou encore les attaques de déni de service sur la connectivité ou sur les ressources [61].

Défis relatifs à la couche de consensus : Les mécanismes de consensus présentés précédemment comprennent tous des défis et des vulnérabilités.

Les PoW sont des mécanismes très gourmands en énergie, et donc en ressources monétaires [65]. Ils ne sont donc pas appropriés pour toutes les situations. De plus, les PoW ne peuvent également fournir qu'une finalité probabiliste. En d'autres termes, un nouveau bloc dispose d'une probabilité non nulle d'être abandonné si une chaîne concurrente fait surface, ce qui pose un problème de double dépense. Cependant, cette probabilité diminue exponentiellement avec l'ajout de blocs au-dessus de ce bloc [61, 66]. Les PoW ont aussi des failles de sécurité bien connues [61]. Elles sont vulnérables aux attaques des 51%, ce qui décourage l'emploi des PoW dans les chaînes privées où le nombre de nœuds est limité.

Les PoA sont des mécanismes considérablement moins gourmands en énergie que les PoW, cependant, ils sont bien plus rigides et moins décentralisés que les PoW. En effet, l'introduction de nœuds d'autorité limite centralise les opérations de la blockchain autour de ces quelques nœuds sélectionnés. De plus, la possibilité de corruption de ces nœuds d'autorité introduit des risques de manipulation sur la blockchain.

Les BFT sont très peu évolutifs. En effet, les différentes propagations des messages à tous les nœuds rendent la complexité des communications exponentielle [59, 61, 65]. De plus, ils sont vulnérables à l'attaque des 33% [61] où un tiers du réseau peut embrouiller le consensus pour l'ensemble du réseau.

Défis relatifs à la couche de réplication d'état : Dans un contexte de protection des données, la couche de réplication d'état « Replicated State Machine » (RSM) devrait assurer la confidentialité des identités des utilisateurs, ainsi que la confidentialité et l'intégrité des données stockées sur la blockchain [61]. Ceci inclut l'accès et l'altération non autorisée des données par certains nœuds malveillants.

D'une part, les techniques de dé-anonymisation telles que les transactions entachées et l'analyse de flux du réseau permettent de compromettre l'anonymat des utilisateurs [61].

D'autre part, les données sur une blockchain sont généralement inscrites en clair pour permettre la validation des données. Les méthodes de chiffrement usuelles ont de nombreux inconvénients tels le partage de clé, l'immutabilité des données et la révocation d'accès [67]. D'autres méthodes de conservation de la confidentialité existent. Les preuves sans connaissance « Zero Knowledge Proofs » (ZKP) permettent de valider une proposition sur des données sans nécessairement en avoir connaissance [61]. Les algorithmes de chiffrement homomorphique [61] permettent d'effectuer des opérations mathématiques sur un contenu chiffré sans nécessiter son déchiffrement. Cependant, ces méthodes ne sont pas très performantes

dans de grands réseaux.

Une blockchain est, au plus, aussi sécurisée que les algorithmes cryptographiques sur lesquels elle se base. En effet, la cryptomonnaie IOTA a prouvé cette affirmation lorsque sa fonction de hachage, Curl-P, a été brisée par la communauté informatique [68]. Maintenir la robustesse des algorithmes cryptographiques est donc nécessaire au bon fonctionnement de la blockchain.

1.1.4 Distribution des données dans les larges systèmes informatiques

Dans les larges systèmes informatiques, une seule base de données peut ne pas être suffisante pour stocker l'entièreté des données applicatives. Le concept de base de données distribuée « Distributed Database » (DDB) implique plusieurs bases de données reliées sur différents nœuds du réseau [69, 70]. Ces nœuds coopèrent entre eux pour traiter les transactions sur la DDB.

Distribution des données

Il existe deux philosophies de distribution des données [69, 70] : la fragmentation (ou le partitionnement) et la réplication.

Dans le cas de la fragmentation des données, chaque donnée n'est disponible qu'à un seul site physique [69, 70]. Ceci implique moins de redondances et une meilleure gestion de l'espace mémoire alloué à la base de données. Cependant, le manque de redondance implique un manque de résilience. En effet, si une défaillance survient sur ce site, les données seront perdues.

Dans le cas de la réplication des données, les données sont disponibles sur plusieurs sites simultanément [69, 70]. Pour la réplication complète, chaque donnée est disponible sur tous les sites, tandis que pour la réplication partielle, chaque donnée n'est disponible que sur certains sites. La réplication offre une meilleure résilience pour le système en cas de défaillance. Cependant, elle implique une duplication de la même donnée sur plusieurs nœuds du réseau, ce qui signifie que la même donnée prend plus d'espace mémoire que prévu. La mise à jour des données devient également plus complexe, car il faut synchroniser tous les sites afin d'éviter les discordances entre les données [21].

De façon générale, la réplication est plus avantageuse lorsqu'il y a plus d'opérations de lecture que d'opérations d'écriture sur la base de données. Inversement, la fragmentation est plus intéressante si les données sont rarement lues, mais souvent mises à jour.

Défis relatifs à la distribution des données

La distribution des données soulève plusieurs défis. Parmi ces défis, deux sont particulièrement intéressants dans le cadre de cette recherche [69, 70] : la concurrence et la fiabilité.

Les défis liés au contrôle de la concurrence relèvent de la synchronisation des accès simultanés, par exemple lors de deux écritures simultanées et contradictoires sur une même entrée. Ceci relève également de la consistance des données et des effets des transactions.

Les défis liés à la fiabilité soulèvent la question de la résilience du système face aux défaillances. Il peut s'agir de transactions échouées, de défaillances machinales sur un site quelconque ou d'échecs de communications.

1.2 Éléments de problématique

Au vu des défis présentés dans les concepts de la Section 1.1, plusieurs éléments de problématique se soulèvent. Le premier est relatif à la sécurité des données contre les attaquants internes à l'entreprise. En effet, l'architecture initiale proposée se limite aux attaques extérieures et doit être complémentée. Le second est relatif à la sécurité de la blockchain, en particulier contre les attaques sur la couche réseau et sur la confidentialité et l'intégrité des données dans la couche « Replicated State Machine » (RSM). Le troisième est relatif à la sécurité, soit la confidentialité et l'intégrité des données distribuées.

1.2.1 Menaces internes à la sécurité des données

Bien que de nombreuses attaques visent les données à partir de l'extérieur, un employé même de la compagnie peut porter atteinte à la sécurité des données. Ceci peut se faire de façon involontaire ou de façon délibérée [71–73]. Dans le premier cas, il s'agit souvent de négligence de la part des employés et de l'équipe de sécurité de l'entreprise. Le second cas traite la situation où un employé cherche activement à nuire à l'entreprise. Selon son statut dans l'entreprise et son niveau de connaissances de l'architecture du système, il peut représenter une menace plus ou moins importante [71].

Souvent, certains signes sont typiques d'un employé malveillant tels que l'accès à de larges quantités de données, l'exportation non autorisée de données ou encore la recherche de données confidentielles non reliées à leur fonctionnalité. Ces risques peuvent être détectés par une surveillance accrue des activités des employés et peuvent être prévenus par un contrôle d'accès strict [72]. Cependant, plus l'employé malveillant est élevé dans la hiérarchie de l'entreprise, plus il lui est facile de contourner les mesures de sécurité.

Les lacunes de la littérature actuelle surlignent l'absence d'une architecture concrète qui vise à protéger les systèmes informatiques contre les attaques internes.

1.2.2 Lacunes de sécurité de la blockchain

Les études réalisées sur la blockchain montrent qu'il reste encore de nombreux points à traiter pour pouvoir utiliser cette technologie dans des applications où la confidentialité, l'intégrité et la disponibilité sont essentielles. En effet, la blockchain, malgré sa décentralisation, reste sujette à des attaques internes, notamment pour les blockchains privées [61,74]. Des méthodes de contrôle d'accès plus strictes, mais qui n'entravent pas à son évolutivité doivent être développées pour empêcher les diverses attaques sur la couche réseau de la blockchain, comme l'introduction de nœuds malveillants dans une blockchain privée.

D'autre part, les applications qui traitent des données confidentielles ont tendance à éviter cette technologie, car elle n'est pas optimisée pour le chiffrement de données [61, 67]. Cependant, le besoin pour la confidentialité et le besoin pour une technologie décentralisée ne devraient pas être mutuellement exclusifs. Il serait donc important d'augmenter la technologie de blockchain pour assurer la confidentialité au sein de son paradigme.

1.2.3 Prolématiques liées à la distribution des données

Le concept de distribution des données a soulevé dans la littérature des problèmes de confidentialité, d'intégrité et d'authentification. En effet, dans le cas où une application souhaiterait chiffrer ses données distribuées, la gestion des clés n'est pas une tâche évidente. La blockchain a été explorée dans la littérature pour tenter d'apporter une meilleure confidentialité et intégrité aux bases de données distribuées [75, 76]. Cependant, les méthodes proposées ne sont pas complètes et les problématiques de gestion des clés et de chiffrement persistent.

L'analyse de ces défis nous amène à nous poser la question de recherche suivante : Comment mettre en place une stratégie pour sécuriser les données applicatives d'un système informatique face aux attaques ? Plus particulièrement, nous sommes intéressés à découvrir :

1. Comment identifier les vulnérabilités d'un système informatique vis-à-vis d'attaquants intérieurs ?
2. Comment assurer la sécurité des données dans un système informatique contre des attaques internes ?

3. Comment distribuer des données dans un large système informatique sans compromettre leur sécurité ?

1.3 Objectifs de recherche

L'objectif principal de cette recherche est de proposer une architecture de sécurisation basée sur une technologie adaptée et spécialisée pour protéger les données dans les architectures locales et distribuées. De manière spécifique, cette thèse vise à :

1. Concevoir un modèle d'attaques internes pour identifier les vulnérabilités du stockage des données locales ;
2. Proposer un modèle de sécurité pour protéger les données locales contre les menaces intérieures ;
3. Proposer un modèle de sécurité pour protéger les données distribuées contre les menaces intérieures et extérieures ;
4. Implémenter les modèles proposés pour contrer les menaces envers la sécurité des données ;
5. Évaluer les performances des modèles proposés en termes de sécurité et d'efficacité.

1.4 Plan de la thèse

Cette thèse est divisée en 8 chapitres qui répondent à la problématique d'une architecture de sécurité des données basée sur HSM et la blockchain. Dans ce premier chapitre, nous avons exposé le cadre de la thématique, défini les concepts de base, déterminé les éléments de problématiques desquels ont découlé nos objectifs de recherche.

Le chapitre 2 est une revue de littérature critique des méthodes, architectures et solutions qui permettent de sécuriser les données. Une attention particulière sera apportée aux travaux existants qui traitent de la sécurité du côté serveur afin de mettre en évidence leurs lacunes.

Le chapitre 3 expose la méthodologie suivie pour mener à bien ce travail de recherche. Nous mettons également en évidence le lien entre les objectifs et les différents volets du travail de recherche.

Le chapitre 4 est une présentation du premier article « Insider Attack Model Against HSM-Based Architecture » publié dans le journal IEEE Access. Cet article propose un modèle d'attaques internes contre les architectures basées sur HSM, avec une description détaillée des vulnérabilités de ces architectures.

Le chapitre 5 est une présentation du second article « HSM-Based Architecture to Detect Insider Attacks on Server-Side Data », soumis pour publication dans le journal IEEE Transactions on Information Forensics and Security. Cet article propose une architecture composée de 4 mécanismes de défense pour détecter les attaques internes identifiées dans le modèle d'attaques internes. Ces mécanismes de défense sont basés sur la cryptographie et le HSM pour une meilleure sécurité.

Le chapitre 6 est une présentation du troisième article « Secure Distributed Database Management System Based on HSM and Blockchain », soumis pour publication dans le journal IEEE Transactions on Dependable and Secure Computing. Cet article propose un SGBD distribué pour garantir la confidentialité, intégrité, disponibilité, synchronisation, et exactitude des données dans un système distribué. Le SGBD est basé sur la blockchain, à travers des contrats intelligents et des nouveaux types de nœuds, et sur HSM pour une sécurité accrue.

Le chapitre 7 est dédié à une discussion générale des résultats obtenus dans les trois volets de cette thèse. Il met en évidence les points forts et les points faibles de l'architecture proposée.

Finalement, le chapitre 8 apporte une conclusion à la thèse en mettant en valeur les contributions de notre travail, les limitations et les travaux futurs.

CHAPITRE 2 REVUE DE LA LITTÉRATURE

Les concepts abordés au Chapitre 1 comportent un enjeu important en matière de sécurité. Il peut s'agir de la sécurité de l'information, de son transfert ou de son entreposage. De nombreuses propositions ont été trouvées dans la littérature pour répondre à cet enjeu. Dans ce chapitre, nous résumons les solutions, architectures et protocoles proposés par la communauté scientifique dans les domaines du transfert, l'entreposage et la distribution de données ainsi que de la blockchain. De plus, nous exposons les limitations de ces méthodes afin de justifier le besoin pour les travaux de cette thèse.

2.1 Sécurité des terminaux dans les systèmes de finance mobiles

Les terminaux mobiles représentent un grand risque sur la sécurité des données relatives aux Systèmes de Finance Mobiles (SFM). En effet, ces terminaux sont de caractéristiques très variables comme le système d'exploitation et la version du logiciel. Chaque terminal dispose de normes de sécurité propres et tous ne garantissent pas une gestion optimale des attaques informatiques.

2.1.1 Détection de maliciels

Les terminaux mobiles sont vulnérables à l'installation de maliciels comme les logiciels espions « spyware » et les logiciels ayant pour but de voler, altérer, corrompre ou détruire les données de l'utilisateur. Les travaux réalisés par Rodrigo *et al.* [13] ainsi que Fournier *et al.* [18, 77] permettent la détection de logiciels malveillants sur une plateforme Android. À travers ces travaux, le terminal est considéré comme mieux protégé contre les logiciels malveillants.

2.1.2 Intégration de matériel sécurisé au terminal

Les terminaux mobiles disposent d'une carte d'identification d'abonné « Subscriber Identification Module » (SIM) qui permet de stocker les informations essentielles de l'utilisateur par rapport à son abonnement de télécommunications. Ces cartes SIM sont des cartes à puces intelligentes qui peuvent stocker d'autres types d'informations et effectuer quelques opérations simples. Dans les travaux de Keykhaie et Pierre [16, 78, 79], les propriétés de sécurité des cartes SIM ont été utilisées pour entreposer de façon sécurisée les données biométriques des utilisateurs et d'effectuer l'authentification biométrique dans un environnement isolé du matériel standard du téléphone intelligent. Cette isolation permet de protéger les données

et les opérations effectuées contre des maliciels sur le téléphone ainsi que des bugs ou d'une mauvaise gestion de la mémoire dans l'application qui nécessite l'authentification.

2.1.3 Limitations

Bien que ces solutions soient essentielles à la sécurité des SFM, celles-ci sont ciblées sur les terminaux du système uniquement. Ces travaux doivent être complétés par un effort de recherche qui vise la protection des données entreposées sur les serveurs d'applications.

En effet, les serveurs représentent une surface d'attaque très attrayante pour les attaquants. Le risque relatif à la sécurité des données sur un serveur ne concerne pas un seul utilisateur, contrairement aux attaques sur les terminaux, mais plutôt une multitude, voire tous les utilisateurs de l'application

2.2 Sécurité des transferts de données

Les données envoyées entre deux parties communicantes peuvent être écoutées par un individu malveillant si elles ne sont pas chiffrées. Ceci peut se traduire par un utilisateur qui envoie une demande de transfert de fonds sur son application bancaire, ou au serveur bancaire qui envoie la transaction à une autre banque. Toutes ces données qui transitent sont confidentielles et doivent être protégées contre l'écoute, la modification et la réutilisation. La sécurité des données se fait en partie par les protocoles cryptographiques (cf. Section 1.1.2) comme le protocole « Transport Layer Security » (TLS) [28–30].

Tsobdjou *et al.* [12] propose un protocole basé sur la cryptographie de courbes elliptiques pour l'échange de clés et l'authentification mutuelle entre le client et le serveur applicatif. Ce protocole opère de façon sécurisée et efficace. D'autres travaux de Tsobdjou *et al.* [80] protègent le canal de communication contre les attaques de déni de service distribué « Distributed Denial of Service » (DDoS) par une méthode dynamique. Cependant, ces méthodes visent à protéger uniquement le transfert des données, et non leur entreposage sur le serveur.

2.3 Sécurité des données locales sur un serveur contre les attaques externes

Un volet majoritairement exploré dans le domaine de la sécurité des données est la protection contre les attaques externes. Dans ce volet, nous retrouvons des solutions et architectures centrées autour de la détection et la prévention d'intrusion, ainsi que du stockage sécurisé des données.

2.3.1 Architecture basée sur les pare-feux

Les pare-feux sont des éléments réseau qui permettent de réguler le trafic réseau selon des règles d'accès [81]. Murthy *et al.* [82] proposent une approche basée sur la forensique numérique et les pare-feux pour contrer les menaces sur les données. L'utilisation de pare-feux permet de limiter l'exposition des nœuds du réseau interne au réseau externe. Par exemple, une base de données ne doit pas être directement accessible de l'extérieur de l'organisation et son accès doit être fait depuis un processus du réseau interne. Ceci permet une claire séparation entre la zone « interne » et la zone « externe ».

Fleiner *et al.* [83] proposent également une solution de sécurité basée sur les pare-feux. Leurs pare-feux sont spécifiques aux bases de données et empêchent une escalade de privilèges pour une personne provenant de l'extérieur du système de bases de données.

Cependant, les deux solutions proposées ne sont efficaces que contre des attaquants externes. En effet, lorsqu'un attaquant se situe dans le même réseau que les nœuds critiques, il peut accéder aux données sans avoir à passer par les pare-feux, ce qui constitue une vulnérabilité pour la sécurité des données.

2.3.2 Architecture de sécurité des données sur un serveur basée sur HSM

Mousa *et al.* [84] explorent la problématique de sécurité des bases de données en exposant les défis et solutions connus. Parmi les vulnérabilités notées, nous retrouvons la mauvaise configuration des bases de données, les attaques de déni de service, la mauvaise gestion de données critiques, l'exposition publique des données de sauvegarde « backup », et les privilèges excessifs.

Pour répondre à ces vulnérabilités, Dib [85] propose une architecture de sécurité des données sur un serveur d'applications grâce au « Hardware Security Module » (HSM). L'architecture proposée se focalise sur la protection de la confidentialité, de l'intégrité et de la disponibilité des données dans un système informatique. Le principe se base sur le déchargement de toute opération cryptographique du serveur au module HSM pour fournir une meilleure performance ainsi que de profiter d'une grande résistance aux attaques SCA [86]. Il se base également sur une augmentation des capacités du HSM pour permettre à un Système de Gestion de Bases de Données (SGBD) de déchiffrer le contenu de ses colonnes afin d'y appliquer des filtres. Ces travaux ont montré que l'utilisation du HSM est plus sécurisée que les autres méthodes. Cependant, cette architecture ne semble pas protéger les données contre les attaques internes. En effet, aucune politique de contrôle d'accès n'est proposée pour empêcher un attaquant interne d'interférer avec les données.

En nous référant au *White Paper* relatif à l'environnement de sécurité des HSM *nShield* de la compagnie Entrust [87], nous observons plusieurs tentatives pour protéger les données contre les menaces internes.

Les jetons d'autorisation « Authorization Tokens » (AT) permettent d'authentifier et d'autoriser le chargement des clés sur le HSM. Ces AT prennent la forme de cartes à puces intelligentes. Lorsqu'un AT est présenté et vérifié avec la clé chiffrée, le HSM chargera la clé dans son environnement de sécurité pour l'utiliser selon l'accès offert par une liste de contrôle d'accès. Cette méthodologie de chargement des clés est donc assez sécurisée et se retrouve de façon standardisée dans la plupart des HSM commerciaux.

Cependant, une fenêtre de manœuvre pour un employé malveillant existe une fois que la clé est chargée. En effet, si aucun contrôle n'est appliqué sur la façon dont les clés sont utilisées, ceci peut créer des vulnérabilités par rapport à des attaques internes. Par exemple, dans le module de protection des données exposé en [85], un employé malveillant qui accède à la base de données chiffrée peut demander le déchiffrement de son contenu auprès du HSM. Un contrôle d'accès sévère est donc nécessaire au sein du HSM.

Pour réaliser ce contrôle d'accès, les modèles nShield disposent de trois stratégies : la confiance, l'authentification logique, et l'authentification physique.

La stratégie de « confiance » indique que le HSM fait confiance à la connexion établie avec le serveur, et que tout processus venant de ce nœud est autorisé à échanger avec le HSM. Cette méthode est la plus vulnérable contre les attaques internes, car il n'y a aucune restriction. La stratégie « authentification logique » nécessite un mot de passe préétabli entre le serveur et le HSM pour autoriser la communication. Ainsi, une application sur le serveur autorisé qui ne dispose pas de mot de passe valide ne pourra pas communiquer avec le HSM. Cette méthode souffre de l'attaque de l'homme-du-milieu, dans la mesure où une personne malveillante avec les bons accès peut voler le mot de passe d'une application légitime et le donner à un processus non légitime. Finalement, la stratégie « authentification physique » nécessite des cartes à puces spéciales pour l'utilisation des clés. Ainsi, à chaque requête vers le HSM, un consensus de cartes doit être présenté au HSM pour autoriser la communication. Cette méthode est la plus sécurisée des trois, cependant elle souffre d'impraticabilité. En effet, il est peu réaliste de devoir insérer un consensus de carte à puce pour chaque transaction dans le contexte d'un serveur applicatif où plusieurs centaines de transactions par secondes sont nécessaires.

Ces limitations montrent un besoin pour une architecture de sécurité qui protège les données contre les attaques internes.

2.4 Sécurité des données locales sur un serveur contre les attaques internes

Les attaques internes ont fait l'objet de quelques recherches dans la littérature [88, 89]. La méthodologie de travail pour contrer les attaques internes se résume communément en cinq étapes : l'identification, la détection, la protection, la réponse, et la récupération [89, 90]. Chacune de ces étapes est essentielle au bon fonctionnement d'une architecture de sécurité contre les attaques internes.

2.4.1 Identification des attaques internes

La première étape dans la méthodologie de travail contre les attaques internes est l'identification des attaques. En effet, l'identification permet de mieux cerner les vulnérabilités du système afin de concevoir et implémenter les mécanismes de défenses adéquats à la situation.

Joshi *et al.* [91] proposent une méthodologie de modélisation des attaques internes basée sur la théorie de l'Analyse de Risque avec Adversaire (ARA). Cette approche est particulièrement intéressante, car elle suppose la situation comme un jeu entre l'attaquant et le défendeur où le but de chacun est de maximiser son gain. L'étude porte sur les méthodes de prédiction des stratégies de l'attaquant, afin de préparer une meilleure contre-attaque. En résumé, cette méthode se base fortement sur les probabilités, dans la mesure où l'on ne sait pas quelle sera la prochaine tentative de l'attaquant. Cependant, elle fait appel à la mise en place de mesures de sécurité qui peuvent être prises avant et après une attaque. Ceci se fait en fonction de différents facteurs comme la détection de l'attaque et la culture de l'entreprise.

D'autre part, Tukur et Ali [92] ont démontré l'impact des attaques internes dans les systèmes IoT. Leur approche se base sur la conception d'un modèle d'attaques internes avant de développer des mécanismes de défense. Puisque leur approche est centrée sur l'IoT, un paradigme essentiellement différent de celui de cette thèse, les solutions proposées ne sont pas applicables.

Cependant, ces deux travaux démontrent l'impact positif de la définition d'un modèle d'attaques internes préalablement à la conception des mécanismes de défense.

2.4.2 Détection des attaques internes

La détection est la deuxième étape dans la méthodologie de travail contre les attaques internes. Cette détection peut se faire au niveau du comportement des employés, au niveau technique, ou à l'aide de divers outils.

Détection basée sur les comportements de l'employé

La plupart des méthodologies de détection des attaques internes se base sur le comportement des employés malveillants. En effet, les attaques informatiques, comme tout autre crime, se démarquent par un aspect psychologique. Comprendre la psychologie du criminel peut grandement aider à prévenir son attaque [93]. Dans une grande majorité des cas, les attaques informatiques qui ont été commises par des employés malveillants présentent des événements alarmants liés au comportement de l'employé [94]. Ces événements sont remarquables par l'organisation et permettent de prévenir une attaque avant qu'elle n'ait lieu.

Dahmane et Foucher [88] proposent une méthodologie de détection d'attaques internes basée sur l'établissement de profils d'employés. Pour cela, une chaîne de Markov est générée pour chaque employé à partir de ses activités informatiques. Lorsqu'une attaque est suspectée, cette chaîne de Markov est comparée avec la chaîne de Markov qui correspond à la séquence d'événements décrits par l'attaque. Si la similarité est suffisamment élevée, le profil est marqué comme suspect. Cette méthode utilise l'apprentissage intuitif qui est plus efficace que les autres méthodes d'apprentissage automatique. Cependant, comme tout apprentissage, la définition des métriques est un facteur significatif dans la réussite de la détection. Dans l'exemple proposé en [88], les actions recherchées par la chaîne de Markov ne représentent pas, par exemple, les signes alarmants d'une attaque dans le cas d'un employé malveillant qui accède à des données confidentielles sur une base de données. Ceci souligne l'importance d'une identification précise et préalable des attaques internes.

Reinerman-Jones *et al.* [95] explorent de leur côté une autre méthode de détection basée sur le comportement. Dans leur expérience, deux groupes de participants (un groupe honnête et un groupe malveillant) doivent effectuer des tâches assignées sur un logiciel bancaire. Les auteurs mesurent diverses réactions, comme le niveau de stress, lors de l'accomplissement de ces tâches. Les résultats montrent que toute différence comportementale est reliée à une action illégale. Cependant, un entraînement professionnel pourrait permettre à une personne d'éviter ces signes flagrants.

Détection basée sur les aspects techniques

Après considération des inconvénients des méthodes basées sur l'analyse comportementale, il est évident qu'un volet technique pourrait être également implémenté pour complémenter la sécurité. Cependant, jusqu'à récemment, les méthodes techniques étaient peu concluantes dû à la difficulté de supervision et de gestion des accès pour une personne interne au système informatique [94]. Aujourd'hui, nous pouvons diviser ces méthodes de sécurité selon l'aspect

de sécurité qu'elles cherchent à accomplir : la confidentialité, l'intégrité, la disponibilité. Certaines méthodes s'étendent également sur les trois aspects.

Études globales : Duncan *et al.* [96] proposent une combinaison des techniques « Attack Tree » et « Kill Chain » pour détecter les attaques internes sur les services cloud. Tout d'abord, il s'agit de créer un arbre de possibilités qui recense toutes les attaques possibles avec les différentes actions à entreprendre pour réaliser l'attaque. Dans leur méthodologie, le résultat final de l'attaque est inscrit dans la racine de l'arbre, et les étapes sont les feuilles et branchements. Ils proposent ensuite diverses améliorations ciblées sur leur contexte de service cloud. Cependant, la différence de paradigme fait en sorte que leurs améliorations proposées ne correspondent pas à notre situation. De plus, la forte dépendance de leur solution sur une identification préalable des résultats finaux des attaques justifie une étape préliminaire d'identification des attaques internes dans notre architecture.

Egbunike et Ranjendran [97] proposent une solution de base de données négative contre le vol et l'altération de données. Leur solution se base sur l'injection de valeurs secrètes à l'intérieur d'une donnée pour la brouiller et la rendre inaccessible sans ces valeurs secrètes. Ces valeurs peuvent être statiques (i.e., déterminées par un administrateur) ou dynamiques (i.e., déterminées par l'utilisateur). Dans le cas des valeurs statiques, un administrateur malveillant peut accéder aux données, puisqu'il connaît les valeurs secrètes à injecter pour le déchiffrement des données. Dans le cas des valeurs dynamiques, l'utilisateur doit retenir et saisir plusieurs informations supplémentaires, ce qui n'est pas désirable dans une optique de simplifier l'utilisation de l'application. De plus, l'évaluation des auteurs montre que les performances en temps de la base de données négative ne sont pas satisfaisantes, ce qui entrave son évolutivité.

Études sur la confidentialité : Le chiffrement de la base de données est le moyen le plus évident pour assurer la confidentialité des données. En effet, les données confidentielles sont chiffrées à l'aide d'un algorithme symétrique comme AES [22, 23] et la clé symétrique est cachée en lieu sûr. Cette méthodologie a souvent été utilisée dans la littérature [84, 85, 90, 98–100]. Cependant, la sécurité du chiffrement des données est sujette à la sécurité du stockage de la clé. Ainsi, un attaquant intérieur au système pourrait éventuellement avoir accès à cette clé, et alors pourra déchiffrer toutes les données confidentielles.

Études sur l'intégrité : Un autre aspect de sécurité des données est l'intégrité, soit la garantie que l'information est authentique.

Xu *et al.* [101] proposent une méthodologie de vérification de l'intégrité des données en

utilisant la cryptographie. Les auteurs proposent un hachage et chiffrement des données avec leur numéro de ligne dans une table séparée qui sert à la vérification. Cependant, leur méthode peut être améliorée au niveau du stockage des données, du temps de calcul, et de l'efficacité globale de la solution.

Études sur la disponibilité : La contre-mesure la plus triviale pour assurer la disponibilité des données est la sauvegarde fréquence des données [73, 84, 90]. Dans notre travail antérieur, nous proposons une méthodologie hybride de sauvegarde et restauration pour assurer la disponibilité des données en cas de perte. Cependant, cette méthodologie se retrouve à l'étape 5 (i.e. « récupération ») de la méthodologie de gestion des attaques internes. Cependant, un mécanisme différent doit également être proposé pour détecter l'attaque afin de pouvoir répondre et récupérer. Sinon, il est impossible de savoir quand la récupération est nécessaire.

Outils d'évaluation de la préparation contre les attaques internes

L'outil « Insider Threat Cybersecurity Framework » (ITCF) [89] a pour but d'évaluer la maturité des organismes quant à leur sécurité vis-à-vis des attaques internes. Il se base sur un questionnaire qui doit être rempli par les autorités compétentes au sein de l'entreprise et détermine leur maturité par rapport à 5 axes : l'identification, la protection, la détection, la réponse et le rétablissement. Les lacunes de l'entreprise sont relevées pour permettre à l'équipe responsable de les combler. Bien que cet outil soit utile pour estimer la sécurité d'une entreprise, la solution des auteurs ne fait qu'indiquer les problèmes à résoudre et non la méthode à suivre pour les résoudre. Ceci peut s'avérer dangereux, car l'entreprise peut implémenter des solutions inadéquates et inefficaces.

Le rapport émis par le « Center for Internet Security » (CIS) sur les différents modèles d'attaques [90] prend en compte également les attaques internes. Les quinze modèles d'attaques proposés peuvent être contre-attaqués par l'implémentation des techniques de défense de niveau 1 du Guide de contrôle CIS [102]. Ce guide comporte diverses méthodes appelées « Safeguard » qui cherchent à protéger le système contre différentes attaques. Ces méthodes se divisent en cinq catégories, selon leur fonction de sécurité : détecter une attaque, identifier une attaque, protéger contre une attaque, répondre à une attaque et récupérer après une attaque. Les safeguards exposés dans le Tableau 2.1 résument ceux qui sont pertinents à une attaque interne sur les données [102].

Tableau 2.1 Description des Safeguards relatifs à la sécurité des données

ID	Safeguard	Étape
2.1	Inventaire des logiciels	Identification
2.3	Gestion des logiciels non autorisés	Réponse
2.5	Liste de logiciels et scripts autorisés	Protection
3.1	Processus de gestion des données	Identification
3.3	Configuration d'ACL relative aux données	Protection
3.5	Suppression sécurisée des données	Protection
3.11	Chiffrement des données stockées	Protection
3.13	Prévention de perte des données	Protection
3.14	Journal des accès aux données critiques	Détection
4.1	Processus de configuration sécurisé	Protection
5.4	Restriction des privilèges d'administrateurs	Protection
6.1	Processus d'attribution d'accès	Protection
6.2	Processus de révocation d'accès	Protection
11.1	Processus de récupération de données	Récupération
11.2	Sauvegardes automatiques des données	Récupération
11.3	Protection des données de récupération	Protection
13.1	Centralisation des alertes liées à la sécurité	Détection
14.1	Programme de sensibilisation à la sécurité	Protection
14.4	Formations sur les bonnes pratiques de sécurité	Protection
14.6	Formations sur la reconnaissance d'incidents	Protection
17.1	Désignation d'un personnel de gestion d'incidents	Réponse
17.2	Établissement d'un contact pour reporter des incidents	Réponse

Ces informations constituent des méthodes et de bonnes pratiques à adopter, mais ne correspondent pas à une architecture spécifique. Cependant, une architecture plus structurée pourrait découler de ces méthodes.

Ainsi, le Tableau 2.2 compare les différentes méthodes existantes dans la littérature pour la

détection des attaques internes.

Tableau 2.2 Comparaison des méthodes de détection des attaques internes

Solution	**Avantages**	**Lacunes**
Dahmane et Foucher [88], Chaîne de Markov basée sur le comportement de l'employé	Méthode d'apprentissage automatique qui détecte les attaques internes ; Plus efficace que les autres apprentissages.	Étude psychologique qui peut être inexacte ou imprédictible ; Possibilité de biais malicieusement induit dans la classification ; Exactitude de la solution dépend de la bonne identification des attaques internes au préalable.
Reinerman-Jones *et al.* [95], Mesure du niveau de stress lors d'une attaque interne	Détection liée au comportement psychophysique de l'employé ; Fonctionne en arrière-plan et n'interfère pas avec le fonctionnement de l'application.	Possibilité d'évader les signes de stress avec entraînement.
Duncan *et al.* [96], Détection des attaques internes sur les services Cloud	Utilisation d'un arbre d'attaque pour modéliser les différentes possibilités d'attaques ; Proposition d'améliorations dans le contexte de services Cloud.	Différence de paradigme : les améliorations proposées ne sont pas adaptées à notre situation ; L'efficacité de leur arbre d'attaque dépend de la bonne identification des attaques internes au préalable.

(Suite à la page suivante)

Tableau 2.2 Comparaison des méthodes de détection des attaques internes (Suite)

Solution	**Avantages**	**Lacunes**
Egbunike et Ranjedran [97], Base de données négative contre le vol et l'altération de données	Possibilité de brouiller les données chiffrées grâce à des valeurs secrètes ; Meilleure confidentialité et l'intégrité des données.	Valeurs statiques vulnérables à des attaques internes ; Valeurs dynamiques nécessitent un travail supplémentaire de l'utilisateur.
Mousa *et al.* [84], Vinayagamurthy *et al.* [98], Chen *et al.* [99], Okada *et al.* [100], Dib [85], Chiffrement de la base de données avec AES	Empêche un attaquant d'accéder aux données pour les lire ou les modifier.	Un attaquant interne peut avoir accès à la clé de chiffrement.
Xu *et al.* [101], Vérification de l'intégrité par hachage	Meilleure intégrité des données grâce à la signature et le chiffrement des données.	Complexité spatiale due à la table de vérification ; Complexité temporelle due au nombre d'opérations supplémentaires.
Dib [85], Sauvegarde et récupération des données	Permet de récupérer les données perdues après une attaque.	Se focalise sur l'étape 5 (i.e., récupération) mais ne propose pas de méthodologie de détection de l'attaque.
Mylrea *et al.* [89], Insider Threat Cybersecurity Framework	Relève les lacunes d'une entreprise par rapport à ses politiques de gestion des attaques internes.	Ne propose pas de solution concrète pour répondre aux lacunes identifiées.
CIS [90, 102], Safeguards pour protéger un système informatique	Liste exhaustive des sujets à traiter pour l'identification, la détection, la protection, la réponse et la récupération dans le contexte d'attaques internes.	Seulement des méthodes génériques et des bonnes pratiques, mais ne donne aucune piste de mécanisme concret.

Cette comparaison justifie le besoin d'un modèle de sécurité qui protège les données locales contre les menaces intérieures.

2.5 Sécurité des données distribuées dans un large système informatique

La problématique des bases de données distribuées a attiré l'attention de la communauté scientifique. En effet, une distribution large implique une considération accrue de sécurité. Parmi les solutions proposées dans la littérature, certaines se basent sur l'utilisation de la blockchain pour permettre une distribution plus efficace et sécurisée.

En effet, les algorithmes cryptographiques standard, tels que le chiffrement symétrique AES [22, 23], peuvent être utilisés pour assurer la confidentialité des informations sur une base de données distribuées. Cependant, ils ne sont pas suffisants pour répondre aux problématiques d'intégrité, d'authenticité, de révocation d'accès, et de synchronisation dans ce genre de situations.

2.5.1 Distribution des données basée sur la blockchain

Fan *et al.* [76] explorent le concept de banque distribuée basée sur la blockchain. Leur solution diffère de la cryptomonnaie, car il ne s'agit pas de créer une nouvelle monnaie ni d'assurer l'anonymat des participants. Il s'agit d'une approche décentralisée pour un système bancaire standard auquel les utilisateurs doivent s'inscrire. Bien que leur solution utilise de façon efficace les fonctionnalités de la blockchain pour assurer l'intégrité des données contre un attaquant interne, elle ne traite pas la problématique de confidentialité. Cette limitation est critique, car les données des clients doivent non seulement être protégées contre l'altération, mais aussi contre la divulgation. De plus, les auteurs proposent deux algorithmes de consensus à utiliser pour l'implémentation de leur solution : le consensus majoritaire ou bien le consensus « Practical Byzantine Fault Tolerance » (PBFT).

Plusieurs tentatives de réinvention des bases de données distribuées en utilisant la blockchain ont été proposées dans la littérature [75, 103, 104]. Ces solutions ont pour but de sécuriser les données dans les DDB en enregistrant toutes les transactions de la base de données dans une blockchain. En somme, les nœuds de bases de données font partie intégrante de la blockchain, et le consensus permet d'ordonnancer les transactions qui ont lieu sur ces bases de données. Cette approche améliore la synchronisation et résout alors la problématique de concurrence.

De plus, Muzammal *et al.* [75] proposent une base de données basée sur la blockchain avec la possibilité de chiffrer les données. Cependant, seul le propriétaire des données ou l'utilisateur authentifié peuvent accéder aux données chiffrées. Ceci soulève la question de praticabilité

de la solution, en particulier lorsque les données doivent être utilisées par un serveur d'application. De plus, ceci souligne l'importance de la gestion des clés, qui n'est pas abordée par les auteurs. En effet, une problématique importante du chiffrement des données distribuées est la gestion des clés, des autorisations et surtout de la révocation d'accès [67].

Cependant, l'efficacité des travaux existants dépend fortement de l'algorithme de consensus choisi. Les algorithmes de consensus majoritaire, de PBFT [105] et de Raft [106] sont vulnérables aux attaques de 51% et 33% respectivement et sont réputés pour le lourd trafic réseau qu'ils génèrent [59]. De plus, aucune approche ne propose un mécanisme de chiffrement satisfaisant qui autorise les nœuds à partager les données en maintenant une possibilité de révocation d'accès. Finalement, la problématique d'authenticité des résultats n'est pas traitée par les solutions proposées.

2.5.2 Sécurité des couches réseau et de réplication d'état sur la blockchain

Une solution de sécurité basée sur la blockchain est régie par les limitations de celle-ci. En particulier, les limitations de la couche réseau et celles de la couche de réplication d'état. En effet, l'authentification des nœuds est primordiale pour empêcher la fuite d'informations confidentielles, ce qui relève de la couche réseau. De plus, certaines informations critiques doivent être partagées sur la blockchain de façon sécurisée, ce qui relève de la couche de réplication d'état.

Sécurité de la couche réseau de la blockchain

La couche réseau d'une blockchain privée ou publique est la cible de différentes attaques informatiques.

Dans le cas d'une blockchain privée, Homoliak *et al.* [61] proposent des contre-mesures basées sur la surveillance des nœuds, l'authentification à double facteur, ou bien d'intégrer le contrôle d'accès au mécanisme de consensus. Cette dernière solution apporte un grand ajout en sécurité dans la mesure où elle élimine la centralisation du contrôle d'accès. Cependant, les auteurs soulèvent le problème de dynamisme du réseau en matière d'entrée et sortie des nœuds du consensus.

Dans le cas d'une blockchain publique, les mêmes auteurs dans [61] proposent des contre-mesures telles que l'utilisation du protocole « Domain Name System Security Extensions » (DNSSEC) pour les attaques DNS, l'utilisation d'un « Virtual Private Network » (VPN) pour les attaques de routage, ou encore l'implémentation d'un contrôle d'accès pour limiter les attaques de type DoS. Cependant, ces solutions ont un effet négatif sur la réactivité du

réseau. Ceci pourrait ralentir la distribution d'un bloc miné et amener à des pertes de revenus pour les nœuds miniers.

Il est donc essentiel de prendre en considération un protocole d'authentification léger qui permet aux nœuds critiques de la blockchain de rejoindre le réseau de façon sécurisée, sans ralentir le fonctionnement de la blockchain.

Sécurité de la couche de réplication d'état

Les solutions de protection de la couche « Replicated State Machine » (RSM) cherchent à assurer la confidentialité et l'intégrité des données, ainsi que l'anonymat des utilisateurs.

Dans l'optique de protection de la confidentialité des données, une méthode proposée par le « Commonwealth Scientific and Industrial Research Organisation » (CSIRO) [67] repose sur l'échange d'une clé préalable à la conception de la blockchain. La clé échangée permet de chiffrer les informations sur la blockchain, afin que seuls les participants autorisés y aient accès. Cependant, une situation où la clé de chiffrement est divulguée implique une perte totale de la confidentialité. Empêcher une telle fuite revient alors à supposer une grande confiance envers les participants ainsi que leur capacité à conserver le secret de la clé. Un second inconvénient est la difficulté de la révocation d'accès. En effet, une fois qu'un participant rejoint le réseau, il est impossible de le forcer à quitter le réseau. S'il dispose toujours de la clé, il peut continuer à accéder aux données chiffrées. De plus, la notion de partage sécurisé et efficace de la clé de chiffrement est relevée par les auteurs.

Ajayi et Saadawi [74] proposent une méthode de détection d'attaques internes à une blockchain basée sur les contrats intelligents. Un nœud est d'abord authentifié au moyen de son couple de clés privée et publique. Ensuite, la transaction est vérifiée selon quatre conditions : la transaction respecte un format spécifique, le coût de la transaction et le nombre de transactions minées par seconde ne dépassent pas une certaine limite et le propriétaire de la transaction ne doit pas la miner. La dernière condition dispose d'une faille évidente : la collaboration entre plusieurs nœuds. En effet, une même personne qui dispose de deux nœuds sur la blockchain peut soumettre une transaction à partir d'un nœud et la miner à partir de l'autre nœud. Cette méthode d'évasion est facilement réalisable et ne sera pas relevée par le contrat intelligent. D'autre part, l'authentification des nœuds est également à risque si la clé privée du nœud est compromise. En effet, les nœuds, étant des machines usuelles, sont vulnérables aux SCA qui permettent aux attaquants de se procurer les clés privées à partir des caractéristiques électroniques des nœuds [49, 50, 53].

2.6 Synthèse

Avec la littérature actuelle, nous réalisons les lacunes qui existent au niveau de la sécurité des données. En effet, les données locales souffrent de vulnérabilités contre les attaques internes, et des méthodologies claires pour identifier et détecter ces attaques sont nécessaires. De plus, la distribution des données apporte de nouveaux défis en termes de synchronisation et concurrence qui peuvent être résolus par la blockchain. Cependant, la sécurité de la couche réseau et de la couche de réplication d'état doit être améliorée avant de pouvoir distribuer les données de façon efficace et sécurisée.

CHAPITRE 3 MÉTHODOLOGIE

Cette thèse vise à créer une architecture de sécurité des données stockées sur les serveurs pour les systèmes locaux et distribués. Pour atteindre cet objectif, nous avons énoncé cinq objectifs spécifiques à la Section 1.3. Nous avons également exposé dans le Chapitre 2 les lacunes des solutions existantes concernant la sécurité des données locales et distribuées. Ainsi, nous exposons dans ce troisième chapitre la méthodologie de travail suivie dans cette thèse en mettant en évidence le lien entre les objectifs et les volets de la recherche. Ce travail est donc divisé en trois volets principaux, soit la conception d'un modèle d'attaques internes, la conception d'une architecture de sécurité locale contre les attaques internes, et la conception d'un système de gestion de bases de données distribuées basé sur la blockchain. Chacun de ces volets comprend les étapes de conception, d'implémentation et d'évaluation de performances, et fait l'objet d'un article de revue scientifique.

3.1 Volet 1 : Conception d'un modèle d'attaques internes

L'architecture initiale présentée dans le travail de recherche antérieur vise à protéger les données stockées sur un serveur contre les attaques externes [85]. Cependant, les statistiques montrent que 36% des attaques proviennent de l'intérieur de l'organisme concerné [107, 108]. Ainsi, afin de protéger les données contre les attaques internes, une architecture de sécurité plus adaptée doit être proposée. Cependant, la littérature a démontré que pour garantir la meilleure défense, il est impératif de créer un modèle d'attaques spécifique à notre architecture. Ainsi, le premier article « Insider Attack Against HSM-Based Architecture », présenté au Chapitre 4, répond à cette portion de l'objectif de recherche.

3.1.1 Vecteurs d'attaque et modèle d'attaques

Pour créer le modèle d'attaques adéquat, nous avons suivi une méthodologie d'analyse exhaustive sur tous les vecteurs d'attaques. Ainsi, nous commençons par dresser une liste exhaustive de tous les vecteurs d'attaques possibles dans l'architecture (i.e., chaque canal de communication, chaque nœud dans le réseau). Ensuite, chaque vecteur d'attaques est analysé pour dresser une liste exhaustive de toutes les attaques possibles sur ce vecteur (i.e., usurpation d'identité, interception de données, altération de données). Finalement, ces attaques sont analysées selon leur faisabilité et les attaques pertinentes sont retenues dans le modèle d'attaques.

3.1.2 Évaluation du modèle d'attaques

Pour évaluer le modèle d'attaques, nous implémentons chaque attaque afin d'en déterminer la faisabilité, la difficulté, et l'impact. La faisabilité est un résultat binaire qui détermine si l'attaque est possible ou non avec les circonstances et les technologies actuelles. La difficulté est une métrique numérique décimale qui est calculée en faisant la somme pondérée des obstacles auxquels l'attaquant fait face pour une certaine attaque. La pondération de l'obstacle est directement liée à sa difficulté. Par exemple, l'accès au HSM est plus difficile que l'accès à la base de donnée à cause des restrictions d'accès strictes sur le HSM. Ainsi, l'obstacle « Accès au HSM » aura une pondération plus élevée que l'obstacle « Accès à la base de données ». L'impact est une métrique numérique entière qui somme le nombre de piliers de sécurité affectés par une certaine attaque (entre 0 et 3).

Les attaques sont ensuite notées selon notre système de pointage « δ-score ». Le δ-score est le rapport de l'impact et de la difficulté d'une attaque. Ainsi, une attaque peut avoir une « δ-score » compris entre 0 et 3 inclusivement. Une attaque avec un δ-score supérieur à 1 indique une attaque facile à réaliser avec un grand impact. À l'opposé, une attaque avec un δ-score inférieur à 0.5 indique une attaque trop difficile à réaliser ou une attaque avec un impact négligeable.

3.2 Volet 2 : Conception d'une architecture de sécurité locale contre les attaques internes

Le modèle d'attaque internes qui résulte du premier volet indique que les données au sein des architectures basées sur HSM sont vulnérables à de nombreuses attaques internes. Ces attaques permettent d'usurper l'identité du serveur, d'accéder illégalement aux données, ou encore de les altérer ou supprimer [109].

La revue de littérature nous a bien montré que les méthodes existantes sont insuffisantes pour répondre à ce problème. En effet, les approches basées sur les anomalies ou l'apprentissage machine peuvent être déjouées par des administrateurs haut-placés. Ainsi, sans pour autant se défaire de ces méthodes, nous proposons une approche basée sur la cryptographie appliquée pour rajouter une couche de sécurité importante.

Ainsi, la deuxième étape de ce travail est de proposer une architecture de sécurité pour détecter les attaques internes, en se limitant dans un premier temps aux données localisées, en opposition aux données distribuées. Cette étape est atteinte dans le second article « HSM-Based Architecture To Detect Insider Attacks On Server-Side Data » présenté dans le Chapitre 5.

3.2.1 Architecture de sécurité et implémentation

Pour se démarquer des architectures proposées dans la littérature, nous adoptons une approche basée sur la cryptographie. Il ne s'agit pas de créer de nouveaux algorithmes cryptographiques, mais plutôt de combiner et utiliser les algorithmes existants pour en tirer de nouvelles propriétés. Par exemple, nous utilisons la cryptographie symétrique pour garantir l'intégrité des données.

Ainsi, nous proposons plusieurs mécanismes de détection qui, combinés, résultent en une architecture modulable selon les besoins du système. Les mécanismes sont orientés vers l'authentification des processus en utilisant les nonces (NBPA), ainsi que le maintien de l'intégrité des champs (HBFI) et la disponibilité des champs (HBFA) et des entrées (HBRA) dans la base de données en utilisant les hachages cryptographiques. Chaque mécanisme de détection a été conçu pour déjouer une ou plusieurs attaques du modèle d'attaques internes qui a été présenté dans le Chapitre 4.

3.2.2 Évaluation des performances

Pour évaluer les performances de notre architecture, nous procédons en deux temps. Tout d'abord, une évaluation qualitative permet de déterminer la valeur ajoutée par cette architecture au niveau de la sécurité des données. Ensuite, l'évaluation quantitative permet de mesurer les délais introduits par cet ajout de sécurité afin de déterminer leur impact sur le bon fonctionnement du système.

Évaluation qualitative

L'évaluation qualitative répond à la question : « Est-ce que l'architecture proposée permet de répondre aux attaques données par le modèle d'attaques internes ? ». Pour déterminer cela, les attaques sont menées une à une contre les mécanismes de détection concernés, et les résultats de l'attaque sont enregistrés de façon binaire (*détectée* ou *non détectée*).

Évaluation quantitative

L'évaluation quantitative étudie l'impact des mécanismes de détection, ainsi que leurs paramètres, sur les trois métriques suivantes : 1. le taux de service, 2. le temps de détection, et 3. l'utilisation de la mémoire.

Pour évaluer le taux de service, tous les mécanismes de détection sont désactivés dans un premier temps, et un nombre variable de requêtes n est envoyé au serveur, $n \in [100\ ; 3000]$.

Pour chaque expérience, nous mesurons le temps de traitement t pour ces n requêtes. En utilisant l'estimateur « Minimum Mean Square Error » (MMSE) [110], nous obtenons le profil linéaire du temps de traitement de la forme $y = a_{STD} \times x + b_{STD}$. Nous calculons ensuite le taux de service $\mu_{STD} = (a_{STD})^{-1}$. Dans un deuxième temps, nous répétons cette même expérience pour chaque mécanisme de détection, que nous activons un seul à la fois, puis tous ensemble. Cette expérience est réalisée pour les opérations de lecture et d'écriture. Nous obtenons le taux de service μ_X où $X \in \{NBPA, HBFI, HBFA\}$. Finalement, le délai relatif est calculé au moyen de l'équation 3.1.

$$\delta\mu = \frac{\mu_X - \mu_{STD}}{\mu_{STD}} \tag{3.1}$$

Pour évaluer l'utilisation de la mémoire, tous les mécanismes de détection sont désactivés dans un premier temps, et un nombre élevé de requêtes n est envoyé au serveur. Nous mesurons alors l'utilisation totale de la mémoire pour les n requêtes et nous calculons la moyenne. Dans un deuxième temps, nous répétons cette même expérience pour chaque mécanisme de détection, que nous activons un seul à la fois, puis tous ensemble. Nous analysons alors l'impact des mécanismes de détection sur l'utilisation de la mémoire pour les opérations de lecture et d'écriture.

Pour le mécanisme de détection HBRA, la métrique de l'évaluation est le temps minimal de détection d'une attaque. En effet, ce mécanisme de détection opère en arrière-plan et donc ne perturbe pas les performances du service. Dans un premier temps, une étude probabiliste permet de déterminer le nombre de vérifications nécessaires pour détecter l'attaque avec une certitude de α (i.e., 90%) en fonction de 3 paramètres : le nombre d'entrées dans la base de données m, le nombre de hachage testés à chaque vérification k, et la taille de la fenêtre de hachage L. Ensuite, nous mesurons le temps nécessaire pour conduire une vérification en fonction de $k \in [1\ ; 20]$ et $L \in [5\ ; 25]$, puis nous calculons le temps minimal de détection garantie en fonction k et L.

3.3 Volet 3 : Conception d'un système de gestion de base de données distribué sécurisé basé sur la blockchain

En considérant la quantité croissante de dispositifs connectés au réseau, et par conséquent la quantité croissante de données générées, de nombreuses organisations optent pour une solution de distribution de données.

La revue de littérature nous a indiqué que les architectures existantes aujourd'hui ne sont pas encore optimisées pour une répartition des données qui maintient un niveau de sécurité

suffisamment élevé. De plus, les mécanismes de détection d'attaques qui ont été proposés au Chapitre 5 sont plus adaptés pour des architectures où les données sont localisées dans un serveur unique.

Ainsi, la troisième étape de ce travail est de proposer un système de gestion de base de données distribué qui respecte les requis de sécurité, soit la confidentialité, l'intégrité et la disponibilité des données. Cette étape est atteinte dans le troisième article « BlockDBMS - a Secure Distributed Database Management System Based on Blockchain and HSM » présenté dans le Chapitre 6.

3.3.1 Distribution des données avec la blockchain

La technologie blockchain, par ses propriétés de décentralisation et d'immutabilité, permet déjà d'introduire la notion de distribution de données et d'intégrité des informations échangées sur la blockchain. Cependant, la notion de confidentialité reste à traiter étant donné que la blockchain n'est pas conçue pour la confidentialité.

Tout d'abord, nous proposons dans BlockDBMS l'introduction d'un nouveau type de nœud, le HSM-Based Security Node (HSM-BSN) qui combine un nœud de blockchain, un HSM, et une instance de base de données. Les HSM-BSNs, au sein d'une blockchain privée où le consensus est déterminé par l'autorité (i.e., Proof of Authority), ajoutent des blocs qui contiennent des requêtes SQL et leurs résultats dans la base de données distribuée. Ces requêtes sont traitées par les HSM-BSN au moyen de contrats intelligents pour assurer la distribution et l'ordonnancement des requêtes auprès des nœuds acteurs de la blockchain. De plus, nous proposons un protocole d'identification et d'authentification pour les nouveaux HSM-BSN rejoignant le réseau.

Au niveau de l'implémentation, le choix d'une blockchain privée Ethereum est motivé par son accessibilité, sa facilité d'implémentation et la possibilité d'introduire des contrats intelligents avec le langage Solidity. Également, l'option du mécanisme de consensus Proof of Authority s'intègre parfaitement dans la logique de notre architecture où certains nœuds (les HSM-BSNs) disposent d'une autorité sur les autres. Ces propriétés font de la blockchain Ethereum un candidat idéal pour l'implémentation de BlockDBMS.

3.3.2 Évaluation des performances

Pour déterminer les performances de notre système de distribution des données, nous soumettons notre proposition à une évaluation qualitative et une évaluation quantitative.

Évaluation qualitative

L'évaluation qualitative permet de déterminer la valeur ajoutée par ce système en termes de sécurité, et de déterminer l'efficacité du protocole d'authentification des HSM-BSN.

Dans un premier temps, la valeur ajoutée se détermine en analysant les propriétés de sécurité que notre système respecte. Entre autres, nous nous attardons sur la résilience, la véracité et le chiffrement des données, la révocation d'accès et l'authentification des nœuds du réseau. Le protocole d'authentification est également évalué en le confrontant, à chaque étape, aux attaques communes contre les protocoles de sécurité.

Dans un second temps, nous utilisons la logique « Burrows-Abadi-Needham » (BAN) [111] pour évaluer de façon formelle la sécurité du protocole d'authentification des HSM-BSN. Cette logique permet de détecter les failles et redondances dans un protocole, en mettant en évidence son fonctionnement, ses étapes, les hypothèses requises pour la réussite du protocole, et les potentielles étapes inutiles [111].

Évaluation quantitative

L'évaluation quantitative permet de déterminer la tendance du temps de traitement des requêtes dans un tel système ainsi que le taux d'occupation du réseau.

L'introduction de la blockchain risque de changer la tendance dans le temps de traitement par rapport à un système localisé. Ainsi, pour déterminer l'impact de notre solution, nous envoyons un nombre variable de requêtes simultanées n au serveur, $n \in [1\ ;40]$. Pour chaque expérience, nous mesurons le temps de traitement t pour ces n requêtes. De plus, pour des valeurs faibles de n, nous analysons en détail la frise chronologique de la blockchain, en notant des temps clé tels que la soumission d'un bloc, la transmission d'une transaction, ou la réception d'un résultat. Nous utilisons ensuite cette frise pour émettre des hypothèses crédibles sur la tendance des temps de traitement.

Le taux d'occupation du réseau pourrait également être impacté par le protocole d'authentification des HSM-BSN. Pour cela, il est important de se pencher sur la quantité de messages transmis dans le réseau pour chaque tour de ce protocole.

3.4 Synthèse

En somme, les travaux réalisés dans cette thèse, présentés dans les trois prochains chapitres, nous permettent d'atteindre les objectifs de recherches de la Section 1.3. Le premier objectif spécifique est atteint à travers le premier volet, le second objectif est atteint à travers le

second volet, et le troisième objectif est atteint à travers le troisième volet. Les quatrième et cinquième objectifs spécifiques, soit l'implémentation et l'évaluation, sont intégrés à chacun des trois volets de la thèse. Ainsi, la combinaison de ces travaux permet d'atteindre l'objectif principal de sécuriser les données dans les architectures locales et distribuées.

CHAPITRE 4 ARTICLE 1 : INSIDER ATTACK MODEL AGAINST HSM-BASED ARCHITECTURE

Marc Dib, Samuel Pierre (Senior Member, IEEE)

Mobile Computing and Networking Research Laboratory (LARIM), Department of Computer and Software Engineering, Polytechnique Montréal, Montreal, QC H3T 1J4, Canada

Corresponding author : Marc Dib (marc.dib@polymtl.ca)

This work was supported by the Natural Sciences and Engineering Research Council of Canada (NSERC), Flex Group, and PROMPT.

Revue : Accepté et publié le 14 Août 2023 dans le journal IEEE Access, Volume 11, pages 86848-86858

ABSTRACT

Data security is an increasingly important issue in 2023. Whether about user privacy, data availability, or integrity, consumer information is getting targeted by cyber-pirates for various motives. Many strategies and tools have been developed to keep outsider attackers from accessing server-side data, but there needs to be more solutions that target insider attacks. In this paper, we propose a combinatory attack model to identify the risks of insider attacks against HSM-based security architectures. Our proposed model is based on the study of attack vectors in the security architecture and the conduction of all possible attacks on those vectors. It shows that these typical architectures are vulnerable to private key theft and replacement and data theft, alteration, swapping, nullification, and deletion. Results show that we successfully conducted each attack on an HSM-based security architecture relatively easily. They prove the essential need for a security architecture considering insider threats.

INDEX TERMS

Application server, cryptography, cybersecurity, confidentiality, hardware security module, insider attacks, integrity, privacy.

4.1 Introduction

Application data security is a hot topic amidst the rising concern about privacy and confidentiality. The last five years have seen many data breaches in various fields. Medical organizations targeted include Infinity Rehab, Baptist Medical Center, and Shields Health Care [112]. Financial organizations include Nelnet Servicing and Desjardins Bank [112, 113]. These attacks can have disastrous financial repercussions on the organization, whether direct or indirect. A direct repercussion would be a denial of service through an attack on server-side data, paralyzing the organization's functioning and inducing financial losses. An indirect repercussion can be in the Desjardins case, followed by a class-action lawsuit against them, approved for CA$200 million [113].

Most cases are not random attacks against small businesses but targeted attacks against robust and supposedly secure networks. Indeed, many of these websites dedicate a whole page to their "Security Measures" [114]. The gravity of the witnessed attacks makes us wonder where these organizations are missing the mark. In other words, organizations need to better identify the threats against their server-side data.

These attacks can originate from either outside or inside threats. Statistically, we estimate 64% of the attacks against a cyber system originate from outside the organization, which still leaves 36% of insider attacks [107, 108].

Many organizations abide by the Inside-DMZ-Outside architecture, where the "Inside" zone is considered safe [115]. However, the non-negligible proportion of insider threats refutes the assumption of the safety of the "Inside" zone. However, the amount of research and industry practices targeting insider attacks is disappointingly low.

In addition, our literature review shows that without a clear insider attack model, the proposed defense mechanisms can be irrelevant or incomplete. Hence, to establish a secure inside zone in a cyber system, we must first create an attack model to help us understand the possible threats. This attack model must be exhaustive to ensure to document all threats. An incomplete attack model results in unknown vulnerabilities that can be exploited against our system. The best approach to reach an exhaustive attack model is a combinatory method, where we test every known attack on every attack vector in the architecture.

To choose the architecture upon which the attack model is based, an increasingly popular method to protect server-side data is the usage of a Hardware Security Module (HSM) [86]. HSMs are known for their powerful cryptographic processors and their tamper-proof partitions that make them robust and resilient against cyber-attacks [86, 116, 117]. Considering the advantages of the HSM, many organizations would be tempted to believe that adding this

component to their cyber system is enough to ensure the full-scale security of their server-side data. In this paper, we study this claim carefully by centering the attack model on an HSM-based architecture.

Moreover, we propose a combinatory attack model that defines and identifies insider attacks against modern architectures protected by an HSM. The main contributions of this paper are as follows:

1. Establish attack vectors on a selected baseline security architecture;
2. Design the attack model according to the available attack vectors;
3. Evaluate the feasibility and impact of the proposed attacks.

The rest of this paper is organized as follows. Section 4.2 details the technical background knowledge. Section 4.3 presents a critical review of the existing insider attack mitigation solutions. Section 4.4 reviews the baseline security architecture. The attack vectors on the selected security architecture are detailed in Section 4.5. Section 4.6 describes the attack model on the selected architecture. A discussion of the obtained results following the implementation is presented in Section 4.7. Finally, Section 4.8 concludes the paper.

4.2 Background

In this section, we define the necessary background on insider attacks, attack vectors, and Hardware Security Modules (HSM).

4.2.1 Insider Attacks

We categorize attacks on a cyber system can into two groups: insider attacks and outsider attacks. Since this paper focuses only on insider attacks, we do not define outsider attacks.

An attack is considered insider when the attacker belongs to the targeted organization [108, 118]. These attacks can be further divided into two groups: *Accidental harm* and *Intentional harm* [119].

Accidental Harm

Harm is considered accidental when it is caused by employee negligence. There is no intention to harm the organization, but the lack of thoroughness in the work causes damage [108]. Poorly programmed applications are an example of employee negligence. They result in erroneous results, data leaks, or recurrent crashes. Misconfigured networks and nodes are other examples of negligence that result in open vulnerabilities for outsider attackers.

Intentional Harm

Harm is considered intentional when there is an intention of causing damage to the organization [108]. An example of intentional harm is accessing and extracting sensitive data with the intent of sharing them publicly, jeopardizing data confidentiality. Another example is denying the service through data tampering, such as modifying or suppressing user data [72]. This attack jeopardizes the integrity and availability of user data.

Overall, an insider attack is easier to implement since the attacker already has access to the cyber system and is harder to detect [118].

4.2.2 Attack Vector

We define an attack vector (AV) as a component that can be targeted during an attack. An attack vector can be:

— A network component (i.e., server, router, machine);
— A communication channel between two network components;
— An accessory device used on a machine (i.e., USB Key, Smart Card);
— An application or a process running on a network component (i.e., backend server, database).

4.2.3 Hardware Security Module (HSM)

Sustek [116] defines an HSM as a trusted computer designed to perform cryptographic operations like key management, electronic signature, encryption, and decryption. HSMs are tamper-proof, resistant to side-channel attacks, strictly limited in terms of access, and thoroughly tested and certified by competent authorities [86, 116, 117].

Moreover, HSMs can generate random numbers with a higher entropy than regular processors due to their advanced Random Number Generator (RNG), which considers multiple physical and hardware factors compared to a software-based RNG [117]. Encryption keys created by an HSM are less predictable than those produced by a regular machine, making them more secure.

4.3 Related Work

Most of the related works focus on outsider attacks. Therefore, very few define the abilities of an insider threat to perform an attack.

In this section, we present *behavior-based attack detection*, *technical-based attack detection*, and *insider attack mitigation evaluation tools*. Even though these resources propose a solution against insider attacks, we only focus on their ability to define them properly.

4.3.1 Behavior-Based Attack Detection

The most predominant research regarding insider attacks is related to employee behavior. Indeed, a cyber-attack follows criminal psychology; identifying its red flags can help detect and prevent it [93]. Highlighting suspicious behavior patterns is a key to insider attack detection [94].

Dahmane and Foucher proposed an employee profiling method based on Markov chains [88]. Each employee is assigned a Markov chain that describes their usual activities. The method studies the following features: working hours, visited websites, login and logout time, and connection and disconnection of a device. When an attack is suspected, a Markov chain is generated for the pattern of this attack, and the employees with the highest similarity with this chain are marked as suspicious. This method uses intuitive learning, which is more efficient than other proposed methods in complexity and computation time. However, as with all automated learning processes, the definition of the metrics is a significant factor in the success of the proposed tool. The features chosen by the authors are related to mundane activities like the visited websites. In the case of an application server mounted with a database, a Rogue Employee (RE) with the correct privileges can extract sensitive data without affecting any of the defined features. This attack is a prime example of the importance of defining the insider attack model, which is the key to preventing it.

Reinerman-Jones *et al.* explored situation awareness as another behavior-based detection method for insider attacks [95]. In this work, one honest group of participants and one malicious group performed various tasks on a Swiss banking simulation software. The authors measured the participants' behavior according to their immersion in the situation, graphic and system usability, and stress level. The results show that any difference in behavioral response is due to the performance of an illegal activity. However, we believe that such methods can be easily evaded by practicing not showing any sign of stress during the attack. Despite the advantage of this approach, it would be dangerous to assume that the attackers will produce the expected psychological response to detect the attack. Moreover, this method proves that a thorough attack model, based on the attacker's actions, is necessary to defend against insider attacks.

4.3.2 Technical-Based Attack Detection

Considering the drawbacks of behavioral analysis, we need a technical attack model that works in parallel to ensure a maximized detection rate. However, the authors in [94] note that, back in 2013, technical systems were often inconclusive and raised few warnings for insider attacks. More recent literature attempted to design technical-based attack detection systems.

Duncan *et al.* [96] proposed a combination of Attack-Tree and Kill-Chain approaches to detect insider attacks on cloud services. Since our focus is on the attack model and not the defense mechanisms, we are more interested in the attack-tree aspect of their work. The authors used the tree structure to define possible actions that lead to an insider attack. A tree is built from root to leaf nodes, where the root node is the result of the attack, the intermediate nodes are the intermediate actions, and the leaf nodes are the first actions. The attack is then read from leaf nodes to the root node, where a path is chosen by the attack that leads, step by step, to the result. However, since the tree is built from the root node, it requires a definition of all possible attacks. An improper root definition will result in an inaccurate or incomplete attack tree.

Tukur and Ali published a demonstration of the effect of insider attacks on Internet of Things (IoT) Systems [92]. Their approach relied on establishing an insider attack model before developing any defense mechanism. However, their approach is centered on IoT systems and is an entirely different paradigm than our proposed methodology. Nevertheless, the authors highlighted the importance of having a proper attack model.

4.3.3 Insider Attacks Mitigation Evaluation Tools

Evaluation tools have been proposed in literature and industry to help organizations assess their maturity regarding insider attack mitigation. The "Insider Threat Cybersecurity Framework" (ITCF) by Mylrea *et al.* [89] evaluates such maturity. This framework is based on a questionnaire that analyses the organization's ability to identify, detect, protect, respond, and recover from an insider attack. For each category, notable flaws are highlighted. We noted that the first step, according to the ITCF, is to identify an insider attack, which means the organization should be able to define what actions constitute an attack. However, with this tool being a closed source, we have no means to verify their level of detail in identifying an attack.

In addition, the Center for Internet Security (CIS) regularly publishes tools to help the organization better prepare against cyber-attacks. Their attack models presented in [90]

include *Insider and Privilege Misuse* scenarios, such as replication of sensitive data and theft of confidential information. However, this attack model is very generic, and the application of the scenarios varies with the architecture. Depending on the initial defense mechanisms in place, some attack scenarios might feel obsolete. In addition, without a concrete architecture to analyze, a scenario can regroup multiple attacks that must be identified individually to ensure maximum protection. This highlights the need for a specific attack model that is applied and tailored to the concerned architecture we want to defend.

4.4 Baseline Security Architecture

As we specified in Section 4.3, generic and vague attack models are not as effective as targeted and precise attack models. Therefore, we must first determine an architecture upon which we build our attack model.

In our previous work [85], we designed a security architecture based on HSM to protect server-side data against outsider attacks.

This architecture holds three modules that protect the integrity of the private keys of the application server, the confidentiality of the database contents, and the availability of the database contents through encrypted backups. Overall, we proved that this architecture performs well against outsider attacks by using the HSM cryptographic abilities and tamper-proof partition.

The baseline for this architecture is depicted in Fig. 4.1. A Trusted Administrator Group (TAG) holds a consensus of smart cards that allows them to control the HSM.

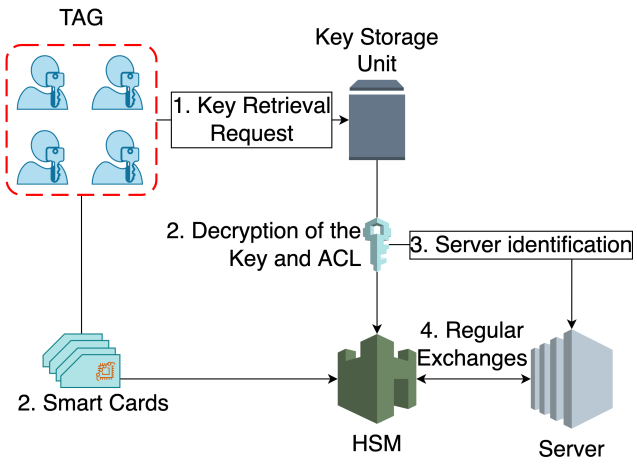

Figure 4.1 HSM-based Security Architecture Baseline

The architecture respects the following steps:

1. **Key Retrieval Request:** the TAG order the retrieval of the key from the key storage unit to load it inside the HSM and setup the TLS protocol for the application;
2. **Smart Cards & Decryption of the Key and ACL:** the private key and the ACL are decrypted by the HSM and loaded in the tamper-proof partition;
3. **Server Identification:** the HSM uses the ACL to identify the application server;
4. **Regular exchanges:** at this point, all processes related to the identified server can communicate with the HSM.

We believe that this architecture is a suitable candidate for the attack model. Indeed, this architecture aims to protect server-side data against cyber-attacks. However, the insider threat component was not evaluated nor considered during the architecture design. Hence, we must review this architecture by analyzing its attack vectors and designing an insider attack model.

4.5 Attack Vectors on the Architecture

The first step in the methodology of defining the attack model is to properly identify all the attack vectors (AV) on the security architecture.

Based on the information presented in section 4.4, we identify six attack vectors: Key Storage Unit (AV-A), Key Transfer (AV-B), Smart Cards (AV-C), HSM (AV-D), Application Server (AV-E) and HSM-Server communication (AV-F). We study confidentiality, integrity, and availability for each attack vector to determine the possible vulnerabilities a Rogue Employee (RE) can exploit.

4.5.1 Key Storage Unit (AV-A)

The Key Storage Unit (KSU) only contains the private keys used for the TLS purposes of the application server. According to the security architecture in [85], we use the LMK of the HSM to encrypt all the private keys by the strongest available symmetric encryption algorithm. Since the LMK is securely stored in the tamper-proof partition of the HSM [87], the RE cannot decrypt the private keys unless they gain access to the LMK. The integrity of the TLS protocol and the confidentiality of the application data are at little risk with the KSU.

However, the availability of the keys is at risk since any RE who has read-write access to the KSU can delete them. Nevertheless, the private keys stored in the KSU are only useful

when not loaded on the HSM, meaning the associated applications are not running. Losing the private key of a non-running application can be fixed by generating a new key pair and associating it to the application before runtime. This loss will only induce a short delay in starting the application compared to loading a ready-to-use key pair. Hence, the impact of this attack is almost null.

Overall, the only significant threat to AV-A depends on the RE's ability to access the LMK.

4.5.2 Key Transfer (AV-B)

Since AV-B is a communication channel, the two main possible attacks are eavesdropping and Man-in-the-Middle (MitM). The eavesdropping attack is ineffective since the private keys are encrypted with the LMK. As for MitM attacks, the communication between the KSU and the HSM is point-to-point with no intermediate network component per the architecture specification in [85].

If the communication channel is jeopardized and the HSM does not receive the key, we know the application is not yet running. We can attempt a new transfer until the key safely reaches the HSM. If the communication channel is jeopardized, we can expect the RE to provide a malicious key to replace the original key. This attack depends on the RE's ability to encrypt his malicious key using the LMK of the HSM.

4.5.3 Smart Cards (AV-C)

The Smart Cards used to activate and control the HSM are distributed to a few trusted employees. A consensus of these Smart Cards is required to make administrative changes to the HSM configuration. However, if malicious employees reach this consensus, we can assume the LMK is compromised, and all data encrypted by the LMK can become available to the RE. The AV-C represents one of the biggest threats to application data security.

4.5.4 Hardware Security Module (AV-D)

Due to the protective and strict nature of the HSM, we assume there are no real significant threats in AV-D. Indeed, the HSM is a highly tested and secure physical element. It is shielded from the regular network and can only be accessed by a few select network entities. Its access regulation policies are very strict to prevent attacks on its software.

4.5.5 Application Server (AV-E)

The application server includes both the backend server and the database server. Each has its possible vulnerabilities.

Backend Server

A Rogue Employee can attempt to access, modify, or delete sensitive data by altering the behavior of the source code of the backend server. Indeed, the backend application requires the data to be decrypted to process it. By changing the source code of the application, it is possible to expose sensitive data by printing it to the console, saving it to a file, or sending it to another machine on the network. An employee with authorized access to the source code can, theoretically, push those changes to production.

Database Server

The database server is protected from outsider attacks using the ESDB mechanism of the security architecture, as described in [85]. The ESDB mechanism encrypts the data using the LMK of the HSM. For an insider threat, extracting data from the database and passing it through the HSM for decryption is possible. With the proper access to the file system of the application's server, it is also possible to delete fields, rows, and tables from a database.

4.5.6 HSM-Server Communication (AV-F)

The communication between the HSM and the application server is considered secure and privileged according to architecture in [85]. It is secure because the connection is point-to-point, with no intermediate network component between the two nodes. It is privileged because the application server is listed as an authorized network component in the HSM ACL, allowing them to communicate together.

These properties prevent an outsider attacker from injecting or eavesdropping data between the HSM and the application server. However, an insider attacker can tamper with this communication. A rogue employee with the proper rights can open a new session on the server. From this session, the RE can send any encryption or decryption request to exploit the communication channel between the server and the HSM. This vulnerability is a complement to the AV-E threats related to the database files by providing the decryption of the sensitive data stored in the database.

A comprehensive summary of the available attack vectors and their vulnerabilities is exposed in Table 4.1.

Tableau 4.1 Summary of the Attack Vectors and their exposed vulnerabilities

	Attack Vector	Vulnerability
AV-A	Key Storage Unit	Theft & replacement of the private key[a]
AV-B	Key Transfer	Theft & replacement of the private key[a]
AV-C	Smart Cards	Theft of the LMK[b]
AV-D	Hardware Security Module	N/A
AV-E	Application Server	Vulnerability implantation in the source code[b] Data theft from the database[a]
AV-F	HSM-Server Communication	Unauthorized use of the LMK

[a]Requires access to the LMK
[b]Requires rogue admin consensus

4.6 Attack Model

The second step in our methodology is to design an attack model that targets the vulnerabilities shown by the attack vectors.

In this section, we start by defining the hypotheses of our attack model and studying their impact on the attack vectors. Then, we establish the attack scenarios that constitute our model. Finally, we describe an implementation of our attack model.

4.6.1 Attack Model Target and Goals

A good definition of any attack model starts by defining the targets and goals of the attacker. It is essential to know the goal of an attacker to simulate their possible moves.

Attack Motives

In the context of a cyber system, the attacker has three possible motives:

— to obtain sensitive data and use them for personal gain, share them publicly, or threaten to share them publicly;
— to modify data for personal gain or to cause harm to the target of the attack;
— to suppress data to cause harm to the target of the attack.

A few examples exist for each of those goals. Obtaining sensitive data about the users' credit cards can help the attacker to commit credit card fraud. Modifying data in the database of

a game server that uses in-game money can help the attacker to fake their balance and/or score compared to other players. Illicitly suppressing medical information about a patient can harm them if the suppressed information is the patient's allergic reaction to certain medications. As we can see, the impact of those attacks can affect the targets in very serious ways related to finance, health, and safety.

Undetectability

An important aspect of those attacks is the wish to remain undetected. Once the organization detects that its system has been the target of an attack, it can attempt to rectify the harm that was done. It is in the attacker's best interest that the attack remains undetected for as long as possible. Ideally, the attack should remain undetectable until the harm becomes irreversible.

4.6.2 Attack Model Hypotheses

The proposed attack model is based on one main hypothesis and several ramifications of this hypothesis.

Main Hypothesis

The main hypothesis dictates: "At most, one TAG member is, in fact, a rogue employee." We esteem this hypothesis to be both realistic and necessary.

The hypothesis is realistic because the TAG is a highly exclusive group of security administrators. Employees must prove their loyalty and implication in the organization's success and hold seniority before becoming a TAG member. Having several rogue TAG employees means that the company cannot effectively assess the loyalty of their highest-ranking administrators, which falls into a Human Resources (HR) issue. Since this problem is outside our scope of work, we suppose the HR department is competent enough to avoid such mistakes. Therefore, the hypothesis is realistic.

Moreover, the hypothesis is necessary to avoid unsolvable vulnerabilities. Considering the HSM configuration in [85], the presentation of a consensus of the smart cards completely unlocks the HSM. If this consensus belongs to rogue employees, the HSM is therefore compromised. Since the HSM is the main security component of the architecture and is responsible for all encryption of data, a rogue HSM implies an immediate failure of every security mechanism based on it. It is then necessary that the number of rogue TAG members does

not exceed the smart cards consensus. Without loss of generality, we assume this number should not exceed one.

Ramification #1: Smart Card Security

A direct consequence of the main hypothesis is the security of the Smart Cards used to unlock the HSM. Indeed, considering that the number of rogue TAG members is at most one, it is impossible to reach a malicious consensus in the smart cards. This hypothesis allows us to dismiss the threats of AV-C.

Ramification #2: Source Code Security

Another ramification of the main hypothesis is related to the security of the source code. With a proper review team of competent and honest administrators, the application should hold no intentional vulnerabilities that allow an exploit from the inside or the outside.

This ramification is linked to AV-E with the backend vulnerabilities. Any malicious change in the source code will need to be reviewed by the review team using Pull Requests [120] or similar practices. Therefore, we can dismiss this portion of AV-E under the assumption that the honest TAG members will review and refuse any changes to the source code that pose a security threat to the data.

4.6.3 Attack Scenarios

By studying the Attack Vectors presented in Section 4.5, we propose several attack scenarios that cover all possible Rogue Employee (RE) attacks on the security architecture. These attacks attempt to access, tamper with, or suppress sensitive data. These attacks span across AV-A, AV-B, AV-E and AV-F.

Access to the LMK

The access to the LMK is not an attack in itself but allows the RE to perform several attacks if he succeeds with this step. As shown with AV-F, the application server is listed as an authorized component under the ACL, which allows any process coming from the server to communicate with the HSM and send encryption or decryption requests with any data. Therefore, even though the LMK itself is not directly accessed, the RE can still access the HSM functionalities with the LMK as if they had access to the LMK.

Access to Sensitive Data

The first set of attacks deals with an attempt to illicitly access sensitive data. For this attack to happen, the RE needs at least Read-Only access on the application server and/or the Key Storage Unit (KSU). From there, several variations of the attack are possible.

Theft of the Private Key The Private Key that authenticates the server through the TLS security protocol should remain secret. As long as this secret is preserved, the TLS protocol guarantees the server's identity to the client [28].

If the attacker gains access to the LMK, they can use AV-A vulnerabilities to steal the private key from the KSU or AV-B to intercept the private key from the communication channel. They can then decrypt the private key with the LMK and use it to create a fake server identity with the clients.

On one hand, the attacker can initiate a malicious TLS session with the client and act as a Man-in-the-Middle between the client and the server. This attack targets both the confidentiality and integrity of data. On the other hand, the attacker can eavesdrop on the TLS handshake and use the stolen private key to decrypt the session key between the client and the server. With this session key, the attacker can decrypt all exchanged information between the client and the server, which targets the confidentiality of data.

Replacement of the Private Key Another attack, similar to the previous one, replaces the private key with a key of the attacker's making.

If the attacker has access to the LMK, they can use the vulnerabilities on AV-A and AV-B to provide the HSM with a fake key encrypted by the LMK. Since the HSM will successfully decrypt the key and load it in its secure environment, no warning will be raised. However, the attacker can access the private key and initiate a malicious TLS session with the client. The impact on confidentiality and integrity is similar to the previous attack.

Access to the database A third attack related to data theft is access to the encrypted database. Therefore, the attacker first needs access to the LMK of the HSM. Then, the attacker can exploit the vulnerabilities in AV-E.2 to extract encrypted database information. To complete the attack, they feed the encrypted data to the HSM decryption mechanism and obtain the plain-text information.

These three attacks are easily doable by an insider attacker that can gain access to the LMK

of the HSM. They allow the attacker to jeopardize the *confidentiality* and *integrity* of sensitive information.

Tableau 4.2 Summary of the attack scenarios

Attack	**Description**	**Effect**
LMK Access	Encryption or decryption from an unauthorized process on an authorized network component	N.A.
Theft of Private Key*	Decryption of the private key from the KSU to impersonate the application server	Confidentiality
Replacement of Private Key*	Encryption of a malicious private key in the KSU to impersonate the application server	Confidentiality
Access to the database*	Extraction of encrypted data from the database files and decryption using the LMK	Confidentiality
Alteration*	Replacement of a field in the database with a fake encrypted value	Integrity
Corruption	Replacement of a field in the database with a random byte array	Availability
Swapping	Replacement of a field in the database with another field	Integrity
Nullification	Deletion of a nullable field in the database	Availability
Deletion	Deletion of an entire entry in the database	Availability

*Requires access to the LMK

Data Tampering

The second set of attacks deals with the illicit modification of data. For these attacks to happen, the RE needs Read-Write access to the application server files. Three attacks are possible: alteration, corruption, and swapping of data. These attacks differ by the abilities and the goal of the attacker.

Alteration If the RE can access the LMK (see Section 4.6.3), they can encrypt any value they wish to store in the database. Then, they must replace the target value with the fake value inside the database table. This attack jeopardizes the integrity of the stored data.

Corruption If the RE cannot access the LMK, they can still corrupt the existing data by replacing it with a randomly generated byte array. This action is more elaborate than simply deleting the value from the database, as some Database Management Systems (DBMS) can raise errors when a non-null constraint is enforced. The DBMS cannot detect corruption with this attack, but the actual information is lost.

Swapping Regardless of the RE's access to the LMK, they can swap two target values in a database. Provided with prior knowledge of a second target, the RE can either swap the two values or copy the value of the second target inside the first target to tamper with the original information. This attack does not require access to the LMK because the HSM already encrypted the fake value. The DBMS will not detect incoherence, and the HSM can decrypt the stored information as if no attack occurred. However, that information will be fake and unrepresentative of reality.

These three attacks are all related to the database aspect of AV-E and allow the attacker to jeopardize the *integrity* (Alteration and Swapping) or *availability* (Corruption) inside the database.

Dat Suppression

The third set of attacks deals with suppressing any information in the database. The RE needs Read-Write access on the application server. Two attacks are possible in this set: nullification and deletion of data. These attacks differ by the amount of suppressed data.

Nullification The "corruption" attack presented in Section 4.6.3 is designed to avoid the DBMS errors of a non-null constraint. However, some fields in the database can be "nullable," meaning the field can hold an empty value. A possible attack would be accessing the database files and deleting the value a nullable target field holds. The DBMS will not detect the attack nor know if the field was legitimately or illicitly nullified.

Deletion The deletion attack is another kind of attack aimed at suppressing data from the database. In contrast to the nullification attack, the deletion attack attempts to delete an entire entry from a table in the database. The RE can successfully delete an entire table entry by accessing the database files or the SQL command prompt. Outside the context of a cyber-attack, data suppression is a regular operation. For example, if a user unsubscribed from the product the application offers, removing their information from the database would

be normal. Hence, the DBMS cannot know if an entry was legitimately or illicitly removed, making the attack undetectable.

These two attacks are related to the database aspect of AV-E and allow the attacker to jeopardize the *availability* of data. However, in contrast to the tampering attacks, the attacker does not need access to the LMK of the HSM to perform those attacks.

The entire set of possible attacks is summarized in Table 4.2.

4.7 Results and Discussion

To verify the accuracy of the proposed attack model, we simulated the security architecture within a private network, with a mock application and database. We applied the proposed attack model against this architecture and noted the feasibility and impact of each attack.

4.7.1 Implementation

Our security architecture comprises 2 components: the HSM and the application server.

The application server contains the key storage unit, the application's source code, and the database server. The key storage is simply a dedicated directory in the file system. The application's backend is programmed in Python using the Flask framework [121], and it is available at port 5005 of the server. The database is deployed using PostgreSQL [122] at port 5432. The database contains a table called "seedings", which has three columns:

— "id": a clear-text integer, sequentially increasing;

— "salary": a non-null encrypted field;

— "name": a clear-text nullable character array.

Since the HSM is a very costly component, we replicated the functionalities of an HSM in a series of Python scripts that open different TCP ports for each functionality, such as encryption, decryption, hashing, and random number generation.

We used the software Docker [123] to create the two network components using an Ubuntu image. The two Docker containers are isolated in a private network to simulate the *Inside* part of the architecture. Since the scope of this work is insider attacks, we do not need to connect the inside network to the outside network.

4.7.2 Attacks Simulation

Our attack model claims that the LMK is vulnerable to being used by an unauthorized process on an authorized network component. This requires a username and password to access the application server, which leads to conducting the attack. We consider the Rogue Employee (RE) to have a valid username and password to open a session on the server. Once connected, a Python script for each attack is created.

Theft of Private Key

Stealing the private key from the KSU can be done in the following way:

1. Open the file containing the encrypted private key;
2. Decode the ANS.1 format of the encrypted private key under PKCS#8;
3. Extract the key derivation algorithm, the key derivation salt, and the initialization vector from the private key;
4. Open a TCP connection to the HSM from the authorized network component;
5. Send the private key payload to the decryption port with the decryption parameters;
6. Receive the decrypted key.

In our case, we simulated the attack by creating a private key with the HSM. The HSM returns the encrypted private key in the PKCS#8 format. It is distinguished by its header: "BEGIN ENCRYPTED PRIVATE KEY". Using an ANS.1 decoder [124], we extract the information for our private key, as illustrated in Table 4.3.

Tableau 4.3 Encrypted Private Key Information from ASN.1 decoding

Field	**Value**
Key Derivation Algorithm	PBKDF2 [125]
Key Derivation Salt	0x0BF3434A9BC51F8C
Key Encryption Algorithm	AES-256-CBC [23]
Key Encryption Initialization Vector	0x80B40A31933A568F5E42A2B65B9B774D
Encrypted Private Key	1232 bytes of hexadecimal values

One notable mention is that since the standard of PKCS#8 uses a key derivation algorithm [126], the key used to decrypt the private key is not the LMK but a derivation of the LMK using PBKDF2 [125]. Hence, we must provide the HSM with the payload to decrypt, the derivation algorithm, and the salt to perform this attack.

To verify that the key we decrypted corresponds to the original key, we cannot simply compare the two keys because the original never left the tamper-proof partition of the HSM. Therefore, we applied the following steps:

1. Sign a dummy payload using our stolen decrypted private key;
2. Request an HSM verification of the signature using the encrypted private key.

For a payload of "Hello, World!", the signature created with the stolen private key was verified using the encrypted private key. This proves the private key was properly decrypted outside the safe environment by abusing the HSM cryptographic abilities.

Replacement of Private Key

For the second attack, we operated very similarly to the first attack. However, instead of decrypting an existing key, we generated a random IV, a random salt, and a new private key. The malicious private key was then encrypted using the same algorithms, encoded using the ASN.1 format, and stored in the server instead of the honest private key. The verification procedure shows the malicious key was properly encrypted and placed in the key storage unit.

Access to the Database

With access to the DBMS software, we performed the following SQL query to get the data related to John Doe: `SELECT * FROM seedings WHERE name='John Doe';`. The query result is reported in Table 4.4.

The "salary" field contains the Initialization Vector and the encrypted value in Base64 encoding. By feeding this value to the HSM decryption service, we notice the encrypted field corresponds to the value 2500. We conclude that John Doe's salary is 2500, which constitutes a data confidentiality breach.

This attack uses unrestricted access to the HSM resources from the authorized network component.

Data Alteration

On the same table, we now modify the salary of John Doe to write 5000 instead of 2500.

We request the HSM to encrypt the value "5000" and obtain the encryption result. With access to the DBMS software, we perform the UPDATE query reported in the 2nd row of Table 4.4. The result confirms the changes have been saved to the database.

Tableau 4.4 Result of the SQL queries

SQL Query	Result
`SELECT * FROM seedings WHERE name='John Doe';`	**id** | **salary** | **name** 1 | `9CF...C86:IV:KmN...lyg==` | John Doe
`UPDATE seedings SET salary='DAA... 86C:IV:jQ2...r0w==' WHERE name='John Doe';`	UPDATE 1
`UPDATE seedings SET salary='DAA... 86C:IV:jQ2...o31==' WHERE name='John Doe';`	UPDATE 1
`SELECT * FROM seedings WHERE name='John Doe' or name='Alice Smith';`	**id** | **salary** | **name** 2 | `928...311:IV:Txy...dyA==` | Alice Smith 1 | `9CF...C86:IV:KmN...lyg==` | John Doe
`UPDATE seedings SET salary='928... 311:IV:Txy...dyA==' WHERE name='John Doe';`	UPDATE 1
`UPDATE seedings SET name=null WHERE name='John Doe';`	UPDATE 1
`DELETE FROM seedings WHERE name='John Doe';`	DELETE 1

To verify the effectiveness of the attack, we send an API request to the backend server to get John Doe's data. The returned result contains a salary field of 5000 instead of 2500, proving the attack was successful.

Data Corruption

With access to the DBMS software, we perform the UPDATE query reported in the 3rd row of Table 4.4. The value inserted for the salary field was handmade to resemble a value generated by the HSM. To achieve that, we concatenated 16 bytes in hexadecimal to resemble the IV, the ":IV:" separator, and random bytes encoded in base 64. The result confirms the changes have been saved to the database.

However, when we sent the API request to get John Doe's data, we received a 500 HTTP Error instead of the desired data. By further analyzing the error log, we find an issue with the unpadding of data at the HSM level after decryption. While this attack succeeded in jeopardizing the availability of data, it failed the undetectability requirement. Since the padding is performed before the encryption, predicting which bytes sequence for the ciphertext would generate a valid padding after decryption is impossible.

Data Swapping

With prior knowledge that Alice Smith has a salary of 3500 and with access to the DBMS software, we perform the SELECT query reported in the 4th row of Table 4.4. With this information, we copy the value for "salary" for Alice Smith and perform the UPDATE query reported in the 5th row of Table 4.4.

We then send an API request to get John Doe's data. The returned result contains a salary field of 3500 instead of 2500, proving the attack was successful.

Data Nullification

With access to the DBMS software, we perform the UPDATE query reported in the 6th row of Table 4.4.

We then send an API request to get John Doe's data. The result contains an empty name field (null) instead of John Doe's name, proving the attack was successful.

Data Suppression

With access to the DBMS software, we perform the UPDATE query reported in the 7th row of Table 4.4.

We then sent an API request to get John Doe's data. However, there is no result since the row was completely deleted, proving the attack was successful.

4.7.3 Evaluation

To evaluate the impact of the presented attacks on our system, we rank them according to their difficulty and impact.

The difficulty of an attack is based on the sum of obstacles when performing an attack, each having a different weight (p) relative to their difficulty. The possible obstacles in our situation are access to the HSM (1.5), access to the database (1), prior knowledge (1), and cryptographic obstacles (3).

The impact of an attack is based on the number of security pillars affected by the attack, all in the same weight ($n = 1$). The three known security pillars considered in this paper are confidentiality, integrity, and availability.

The score of an attack is reported by the value $\delta = \sum n / \sum p$ where $\sum p$ is the sum of all the applicable obstacles weights and $\sum n$ is the sum of all the appropriate pillar weights for

this attack. The δ-score of an attack represents the impact of the attack versus its difficulty. Therefore, an attack with a high δ-score is much more effective than difficult. Conversely, an attack with a low δ-score is considered too complex for its impact on the architecture.

To calculate all the attack scores, we plot the diagram of insider attacks as shown in Fig. 4.2.

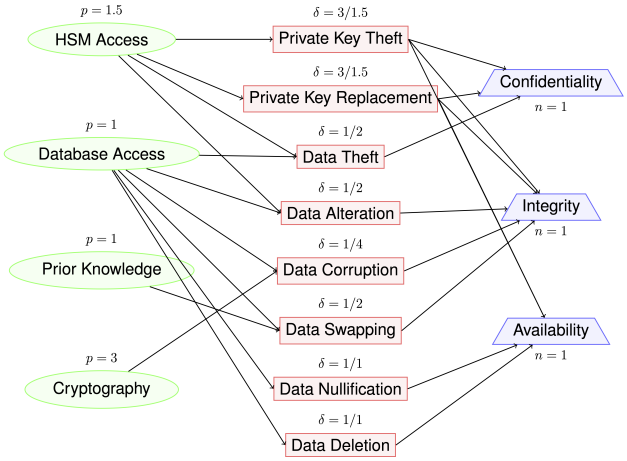

Figure 4.2 Diagram of the impact and difficulty of insider attacks

In this figure, the weighted positive degrees of the attack nodes (in red) represent the difficulty rating of the attack. The negative degrees of the attack nodes represent the impact rating of the attack. For each node, we calculated the overall score δ.

Attacks with a score of $\delta > 1$ are the most dangerous because their impact is worth more than their difficulty. Attacks with a score of $0.5 \leq \delta \leq 1$ are moderately dangerous as their complexity is somewhat equivalent to their impact. Attacks with a score of $\delta < 0.5$ are not very dangerous because their difficulty greatly surpasses the attack's impact, making it not worth the effort to implement.

4.7.4 Discussion

The attacks simulation presented in Section 4.7.2 shows how vulnerable a cyber system is against insider attacks. Even if robust security components are installed, they might not be properly configured to detect and prevent insider attacks.

Among the presented attacks, all but one were successful. Indeed, data corruption without access to the HSM was challenging to achieve due to the encryption of the stored data. We completed all the other attacks in a reasonable amount of time. Key theft, key replacement, database access, and data alteration require access to the HSM to be executed. However, we

found that the data swapping, data nullification, and data suppression do not require access to the HSM.

The attacks that require access to the HSM are only feasible if the attacker can open a session on the authorized server to communicate with the HSM. Depending on the attacker's position in the organization, this access is more or less granted. On the other hand, all attacks related to the database require access to the DBMS. A variation of these attacks requires the attacker to directly access the database files stored in the server. However, these files can be very numerous, and searching for the correct information can get extremely tedious.

We evaluated these considerations in Section 4.7.3. The computed δ-score for each attack shows that private key theft and replacement are among the most dangerous attacks on our system. On the other hand, data theft, alteration, swapping, nullification, and deletion are moderately dangerous. Finally, data corruption is not dangerous since its δ-score is very low. This evaluation raises awareness of the possible attacks and their impact on our system and highlights the priorities when implementing a defense architecture against those attacks.

4.8 Conclusion

In this paper, we designed an exhaustive insider attack model that targets all possible attack vectors on an HSM-based security architecture. The goal was to show the importance of a clear and detailed attack model on the security architecture. Our model demonstrated that several insider attacks can still be conducted, even with the installation of robust security hardware such as an HSM. The simulations showed how easily an insider can exploit those vulnerabilities. As far as we know, our attack model is the only one that highlights the insider vulnerabilities of HSM-based architectures compared to the other works found in the literature. As a future direction, we must further our research by studying the possible defense mechanisms against the proposed insider attack model.

Acknowledgment

The authors wish to thank Dr. Franjieh El Khoury for her constructive comments and the proofreading of this paper. The authors also wish to thank Mr. Paul Cassarino, Mr. Sandip Patel and Mr. Patrick McLarney from *Enstrust Corporation* for providing useful insights and documentations about HSM.

CHAPITRE 5 ARTICLE 2 : HSM-BASED ARCHITECTURE TO DETECT INSIDER ATTACKS ON SERVER-SIDE DATA

Marc Dib, Samuel Pierre (Senior Member, IEEE)

Mobile Computing and Networking Research Laboratory (LARIM), Department of Computer and Software Engineering, Polytechnique Montréal, Montreal, QC H3T 1J4, Canada

E-mail : marc.dib@polymtl.ca ; samuel.pierre@polymtl.ca

Revue : Soumis pour publication le 31 Janvier 2024 dans le journal IEEE Transactions on Information Forensics and Security.

ABSTRACT

In this paper, we propose an HSM-based architecture to detect insider attacks on server-side data. Our proposed architecture combines four cryptography-based defense mechanisms: Nonce-Based Process Authentication (NBPA), Hash-Based Field Integrity (HBFI), Hash-Based Field Availability (HBFA), and Hash-Based Row Availability (HBRA). This novel architecture is designed to detect a predefined comprehensive attack model on server-side data tailored for an HSM-based architecture. The implementation results show that the throughput decrease is mostly manageable (14% for NBPA, 30-50% for HBFI, 25% for HBFA, and 43.74% for the combination of all mechanisms), with the indication that some mechanisms are more or less appropriate depending on the situation. Moreover, the HBRA mechanism performed well regarding the attack detection time (5 minutes for a database of 1000 entries).

INDEX TERMS

Cryptography, confidentiality, hardware security module, insider attacks, integrity.

5.1 Introduction

Storing sensitive information on a distant server is essential to the application mobility paradigm. Unlike storing data on one terminal, it allows users to access information from any device connected to the internet. However, this distant storage solution increases the risk of

breaches, leaks, or alteration of information [112]. Many medical and financial organizations have been targeted by attacks designed to affect data security [112, 113].

Typical cyber systems are built on the Inside-DMZ-Outside architecture, where the typical focus is to protect the system against external hackers [115]. However, a third of the attacks originate from inside the system [107, 108]. Insider attacks are notorious for being difficult to track, since the attacker may already have access to some sensitive information. Moreover, they can have disastrous consequences, as we can see for example in the Desjardins case in Quebec, Canada, where an insider leak of client information cost the organization CA$200 million in damage and retribution to their clients [113].

A large portion of the literature on insider attacks relies on behavioral analysis and machine learning. In contrast, other methodologies adopt a more technical and on-site approach, using cryptography to improve security. We believe a robust cryptography-centered architecture could drastically increase security against insider attacks.

In this paper, we adopt the 5-step approach to data security, as described by the Center for Internet Security (CIS) in their Community Defense Model white paper [90]. The *identification* step requires knowing what vulnerabilities or attack vectors are exposed in our cyber system. This can be achieved by listing, detailing, and classifying the possible insider attacks. The *detection* step corresponds to the strategies we implement to detect an attack on data. It comprises tools, security protocols, or methodologies that allow the security team to be alerted to any insider attack. The *protection* step corresponds to the security elements designed to prevent attacks (i.e., data encryption and Intrusion Prevention Systems). The *response* step comprises the technical and management procedures to follow after an attack (i.e., documentation and logging). The *recovery* step represents the available means to return to normal functioning after an attack (i.e., restoring data from the backup and launching the backup server). While the vast majority of related works concentrate on the *response* and *recovery* steps, the scope of this paper revolves around the *detection* of insider attacks.

One asset to achieve security is using a Hardware Security Module (HSM) to reduce the vulnerabilities of our system. Indeed, an HSM acts as a root of trust in a system for all cryptographic operations [86, 116, 127]. Its Local Master Key (LMK) stored in the tamper-proof partition allows a more robust security scheme than regular computers [87]. Moreover, in comparison to regular computers or other contestants, such as the Intel Software Guard eXtension (SGX) Module [128], the HSM is resistant to side-channel attacks [49].

However, achieving full-scale security using HSM requires an architecture maximizing the use of this module. Hence, in our previous research work [85], we proposed an HSM-based architecture to protect server-side data. In another one of our previous works, we proposed

an Insider Attack Model against HSM-Based Architecture [109]. This work was relative to the *identification* step; in other words, we elaborated on the possible insider attacks, their feasibility, and their impact on the system. In this work, we aim to continue in the insider attack mitigation methodology by working on the *detection* step for the identified insider attacks.

Hence, we propose an HSM-based architecture to detect insider attacks on server-side data. The main contributions of this paper are as follows:

— We proposed an adaptation of the PKCS #11 API to ensure a robust continuous authentication;

— We proposed a user-transparent data restructuration to ensure data integrity and availability;

— We proposed an optimized continuous entry validation mechanism to ensure data availability in the background;

— We evaluated the proposed architecture using qualitative and quantitative metrics to showcase its efficiency.

This paper's originality relies on the novel combination of cryptographic functions on an HSM to achieve data security. Beyond a simple combination of cryptographic primitives, our work is a discreet and efficient restructuring of data storage that detects any breach in confidentiality, integrity, or availability. Moreover, we propose simple additions to the PKCS #11 standard that reinforce the authentication of the cryptography API. We believe that our preventive approach adds a significant level of data security compared to existing methods.

The rest of this paper is organized as follows. Section 5.2 details the technical background knowledge. Section 5.3 presents a critical review of the existing defense mechanisms and architectures regarding insider attack mitigation. Section 5.4 describes the proposed architecture with all its defense mechanisms. A discussion of the implementation and the obtained results is presented in Section 5.5. An integration guide to real-world environments is included in Section 5.6. Finally, Section 5.7 concludes the paper.

5.2 Background

In this section, we define the necessary background on data security, insider attack models, cryptographic operations, and queuing theory.

5.2.1 Data Security

Data security is the implementation of confidentiality, integrity, and availability in a cyber system [15]. This work is centered on the secure storage of sensitive information. Therefore, we consider the data security aspects relative to *at rest data*. As such, we define the three pillars of security, adapted to our paradigm, as follows:

— Confidentiality concerns data secrecy, so only authorized parties can access and decipher sensitive data,

— Integrity is related to the truthfulness of data to prevent an attacker from altering the information. It ensures that data must not be tampered with once it is stored in the database,

— Availability guarantees that data is always accessible to the client when requested. In other words, data must not be illicitly removed or corrupted in the database.

5.2.2 Insider Attack Model and Baseline Architecture

Insider attacks are a category of attacks targeting a cyber system. They differ from outsider attacks because the attacker belongs to the organization and has authorized access to some network components [118,119]. In severe cases, the attacker is an administrator with extensive access to the network components. Insider attacks are accidental when they result from negligence and poor work. However, they are intentional when the harm is done intentionally to steal, alter, or suppress sensitive information [108].

Baseline Architecture

A defense architecture proposal must be based on the same baseline architecture as the insider attack model, as depicted in Figure 5.1.

This architecture defines the Trusted Computing Base (TCB) as follows. All application processes are launched on the server. These applications are considered well-implemented regarding security, bugs, and maintainability. Access to the server is possible for some employees, such as DevOps and administrators. The cryptographic aspect is delegated to a Hardware Security Module (HSM) to ensure more robust security, which communicates with the Server through the PKCS #11 API. The HSM is considered fully trusted as long as the consensus of smart cards for admin control remains trustworthy.

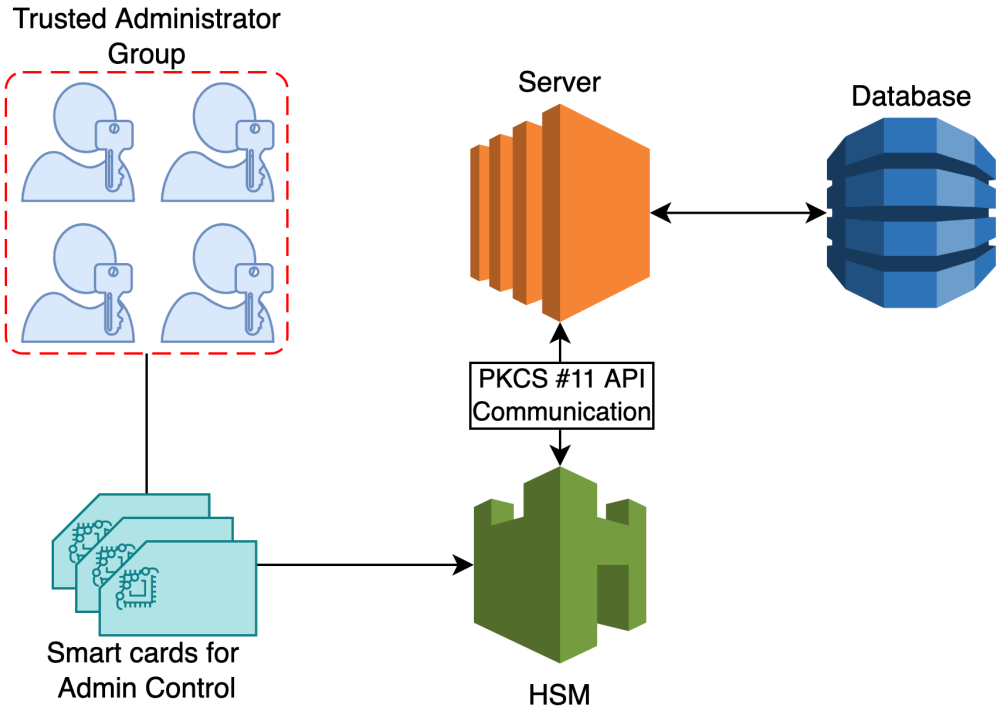

Figure 5.1 Baseline Architecture Layout

Insider Attack Model on the Baseline

In our previous paper [109], we justified the need for a thorough insider attack model before designing and implementing security solutions. Moreover, this exhaustive attack model shows the existing vulnerabilities in the baseline architecture and serves as a launching point for our proposed architecture in this paper.

According to the attack model, a rogue employee could conduct an insider attack on the following Attack Vectors (AV): the private key storage unit (KSU), the key transfer between the KSU and the HSM, the database server, and the communication channel between the application server and the HSM [109]. From these AVs, the attacker can:

— Steal or replace the private key, either from the KSU or during its transfer to the HSM,

— Extract, alter, corrupt, swap, nullify, or delete sensitive data from the database, either from the database server disk or through the Database Management System (DBMS),

— Encrypt fake data or decrypt sensitive information by abusing the communication channel between the application server and the HSM.

These attacks can be conducted by any employee with regular AV access. Therefore, they are flagged as untrusted, and data stored or in transit between the AVs must be protected accordingly.

5.2.3 Cryptographic Functions Notations

In this paper, we adopt the following notations:

— $\{x\}_K$ represents the symmetric encryption [22] result of x using the key K, such as AES-256-CBC [24];

— $h_n(x)$ represents the hash of x using an n-long hashing function, such as SHA-256 [38, 39];

— $x|y$ represents the concatenation of arrays x and y;

— $\widehat{x}$ represents the observed value of x in the database.

5.2.4 Queuing Theory

Using Kendall's notation [129], we define a single-server application serving worldwide populations as an M/M/1/∞/∞/FCFS [130] queue, shortened M/M/1. It allows us to assess the application's capacity to serve clients [131, 132].

In M/M/1 queues, we denote by λ the arrival rate in requests per second (rps) and by μ the service rate in requests per second. To maintain stability, the stability factor $\rho = \lambda/\mu$ must remain below 1 [130]. Therefore, we measure the service rate μ_{eff} of a given process to determine the stability limit $\lambda_{\max} = \mu_{\text{eff}} - \epsilon$ for which the queue remains fully stable, ϵ being a very small number.

While evaluating a security strategy, the measurement of μ_{eff} determines the loss of application throughput induced by the security strategy.

5.3 Related Work

In this section, we critically review the existing literature for detecting insider attacks. We divide the existing work into two categories: *cryptographic* methods for insider attack detection and *non-cryptographic* methods, such as behavioral analysis, machine learning, and network-based architectures.

Mousa *et al.* [84] tackle the database security issue by exposing the challenges and known solutions. Among the noted threats are database misconfiguration, denial of service attacks, unmanaged sensitive data, backup exposure, and excessive privileges. These issues are often treated with the mindset of outsider attackers. However, the methodology to adopt against insider attackers is much more complex, given the privileges of an insider. The attack model we devised on the baseline architecture presented in Section 5.2.2 shows that an insider attacker can circumvent the most common security strategies [109].

5.3.1 Cryptographic Methods for Insider Attack Detection

Many proposed methods in literature use cryptography to detect or prevent insider attackers.

Database encryption using symmetric encryption (i.e., AES) is the most evident tool for achieving confidentiality [84, 90, 98–100]. However, the nature of insider attacks makes it difficult to hide the encryption key, making the encryption useless.

In one of our previous works, we proposed an architecture based on a Hardware Security Module (HSM) to protect server-side data. The three cryptographic modules worked harmoniously to protect the symmetric and asymmetric encryption keys, the database contents, operations, results, and database backups. However, our Insider Attack Model [109] showed that encryption keys are still subject to malicious indirect use. Indeed, a cryptographic key protected with a PIN-based token, according to PKCS #11 [133], is still vulnerable when someone inside the organization knows the PIN.

Priebe *et al.* [134] proposed a secure database using the Intel SGX module. Their solution induces around 40% of overhead, which is considered acceptable. However, the extent of the memory of this trusted hardware could hit a limitation in large databases with lots of sensitive information. Moreover, the authors trust the key management system and do not consider side-channel attacks.

Egbunike and Rajendran [97] proposed a Negative Database scheme targeting the alteration of data inside a database. Their method can be static or dynamic. However, their evaluation shows that the Negative Database is time-consuming and, therefore, unscalable. Moreover, with the static method, the data is at risk of being exploited by a rogue administrator, and the dynamic method requires additional user input, which can be a burden.

Xu *et al.* [101] proposed a data integrity verification method relying on cryptography in the context of relational databases. Their method provides a solution against the data tampering and swapping of our previous attack model [109]. However, their method could face complications when data is genuinely deleted from the database, and line numbers shift. Moreover, their solution implies a strong storage and computation overhead. While this solution is a promising candidate, it contains several flaws that should be resolved.

5.3.2 Non-Cryptographic Methods for Insider Attack Detection

One standard methodology for detecting insider attacks is based on behavioral analysis. Caputo *et al.* [135] propose a methodology to detect insider theft of trade secrets. Their work extensively describes the activity types, warning signs, and detectors of malicious insider behavior. However, their work focuses more on identifying those features than actual detec-

tion. Using artificial intelligence and machine learning on those features could increase their chance of detecting insider attacks by analyzing and recognizing common patterns for insider attacks.

Indeed, Yuan and Wu [136] study the use of deep learning for insider threat detection. The authors showcase many examples of neural networks. However, the authors also note the challenges of using deep learning for insider threat detection, such as subtle attacks and adaptive threats. Indeed, detection algorithms based on human actions are inconsistent and can be evaded. Moreover, new ways of conducting attacks can emerge to avoid the detection algorithm.

Moreover, Duncan *et al.* [96] proposed a combination of Attack-Tree and Kill-Chain approaches to detect insider attacks on cloud services. Their approach relies on creating a tree that exposes the possible actions to perform an attack. Their method was applied to cloud services against a network tap attack. However, their context vastly differs from ours, and their proposed improvements do not apply to our approach.

Non-cryptographic methods for insider attack detection can also include more technical approaches, such as restructuring the network architecture or database management system.

Solutions centered on firewalls [82, 83] can counter privilege escalation attacks and unauthorized access. However, these solutions are more effective against outsiders than insiders.

Moreover, the Digital Forensics approaches [82] are centered on studying previous attacks conducted on the system through log files. In the 5-step approach to data security, this method falls under the *response* and *recovery* steps instead of the *detection* step.

Li *et al.* [137] proposed a methodology to make encrypted databases maintainable while achieving high security. Their method is based on the forking of the Virtual Machine (VM) hosting the database for Database Administrator (DBA) purposes. However, their solution does not clearly differentiate a benign query from a malign query in the regular database instance. Moreover, forking a new virtual machine for DBA purposes could be very costly regarding resources, especially in distributed environments.

Regarding data availability, the most trivial countermeasure is to maintain frequent and secure backups [84, 85, 138]. However, maintaining backups falls under the *recovery* step for the 5-step insider threat management. We need a separate mechanism to detect subtle attacks (i.e., deletion of one row or field) to launch the recovery.

5.4 Proposed Architecture

To defend the system against the attack model proposed in [109], we design a security architecture based on the combination of five defense mechanisms, as shown in Fig. 5.2. In this section, we detail the defense mechanisms *Enhanced StealthDB* (ESDB), *Nonce-Based Process Authentication* (NBPA), *Hash-Based Field Integrity* (HBFI), *Hash-Based Field Availability* (HBFA), and *Hash-Based Row Availability with Optimized Sliding Window* (HBRA).

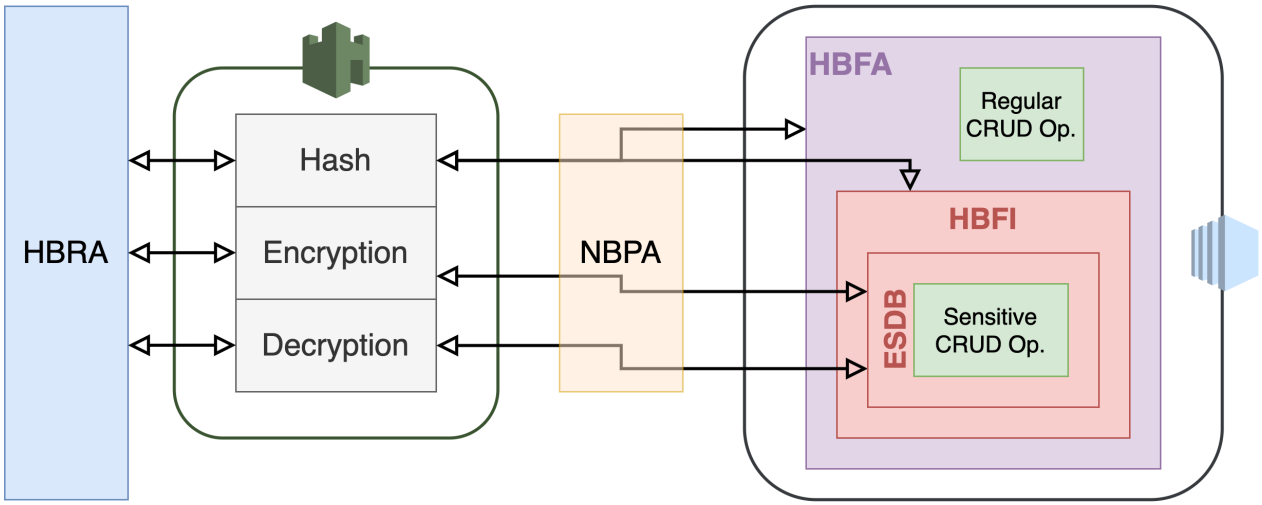

Figure 5.2 Counter-Attack Architecture Against Insider Attacks

The ESDB defense mechanism comes directly from the initial architecture of our previous work [85]. It uses the cryptographic abilities of the HSM to encrypt data before storing them in the database. It delegates the SQL comparison of encrypted fields to the secure element inside the HSM CodeSafe. It ensures confidentiality in the database. While the design and implementation of this defense mechanism result from our previous work [85], we must mention it in this paper to obtain a complete architecture.

5.4.1 Nonce-Based Process Authentication

The first defense mechanism authenticates processes inside a network component, such as the backend server, to communicate with the HSM. The attack scenarios of private key theft, private key replacement, access to the database, and alteration in our attack model [109] showed that insider tampering is possible. Even with token-based protection, an insider attacker might learn about the token PIN and use it to call the HSM cryptographic functions outside the scope of the trusted application.

To ensure a robust but light authentication mechanism, we propose a variable PIN au-

thentication using nonce values, which are random values with a unique usage [139]. The Nonce-Based Process Authentication (NBPA) process is presented in Fig. 5.3.

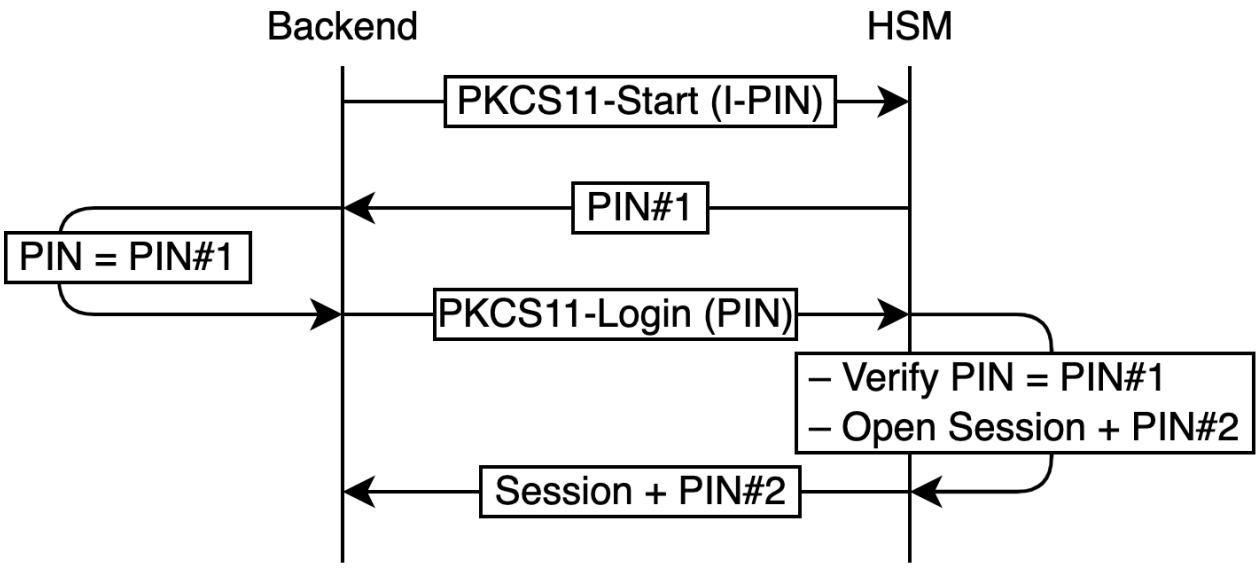

Figure 5.3 Nonce-Based Process Authentication

The process starts with an Initialization PIN (I-PIN). The I-PIN combines k among n PINs, where k and n create a pre-determined consensus (i.e., 2 of 3, 3 of 6). This eliminates the risk of one rogue employee being able to start the process. When the HSM validates the I-PIN, it generates a random PIN and sends it to the application. This PIN must be stored in memory only to make its access more difficult. From that point, every PKCS #11 Login Call from this process must contain the correct PIN. The HSM verifies the PIN and opens the session. This session includes a new PIN for the next PKCS #11 Login Call. The goal of using nonce values as PINs is to prevent replay attacks.

5.4.2 Hash-Based Field Integrity

The second defense mechanism ensures the integrity of fields in a database. The swapping attack scenario in our attack model [109] showed that an attacker could tamper with the database without access to the HSM. The attacker only needs to swap two values in different rows in the database. The challenge here is to modify the database structure to guarantee integrity. We also aim to make a discreet modification that does not require any additional user action and minimizes the changes on the application itself.

We propose the Hash-Based Field Integrity (HBFI) defense mechanism, which modifies the field structure in the database to detect tampering attacks. Let x be the plain-text value of

the field, and PK_x the Primary Key of the corresponding entry in the table, we compute s being the hash of x and PK_x using an n-long hashing function h_n, as illustrated in (5.1).

$$s = h_n(x|PK_x) \tag{5.1}$$

We also compute d being the concatenation of x and its special hash s, as illustrated in (5.2).

$$d = x|s = x|h_n(x|PK_x) \tag{5.2}$$

We use the best available encryption algorithm provided by the HSM and the Application Key (AK) created for symmetric encryption. We store the value c in the database, given by (5.3).

$$c = \{d\}_{\text{AK}} = \{x|h_n(x|PK_x)\}_{\text{AK}} \tag{5.3}$$

Let $\widehat{c}$ be the observed value of a field in the database. The HSM decrypts $\widehat{c}$ to obtain $\widehat{x}$ and $\widehat{s}$. Then, the HSM computes $s = h_n(\widehat{x}|\widehat{PK})$ where $\widehat{PK}$ is the observed Primary Key of the corresponding entry. The HSM compares the computed hash s to the observed hash $\widehat{s}$. The verification works only if $x = \widehat{x}$ and $PK_x = \widehat{PK}$.

An example of the HBFI mechanism is illustrated in Table 5.1.

Tableau 5.1 HBFI Mechanism Applied on an SQL Table

ID	**Name**	**Salary**
1	Alice	$\{2000\|h_n(2000\|1)\}_{\text{AK}}$
2	Bob	$\{5000\|h_n(5000\|2)\}_{\text{AK}}$

5.4.3 Hash-Based Field Availability

The third defense mechanism ensures the availability of nullable fields in a database. The nullification scenario in our attack model [109] showed that an attacker could illicitly remove nullable fields without detection. Similarly to HBFI, our goal is to guarantee availability through a discreet and efficient modification of the database structure.

We propose the Hash-Based Field Availability (HBFA) defense mechanism, which modifies the database structure to detect nullification attacks. We add an extra column to the table for hash verification purposes. Let $r_{j,i}$ be the value of the i^{th} column and j^{th} row of the table. The HSM computes H_j being the n-long hash of all the columns in j and its encrypted value

C_j, as illustrated in (5.4) and (5.5).

$$H_j = h_n\left(r_{j,1}|r_{j,2}|...\right) \tag{5.4}$$

$$C_j = \{H_j\}_{\text{AK}} = \{h_n\left(r_{j,1}|r_{j,2}|...\right)\}_{\text{AK}} \tag{5.5}$$

Let $\widehat{C_j}$ be the observed value for the hash of row j. The HSM decrypts $\widehat{C_j}$ to obtain $\widehat{H_j}$. Then, the HSM computes $H_j = h_n(\widehat{r_{j,1}}|\widehat{r_{j,2}}|...)$ where $\widehat{r_{j,i}}$ is the observed value at the time of verification. The HSM compares H_j and $\widehat{H_j}$. The verification works only if $r_{j,i} = \widehat{r_{j,i}}, \forall i$.

An example of HBFA is illustrated in Table 5.2.

Tableau 5.2 HBFA Mechanism Applied on an SQL Table

ID	**Name**	**Salary**	**Row Hash**
1	Alice	2000	$\{h_n(1\|\text{Alice}\|2000)\}_{\text{AK}}$
2	Bob	5000	$\{h_n(2\|\text{Bob}\|5000)\}_{\text{AK}}$

5.4.4 Hash-Based Row Availability / Optimized Sliding Window

The fourth defense mechanism ensures the availability of entries in a database. The deletion attack scenario in our attack model [109] showed that an attacker could illicitly remove an entry without detection.

We propose the Hash-Based Row Availability with Optimized Sliding Window (HBRA/OSW), which creates a new table to detect deletion attacks. Let T be a table in the database with m rows and ID_j the unique identifier of the j^{th} row. The HSM computes H_T being the n-long hash of all the rows' unique identifiers and its encrypted value C_T, as illustrated in (5.6) and (5.7).

$$H_T = h_n(ID_1|ID_2|...|ID_m) \tag{5.6}$$

$$C_T = \{H_T\}_{\text{AK}} = \{h_n(ID_1|ID_2|...|ID_m)\}_{\text{AK}} \tag{5.7}$$

However, keeping all the IDs in the same hash is not scalable. To efficiently scatter the IDs in smaller hashes, we require each ID to be referenced in precisely two different hashes. Inspired by the TCP Sliding Window mechanism [140], we create the Optimized Sliding Window (OSW) to separate the hashes. Let L be the maximum number of IDs in one hash, we get the following arrangement:

— $H_{T,0}$ contains ID_1;

— $H_{T,1}$ contains ID_1 to ID_L;

— $H_{T,2}$ contains ID_2 to ID_{L+1};

— $H_{T,3}$ contains ID_{L+1} to ID_{2L} since ID_2 to ID_L are already in two different hashes;

— $H_{T,4}$ contains ID_{L+2} to ID_{2L+1};

— The rest follows the same logic.

An example of HBRA/OSW with eight entries and $L = 4$ is presented in Fig. 5.4.

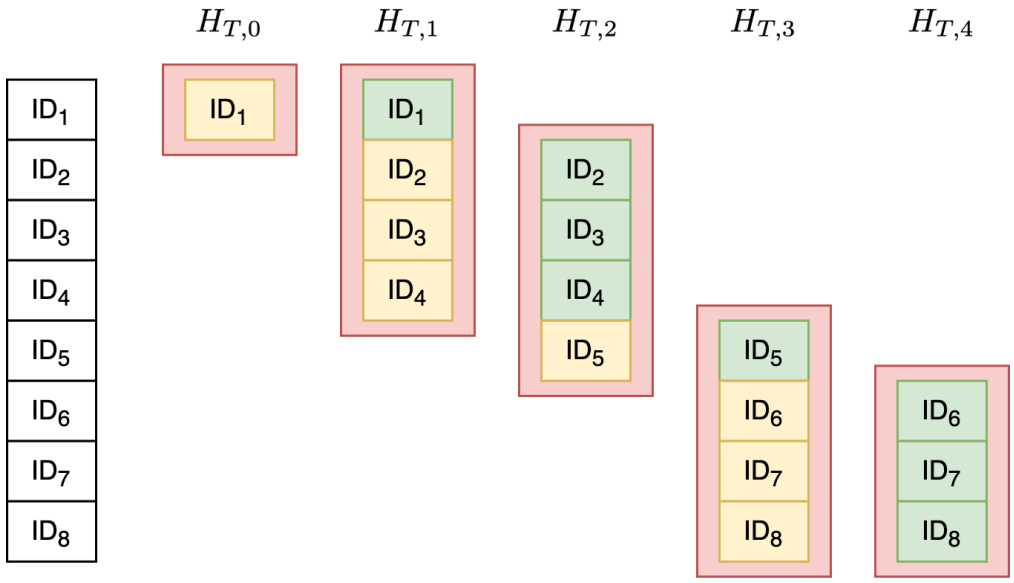

Figure 5.4 Graphical HBRA/OSW Example (8 entries, 4-sized hashes)

The number of hashes required to cover an m-sized table with L-sized hashes is given by η in (5.8).

$$\eta = \left\lceil \left(2 \times \frac{m}{L} \right) \right\rceil + 1 \tag{5.8}$$

When a row is added or deleted, only the affected hashes will be modified to reflect the new state of the database. If one row becomes at risk after a legitimate deletion (i.e., the two hashes are identical), we combine the two adjacent hashes to eliminate the risk.

The verification of the availability of rows is a periodic background process. Indeed, since HBRA/OSW deals with potential missing entries, we do not know in advance where to search. Therefore, a period ΔT determines how often the HSM executes the verification process. The HSM challenges a number k of randomly selected hashes for each verification process.

5.5 Results and Discussion

In this section, we discuss the implementation of the proposed architecture and the results of the qualitative and quantitative evaluation.

5.5.1 Implementation

The proposed architecture was implemented in a system composed of an Application Virtual Machine (AVM) and 2 HSMs.

The AVM is hosted on a server mounted with a 2.10GHz Intel Xeon Gold 5318Y and has 2GB of virtual Random Access Memory (RAM). The AVM contains a testing application developed in Python and a PostgreSQL database [122]. For the evaluation, we create an SQL table called `Seedings`, which contains three fields: `ID` (Integer, Primary Key), `name` (Unicode, Plain-text), and `salary` (Encrypted Integer). The `name` attribute is usually between 10 and 20 characters, whereas the salary is a 4 to 5-digit integer.

The two HSMs are the nShield Connect Base [87], which can handle 795 transactions per second for the AES-256 encryption of 1024 bytes. They are set up in loadshare mode to increase their efficiency. The application and the HSM communicate using the PKCS #11 API [133].

The implementation of the NBPA defense mechanism requires a modification of the HSM's internal structure. It's, therefore, not possible for us to implement it at this stage. However, the implementation of NBPA should go as follows:

— **On the HSM:** the Token protection of an encryption key should accept variable PINs. If this option is enabled, it gives access to a new PKCS #11 function for the I-PIN entry. Moreover, the HSM should hold a database of the most recently generated PINs. Finally, the HSM should accept a new PKCS #11 function to generate the next PIN.

— **On the Application Server:** the Application should always start with a prompt for the I-PIN. Once the I-PIN verification is done, the HSM returns a session PIN that should be stored in memory. Whenever the application needs the HSM, a new session should be opened using the current PIN. The session should also contain the new PIN to be stored in memory and used for the next session. Once a session is open, it can call multiple cryptographic functions without needing a PIN. For example, in one session, we can generate a random IV and encrypt a payload using AES, which accounts for two function calls. However, each session should only be restricted to one end goal to maximize continuous authentication.

To continue with the latency evaluation, we simulate this implementation on a fake HSM

equipped with a fake PKCS #11 API.

5.5.2 Results

A security evaluation determines the added value of the proposed architecture. According to our 5-step methodology for data security, this evaluation asserts the ability of our mechanisms to detect an insider attack effectively. A quantitative evaluation determines the numerical performance of the proposed architecture in terms of latency, memory overhead, and detection time.

Security Evaluation

The security evaluation methodology is based on an analysis of the proposed architecture. During this evaluation, we consider that the security elements and cryptographic algorithms are the best available and contain no known vulnerabilities. We analyze the security implications and their impact for each defense mechanism according to the CIS Safeguards [102].

Evaluation of NBPA The goal of the NBPA is to prevent unauthorized software from accessing the HSM from an authorized network component. An attacker can eavesdrop at the backend server's exit port to retrieve the nonce. However, since the nonce is already going to the HSM and is only valid once, the attacker cannot reuse it because the HSM will have invalidated it. An attacker can eavesdrop at the entry port of the backend server with the HSM to retrieve the value of the nonce. However, they only have a short window before the server uses this nonce and invalidates it. If they manage to use the nonce, they cannot synchronize the legitimate process with the new nonce, and the attack is detected.

Since this mechanism targets unauthorized access to the HSM, it deals with private key theft and replacement, data theft, and data alteration attacks of the attack model. Indeed, these attacks require access to the HSM's cryptographic functionalities. By blocking this difficulty, we protect our system against these attacks.

Evaluation of HBFI HBFI aims to prevent tampering attacks on the database, such as swapping and corruption, without access to the HSM. In the example of HBFI shown in Table 5.1, we can try to swap the *Salary* values for Alice and Bob, as illustrated in Table 5.3.

Upon selecting and verifying row 1, we decrypt and parse the salary into the value 5000 and the reference $h_n(5000|2)$. To proceed with the verification, we compute the hash of the primary key 1 and the value 5000 and compare it with the reference. The computed hash

Tableau 5.3 Swapping attack on an HBFI-protected Table

ID	Name	Salary
1	Alice	$\{5000\|h_n(5000\|2)\}_{\text{AK}}$
2	Bob	$\{2000\|h_n(2000\|1)\}_{\text{AK}}$

$h_n(5000|1)$ differs from the reference. Hence, the attack is detected.

Evaluation of HBFA HBFA aims to prevent the nullification attack on the database for the nullable fields. In the HBFA example presented in Table 5.2, we can try to remove Alice's *Salary* field in row 1, as presented in Table 5.4.

Tableau 5.4 Nullification Attack on an HBFA-Protected Table

ID	Name	Salary	Row Hash
1	Alice		$\{h_n(1\|\text{Alice}\|2000)\}_{\text{AK}}$
2	Bob	5000	$\{h_n(2\|\text{Bob}\|5000)\}_{\text{AK}}$

Upon selecting and verifying row 1, we decrypt the row hash. To proceed with the verification, we compute the hash of all the available field values and compare it with the reference. The computed hash $h_n(1|Alice)$ does not match the reference, so the attack is detected.

Evaluation of HBRA HBRA aims to prevent the row deletion attack by maintaining a record of all rows in a separate table. To verify hash i, we start by selecting the entry IDs for which the hash number is i from the hash association table. Then, we verify that they still exist in the original SQL table and hash the IDs together. Separately, we decrypt hash i from the hash registry and compare the saved hash with the computed hash.

To perform the deletion attack, we can remove the row ID_2 from the original SQL table. The attacker also deletes the ID_2 references from the hash association table. However, they cannot modify $H_{T,1}$ and $H_{T,2}$ in the registry to show the deletion of ID_2. Hence, when the verification algorithm challenges these two hashes, we obtain $h_n(ID_1|ID_3|ID_4)$ for $H_{T,1}$ and $h_n(ID_3|ID_4|ID_5)$ for $H_{T,2}$ which is different from the hashes in the registry.

A possible workaround to avoid detection on $H_{T,1}$ is to delete the hash from the registry and all the entries in it from the database, which are ID_1, ID_3, and ID_4. However, to successfully perform this attack, we must delete $H_{T,0}$ to avoid detection with ID_1. The same goes for the IDs in $H_{T,2}$, some of them which are linked to $H_{T,3}$. This chain of deletion continues until we delete the entire table, which is immediately detectable.

The comparison of the security properties of our proposed architecture with similar work in literature is displayed in Table 5.5. This comparison analyzes confidentiality, integrity, availability, side-channel attack resistance, need for user intervention, and the TCB (where applicable).

Detection Time Evaluation

An important metric to observe when evaluating the efficiency of a solution is the detection time of an attack.

For the NBPA defense mechanism, a nonce theft is detected at the next PKCS#11 login with the HSM (i.e., at the next request). Therefore, the detection is almost immediate in a fast-paced and high-traffic environment.

For the HBFI and HBFA defense mechanisms, an insider attack is detected as soon as the targeted entry is summoned by the DBMS. Indeed, the mechanisms include a verification step when decrypting an entry's protected fields. Since the verification depends on cryptographic hashes, the detection rate is 100%, and it occurs during the request.

For HBRA/OSW, since the mechanism is based on randomly selected hashes to challenge, we adopt a stochastic approach to the evaluation. Let η be the total number of hashes and p_k the probability of success of the Bernoulli experience "At least one hash among the k selected hashes is inconsistent".

The formula for p_k for $k \geq 2$ is illustrated in (5.9).

$$p_k = \frac{2 \times \binom{\eta - 1}{k - 1} - \binom{\eta - 2}{k - 2}}{\binom{\eta}{k}} \tag{5.9}$$

The term $\binom{\eta-1}{k-1}$ represents the presence of at least one wrong hash among the selected hashes. The term $\binom{\eta-2}{k-2}$ represents the case where both wrong hashes are among the selected hashes. This can only happen if $k \geq 2$; else, this term is 0.

Let Z_k be the random variable representing the repetition of this verification; the probability of detecting an inconsistency by repeating the verification follows a geometric distribution. The probability of detecting the inconsistency in at most z attempts is illustrated in (5.10).

$$P(Z_k \leq z) = 1 - (1 - p_k)^z \tag{5.10}$$

The minimal number of attempts $z_{\min}^{(\alpha)}$ for which we guarantee the inconsistency detection with a certainty of α (i.e., 90%) is the solution of (5.11).

$$P(Z_k \leq z_{\min}^{(\alpha)}) = 1 - (1 - p_k)^{z_{\min}^{(\alpha)}} = \alpha \tag{5.11}$$

The solution of this equation is given in (5.12).

$$z_{\min}^{(\alpha)} = \left\lceil \frac{\ln(1-\alpha)}{\ln(1-p_k)} \right\rceil \tag{5.12}$$

Since each verification requires a certain amount of time $\delta t_{k,L}$ and a rest time Δt before the next verification, the minimal detection time with a certainty of α is given in (5.13).

$$t_{\min}^{(\alpha)} = (\delta t_{k,L} + \Delta t) \times z_{\min}^{(\alpha)} \tag{5.13}$$

Hence, the minimal detection time with a certainty of 90% and a rest time of 60 seconds in a database of 1000 entries is depicted in Fig. 5.5.

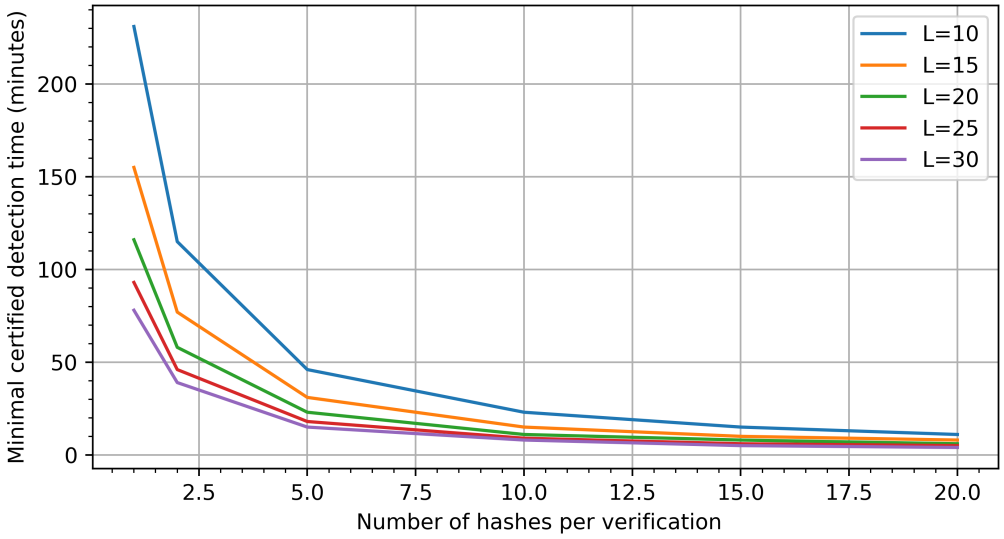

Figure 5.5 Impact of hash size (L) and number of challenged hashes (k) on the minimal time for certified detection

We notice that the values for $t_{\min}^{(\alpha)}$ decrease with L and k. Moreover, these results show that in a table of 1000 entries, with $L = 25$ and $k = 20$, we can detect the attack in under 5 minutes, which is very low.

Latency Evaluation

The latency evaluation allows us to measure our application's service rate loss. This metric is important because we aim to maintain a good quality of service even with improved security.

Latency of NBPA To avoid overloading the application, we measure the NBPA execution time on a simple encryption request to the HSM. The measurements for the latency of NBPA are depicted in Fig. 5.6. We also compute the service rates with and without NBPA and their relative difference.

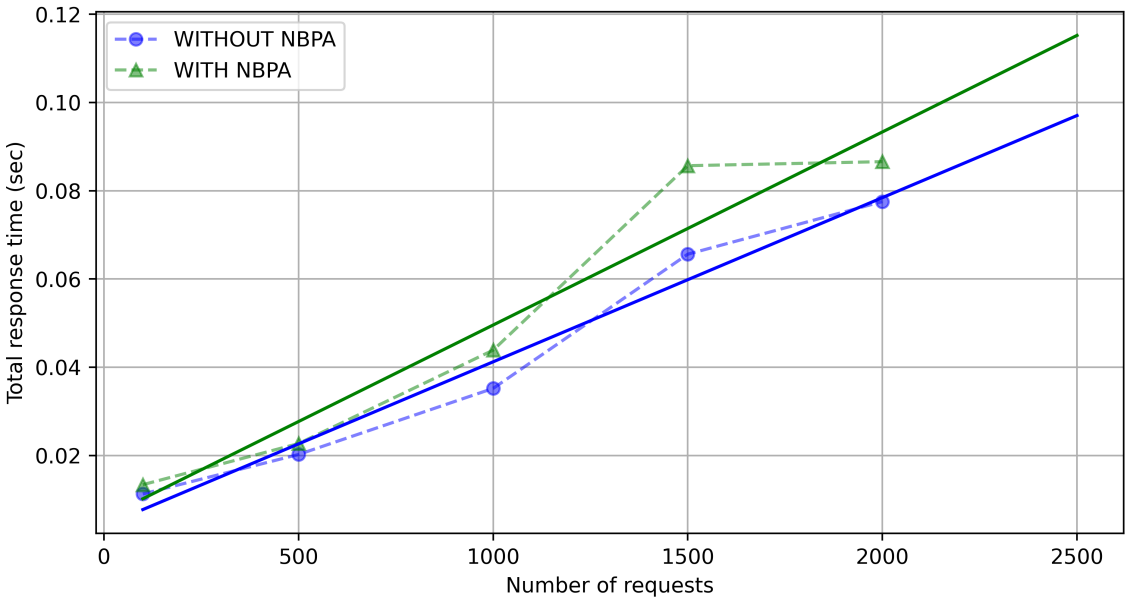

Figure 5.6 Response time against number of requests for the NBPA defense mechanism

The observed values, with and without NBPA, follow a linear tendency that we can extrapolate beyond the values of the graph. The loss of 14% in processing speed is relatively low for the added security.

Latency of HBRA Since the HBRA verification mechanism works in the background, it has no effect on the latency of read operations. However, additions and removals of entries to the database change the table hashes. In this situation, the hash-size parameter L impacts the latency of write operations.

The measurement of the latency of HBRA for 1000 write operations is shown in Fig. 5.7.

The observed values show that the throughput decreases with the hash size L. This decrease is caused by the larger input to the hashing functions, which now takes more time to digest.

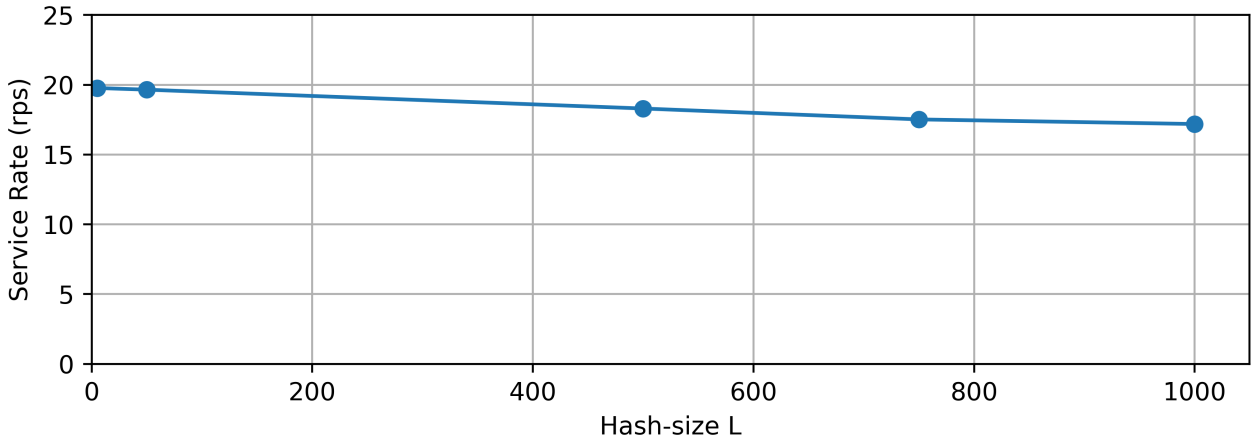

Figure 5.7 Service rate against hash-size L for 1000 requests for the HBRA defense mechanism

However, we note that the difference is very low compared to the gain in detection time.

General Latency To measure the latency induced by each defense mechanism, we send x simultaneous requests for $100 \leq x \leq 2000$ and note the execution time. We then calculate the function's slope using the Minimum Mean Square Error (MMSE) estimator [110]. The service rate μ, in requests per second (rps), is the inverse of the slope. The service rates in rps of our mechanisms are depicted in Fig. 5.8 for Write and Read operations. We also compute the relative difference with and without the mechanisms.

We notice that the HBFI defense mechanism is the greediest regarding service rate. Furthermore, the service rate decreases with the number of fields protected by HBFI. This phenomenon is expected since the more fields are protected, the more hashes are required to implement the defense. We also noticed that HBFA is faster than HBFI.

When all the defense mechanisms are combined, we observe a decrease of 73% for write operations. While these values suggest a drastic performance loss, it is essential to remember that most modern systems are qualified as "read-heavy". In Fig. 5.8, we notice that the read operations are much less affected by the defense mechanisms, with only a 43% decrease when all mechanisms are combined.

Memory Overhead Evaluation

The memory overhead evaluation shows the greediness of our mechanisms in resources.

To determine the impact of HBFI and HBFA on the memory overhead, we measure the memory usage of write and read operations with and without the defense mechanisms. The

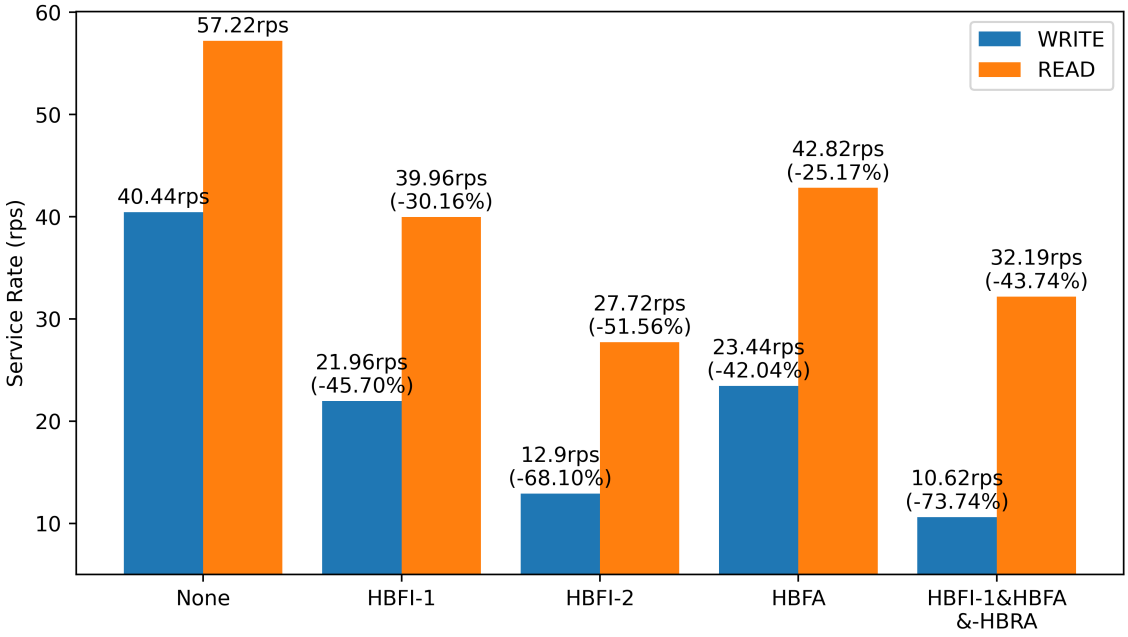

Figure 5.8 Average service rate HBFI, HBFA, and HBRA

obtained values are the average measurements over 100 requests and are depicted in Fig 5.9.

When both mechanisms are activated, we observe an increase of 49% for writing and 30% for reading. However, in absolute values, the increase is 14.19KiB and 4.16KiB, which remains very small and doesn't pose a memory overflow risk on our server.

For the HBRA defense mechanism, we question the impact of the hash size L on the allocated memory. Figure 5.10 depicts the allocated memory, in KiB, for 1500 consecutive creations on a database where $L = 1000$.

We immediately notice that the allocated memory depends on the number of lines in the database. However, after reaching the 1000 lines mark, the allocated memory drops from almost 1MiB to almost 0KiB. This is because, right before 1000 entries, the HBRA mechanism deals with a large and heavily populated hash. Whereas after 1000 entries, the large hash is full, and the HBRA mechanism now deals with a smaller and less populated hash. This indicates that a higher value of L means a much greater need for resources.

5.5.3 Discussion

The implementation and evaluation of our security architecture yielded positive results. Indeed, the four defense mechanisms (NBPA, HBFI, HBFA, and HBRA) significantly improve the detection of insider attacks with reasonable added delays.

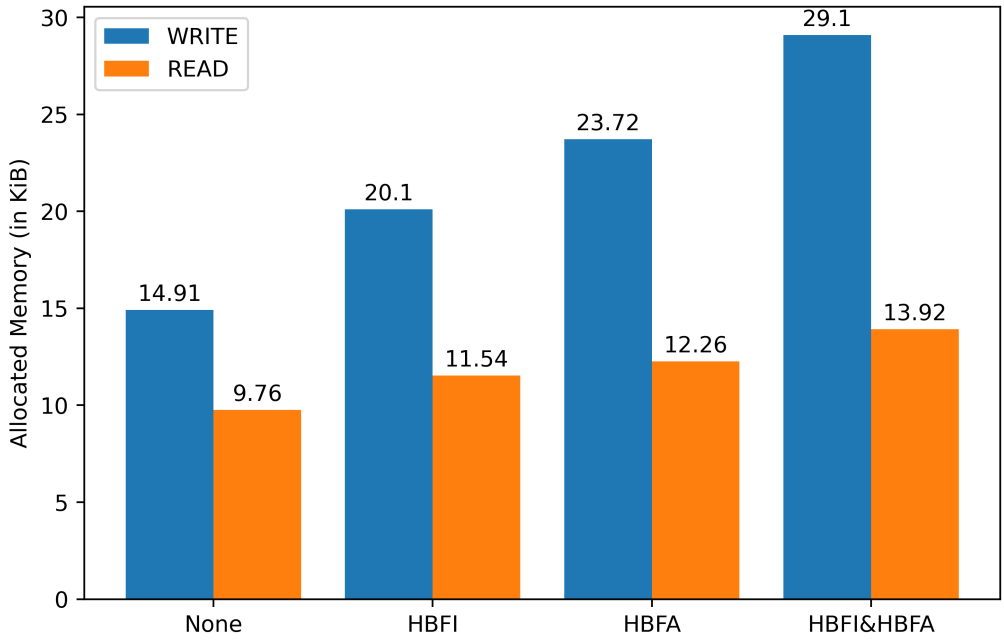

Figure 5.9 Allocated memory for read and write operations with HBFI and HBFA

Detection of an insider attack

We found that all scenarios of the insider attack model can be detected using the proposed defense mechanisms.

The NBPA mechanism detects all illegal access to the HSM, thus preventing attacks such as data access, data alteration, and private key replacement and theft. It is important to note that the goal of NBPA is to detect illegal access, not prevent it. Therefore, this mechanism should not halt the connection between the server and the HSM in the case of a replay attack but instead raise a high-level security alert. This is essential to maintaining the service's availability even after an attack has been detected.

The HBFI mechanism detects the swapping of sensitive fields between two entries, the HBFA mechanism detects the nullification of a field in an entry, and the HBRA mechanism detects the deletion of an entry in the database. It is worth noting that the HBFA mechanism also detects swapping and corruption attacks. However, the HBFI mechanism is more accurate. HBFA indicates something has been changed within the row, but HBFI indicates which field has been altered.

For NBPA, the detection occurs almost instantly for high-traffic applications since the nonces are constantly exchanged between the server and the HSM. For HBFI and HBFA, the detection occurs when the targeted data is queried from the database. For HBRA, the detection

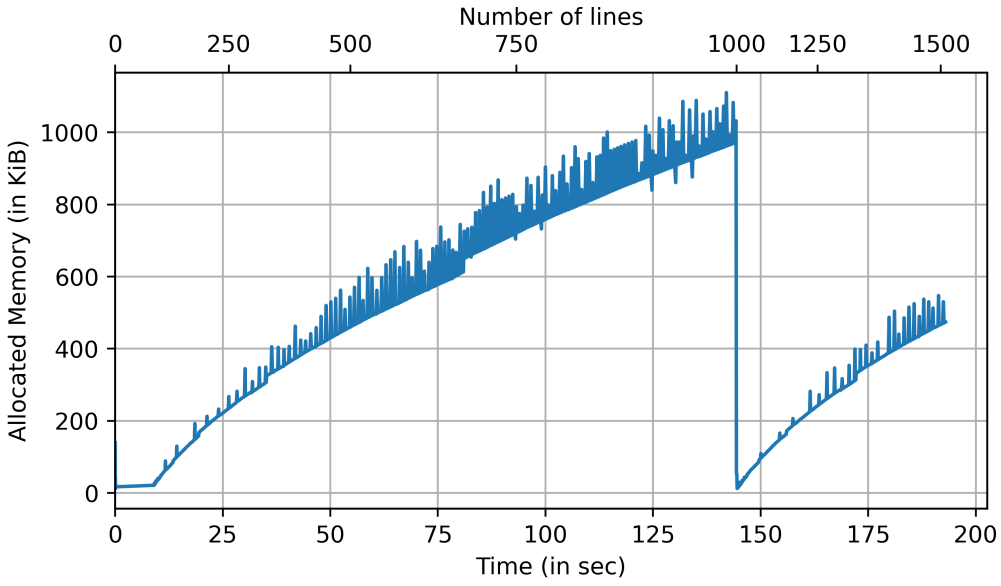

Figure 5.10 Allocated memory for write operations with HBRA when $L = 1000$

time depends on the parameters of the mechanism, such as the number of challenged hashes k and the size of the hashes L.

Security and performance trade-off

The performance evaluation showed that the induced latency is manageable for the proposed mechanisms. Indeed, the NBPA defense mechanism causes a modest throughput decrease of 14%. The HBFI mechanism causes a modest throughput decrease of 30% for one field and a more consequent 51% for two fields for read operations. The HBFA mechanism causes a throughput decrease of 25% for read operations. Most systems nowadays are "read-heavy" systems, meaning the volume of read operations is much higher than that of write operations. Therefore, even though the service rate decrease is higher for the write operations, we estimate that the number of write operations is low enough not to feel the difference.

Nevertheless, when performance is critical, there are a few possibilities to reduce the impact of our defense mechanisms. For example, instead of having many fields protected by HBFI, we could have the entire row protected by HBFA, reducing latency. Furthermore, the latency of the read operations is caused by the immediate validation of data upon request. Another possibility would be implementing a background random validation process similar to HBRA. Hence, a background process validates a given number of entries or fields instead of validating upon each request. We can achieve a low detection time by appropriately tweaking the parameters without impacting the application's performance.

Selection of the parameters for HBRA

The HBRA mechanism works in the background to periodically check for missing entries. Results show that a higher value for the number of challenged hashes (k) and the hash size (L) cause a drastic decrease in the time for a certified detection ($t_{\min}^{(\alpha)}$). However, the increase in L and k comes at a certain cost. Indeed, higher values of L cause a decrease in the service rate for the Write operations. Considering the decline in service rate caused by other mechanisms, it is best to avoid unnecessary losses.

Furthermore, higher values of L also impact the memory overhead of our application. Indeed, with low values for L, such as 25 entries per hash, we maintain a memory allocation of less than 100KiB, whereas $L = 1000$ causes the memory allocation to reach 1MiB. Intuitively, the same could be concluded for k. Since the number of challenged hashes increases, it is evident that the memory overhead rises as well. When observing the results of the minimal time for certified detection, we notice that in a table of 1000 entries, there is no need for k to go above 15 hashes per validation and L to go above 25 entries per hash. Indeed, the gain in detection time becomes negligible compared to the decrease in service rate and memory overhead.

5.6 Integration and Real-World Environments

The proposed security architecture is only one component of a larger system designed to mitigate security risks from different angles. In this section, we offer guidelines to integrate our solution in real-world environments.

5.6.1 Technical Deployment

Considering the airgapped nature of the HSM, this solution is designed to be implemented on the application server itself. The HSM is there to guarantee the secrecy of encryption keys and provide overall a safe key management mechanism. Therefore, we envision any implementation of this architecture to take the form of a framework or middleware that should be added on top of a regularly running application to handle the communication with the HSM.

The implementation and deployment of this framework must take into account the modularity of the architecture. We discussed in Section 5.5.3 the performance gain of choosing which defense mechanism to activate on a given table. This is achievable through the decorator pattern which allows the activation and superimposition of several defense mechanisms.

In the example of a backend server written in Python, where the communication with the database is handled through Object-Relational Mapping (ORM), Figure 5.11 shows an example to integrate the HBFI, HBFA, and HBRA defense mechanisms.

```
@EHBFI
@HBFA
@HBRA(app=app, hash_limit=200, hash_verifs=2, sleep_time=10)
class Seeding(BaseModel, IdModel):
    __tablename__ = 'seedings'
    crud_metadata = ['id', 'salary', 'name']
    id_field_name = 'id'

    salary = Column(EncryptedInt, nullable=False)
    name = Column(Unicode(255), nullable=False)
```

Figure 5.11 Deployment of the defense mechanisms on a sample class

In this figure, `Seeding` represents the python class mapped to the seedings table in the database, whereas `EHBFI`, `HBFA`, and `HBRA` are decorating functions which modify the behavior of the class by adding new columns, tables, and procedures for data handling.

On the HSM side, it is necessary to attempt to mitigate the performance loss and prevent bottlenecks during high-frequency periods. When conditions justify it and financial resources are available, it is possible to acquire several HSM instances and configure them in load-sharing mode in the PKCS #11 parameters. This will allow the environment to multiply the acceptable load by the number of HSM. It is also worth to note that depending on the needs and means, several classes of HSM exist which provide a different number of transactions per second.

5.6.2 Integration With Other Security Frameworks

The goal of our architecture focuses solely on the detection of insider attacks on data security. However, the handling of this detection is yet to discuss. Considering that all the defense mechanisms previously described are able to generate an alert, it is therefore possible to route those alerts to a dedicated security framework (i.e., SIEM, IDS, or SOAR) to more efficiently handle them.

Furthermore, our security architecture requires a preliminary intrusion to the system to conduct the attack. By routing the detailed alerts to the security framework, we can cross-

check the events related to this attack with other events on other nodes that would create a better understanding of the origin of the attack.

5.6.3 Deployment in Real-World Environments

The deployment of this proposed architecture in real-world settings can introduce some initial complications. However, we notice that these complications can be mitigated by thoroughly examining the needs:

Data Sensitivity vs Performance

The modularity of our architecture allows an easy management of the trade-off between data sensitivity and performance. In corporate environments, where stored data often relate to sensitive but not critical information, we can justify the deployment of only the most time-efficient defense mechanisms. However, financial or governmental institution handle critical information and will require to opt for the most secure defense mechanisms, even at the risk of a performance loss.

Compliance with security policies

Several compliance standards exist to supervise the secure handling of information. For example, corporate environments tend to follow consumer-oriented standards such as GDPR and HIPAA. Government environments will go towards stricter standards such as FIPS, and financial institutions follow banking-related standards like PCI-DSS. Integrating our solution to any of those environment first requires an analysis of the required standards and an understanding of the level of security it implies. Afterwards, the proper configuration of the HSM in our architecture can help achieving the requirements of the standard. Considering that our security architecture delegates all cryptographic operations to the HSM, the compliance of the HSM to a standard is automatically extended to the architecture.

5.7 Conclusion

In this paper, we proposed an HSM-based architecture to detect insider attacks on server-side data. Our approach focuses on the detection phase, whereas other related works concentrate more on the response and recovery phases. Moreover, our proposed architecture combines four new defense mechanisms: Nonce-Based Process Authentication, Hash-Based Field Integrity, Hash-Based Field Availability, and Hash-Based Row Availability.

The results show that combining these defense mechanisms covers the baseline architecture's entire *insider attack model*. In other words, the proposed architecture guarantees the detection of all the attacks presented in the model. Furthermore, the throughput decrease is manageable for most modules (14% for NBPA, 30-50% for HBFI in Read, 25% for HBFA in Read, and 43.74% for the combination of all mechanisms in Read). The HBRA module works in the background to achieve a quick detection of a deletion attack (5 minutes in a table of 1000 entries). Our work has shown that this restructuring of data storage can detect confidentiality, integrity, and availability breaches while maintaining a reasonable runtime and memory overhead.

Even with the added security and reasonable latency, we observe a total of 43.74% runtime overhead, which could be problematic for real-time applications. It is necessary to continue working on alternative optimizations to reduce the latency further. Moreover, this architecture is built on the premises of an existing insider attack model. With the constant evolution of insider threats, new attack models might emerge and will require the adaptation of this security architecture. Also, this proposed HSM-based architecture is tailored for centralized systems and does not account for the vulnerabilities of a distributed system.

In future work, we should extend data security (i.e., confidentiality, integrity, and availability) to decentralized systems, such as blockchain, where trust management differs significantly from centralized systems. We can also integrate this work with existing machine learning detection methods and existing security frameworks to cover a broader range of attacks. Furthermore, we can also use machine learning algorithms to improve the adaptability of our architecture against evolving insider attacks.

Acknowledgment

The authors would like to thank Dr. Franjieh El Khoury for her constructive comments and for proofreading this paper.

The authors would also like to thank Entrust Corporation for providing useful insights and documentation about HSM.

Tableau 5.5 Comparison of cryptographic insider attack detection methods

Method	**Conf.**	**Integ.**	**Avail.**	**SCA-Res.**	**User Input**	**TCB**
Ours	✓	✓	✓	✓		HSM, HSM CodeSafe
Database Encryption [84, 90, 98–100]	✓[a]					
EnclaveDB: A secure database using SGX [134]	✓	✓	✓			Hekaton engine, Trusted Kernel, In-Memory Storage, Transaction Manager, Pre-compiled procedures
The implementation of negative database as a security technique on a generic database system [97]	✓	✓			X	
Study on methods for data confidentiality and data integrity in relational database [101]		✓				

[a] Not guaranteed against insider attacks

CHAPITRE 6 ARTICLE 3 : BLOCKDBMS - A SECURE DISTRIBUTED DATABASE MANAGEMENT SYSTEM BASED ON BLOCKCHAIN AND HSM

Marc Dib, Samuel Pierre (Senior Member, IEEE)

Mobile Computing and Networking Research Laboratory (LARIM), Department of Computer and Software Engineering, Polytechnique Montréal, Montreal, QC H3T 1J4, Canada

E-mail : marc.dib@polymtl.ca ; samuel.pierre@polymtl.ca

Revue : Soumis pour publication le 23 Octobre 2024 à IEEE Transactions on Information Forensics and Security.

ABSTRACT

Distributed database is a growing solution to large-scale data storage. However, it suffers from concurrency, confidentiality, integrity, and availability issues. Blockchain technology helps solve some of those issues, but confidentiality remains challenging. In this paper, we proposed BlockDBMS, a secure Distributed Database Management System based on blockchain. This private blockchain uses the Hardware Security Module to ensure confidentiality and key management in the database. The implementation of our solution shows a significant improvement in data security while also providing a scalable system. Indeed, with our physical setup, we obtained a system where the execution time follows a large-step tendency with 200 requests per block, which indicates a possibility for a scalable system.

INDEX TERMS

Blockchain, confidentiality, cryptography, data security, encryption, hardware security module, integrity

6.1 Introduction

Data storage, processing, and protection is an increasingly important problem. Indeed, today's applications rely heavily on data, such as user data for personal applications and sensor data for Internet of Things (IoT) applications. In 2023, we accounted for approximately 30

billion connected devices. 25% of those devices are Smartphones, and 50% of those devices are IoT devices [141, 142]. The amount of data needing to be processed and stored in databases is greater than ever and keeps growing. Therefore, a cyber system must be more scalable, resilient, flexible, and secure to handle growing data.

One solution that provides some of these features is a distributed database (DDB). A distributed relational database is a set of tables spread across interconnected computer systems [143]. Indeed, through fragmentation and replication, DDBs increase the reliability and efficiency of Database Management Systems (DBMS) [144]. However, we still face many challenges related to DDBs. In DDBs, data replication causes concurrency issues, which can happen when two contradictory modifications happen simultaneously on the same portion of data [70]. Moreover, data consistency can be affected if one local instance of the distributed database is attacked. In this situation, the result of a query might vary depending on the instance which replies the fastest to the query.

Using blockchains as a distributed database is a potential solution to the concurrency, integrity, and availability issues [103, 104]. Indeed, blockchains inherently guarantee integrity and availability through their decentralized aspect and immutability [103, 104]. However, it is known that blockchain does not inherently offer confidentiality in its paradigm. Therefore, sharing sensitive queries and results on the blockchain creates potentiel data leaks that can be exploited by malicious nodes in the network. The concept of On-Chain Encryption (OCE), while interesting, comes with its own set of challenges especially related to the management of encryption keys. In addition, choosing the right consensus algorithms could provide a solution for the concurrency issue and allow better performance. For example, the Proof of Work consensus algorithm is notorious for being greedy in time and resources, making the blockchain slow to update and consuming too much energy [145].

To summarize, distributed databases face confidentiality, concurrency, and reliability issues. Is it obvious that simply offloading the database on the blockchain does not solve all problems because of storage and confidentiality issues, especially regarding the access and revocation of encryption keys.

In this paper, we aim to solve this issues by addressing the need for a coherent data management scheme in DDBs, coupled with a data validation scheme for consistency among the DDB instances. Moreover, we aim to address the confidentiality issue with regards to key management and efficiency of encryption.

Hence, we propose BlockDBMS, a secure blockchain-based distributed database which utilizes hardware security to achieve a high level of robustness. The main contributions of this paper are as follows:

— We propose a Distributed Database Management System (DDBMS) based on smart contracts to efficiently and seemlessly manage the storage, access, and modification of data among database instances;
— We introduce the novel Hardware Security Module Based Security Node (HSM-BSN) to the blockchain, responsible for data security. The addition of this node brings a critical improvement to key management and the limitations of OCE;
— We propose a lightweight and efficient two-part authentication protocol for new HSM-BSNs joining the network;
— We perform a qualitative and quantitative evaluation to show the security and efficiency of BlockDBMS.

The originality of this paper relies on our ability to efficiently restructure the communication between the different actors in a distributed database to achieve a better security level. Indeed, while the blockchain and HSM seem two be two disjointed paradigms, we obtained a robust interoperation mechanism that draws from the strength of each paradigm to improve data security without hindering the regular data access processes.

The rest of this paper is organized as follows. Section 6.2 details the technical background knowledge. Section 6.3 presents a critical review of the distributed databases protection methods and blockchain confidentiality approaches. Section 6.4 describes the proposed DDB architecture with its authentication protocols. A discussion of the implementation and the obtained results is presented in Section 6.5. Finally, Section 6.6 concludes the paper.

6.2 Background

In this ection, we define the necessary background on distributed databases, blockchain technology, and the Burrows, Abadi, and Needham (BAN) logic of authentication.

6.2.1 Distributed Database

In large cyber systems, a single localized database might not suffice to store the entire application data. In other cases, data availability is so critical that we require a redundancy of information that a localized database cannot provide. Instead, we use distributed databases to fragment or replicate data across several servers [143]. Distributed databases are also helpful in creating multiple geographical access points to the information, thus reducing the communication costs for data transfer [146].

Fragmentation means spreading data across the servers so that each piece of information is available on one server only [70]. Splitting data would allow better resource management.

However, the lack of redundancy constitutes a vulnerability against data availability should one of these servers fail.

Replication means copying data across the servers so each piece of information is available on more than one server [70]. Copying data would imply a larger storage space but ensure all data remains available if one server fails. Full replication means that the same data is available on all servers.

Some known frameworks for distributed databases include Apache Ignite [147], Couchbase Server [148], and Amazon SimpleDB [149].

However, using a distributed database implies that many database files are dispersed in a network. These files must remain synchronized to maintain data accuracy during user queries [146].

6.2.2 Blockchain Technology

A blockchain is a decentralized ledger to register transactions on different nodes [57, 61]. A blockchain is typically immutable, resistant to tampering and replay attacks, verifiable, and resilient [57, 61]. The Blockchain 1.0 architecture represents cryptocurrencies (i.e., Bitcoin) [57], the Blockchain 2.0 architecture introduced Smart Contracts [61], and the Blockchain 3.0 architecture deals with various application domains such as distributed databases, videogames, IoT, and supply chains [57]. Some known frameworks for blockchains are the Bitcoin Blockchain [150], Ethereum [151], and HyperLedger Fabric [152].

General Description

The blockchain paradigm can be described in four layers [61]:

— The *network layer* for the communication between nodes;
— The *consensus layer* for the protocols and algorithms to reach consensus between nodes;
— The *Replicated State Machine (RSM) layer* for the interpretation and recording of transactions;
— The *application layer* for high-level operations.

Additionally, a blockchain can either be public, semi-private, or private [57]. A public blockchain has no access restriction across the internet. In contrast, a private blockchain is restricted to internal use inside an organization. Semi-private blockchains are restricted to shared use among several organizations.

Consensus Algorithms

To maintain the unity of the blockchain, all nodes must agree when adding a block to the chain. Hence, each blockchain implements a consensus algorithm to reach an agreement. We define hereby the *Proof of Work* and the *Proof of Authority* consensus algorithms.

Proof of Work Proof of Work (PoW) is among the most popular consensus algorithms. A node must solve a complex cryptographic riddle to propose a new block to the chain [153]. This riddle is challenging to solve but easy to verify so the other nodes can quickly validate a proposed solution [153]. However, the PoW is greedy for resources, which causes high power consumption by the mining nodes. Currently, the Bitcoin network has an alarmingly high power consumption [154] and environmental impact [145].

Proof of Authority The Proof of Authority (PoA) is more adapted for private and semi-private blockchains. In the PoA, a limited number of trustworthy nodes act as *Authorized Signers* [153]. Only these nodes are allowed to create blocks from the pool of transactions and add them to the blockchain. Moreover, the Authorized Signers can vote to add a new signer or revoke an existing one. An example of a PoA algorithm is the *Clique* protocol defined for the Ethereum blockchain [155].

In the case of a private blockchain, a PoA algorithm should be more suited than a PoW algorithm to avoid the power consumption issue, as well as to increase the speed of the blockchain [156].

Smart Contracts

A Smart Contract is an addition to the blockchain paradigm to execute a program on the blockchain [157]. Through Smart Contracts, we can achieve a trustworthy execution without fear of tampering. Smart Contracts are deployed on the blockchain and are available to anyone knowing the address of this contract. The contract encompasses functions that the node can call to read information stored in the contract. Moreover, a node can transact with some contracts to change the state of variables inside the contract. Smart Contracts are implemented in the Ethereum blockchain in the Solidity language [158].

To maintain trust and integrity of information, the contract is isolated from the worldwide internet and only deals with on-chain sources of information (i.e., a node or another contract). However, some contracts need real-world information and use Oracles to fetch data from off-chain sources [159]. The implementation of the Oracle should nevertheless maintain the trust

and integrity aspect of the blockchain.

6.2.3 BAN Logic Evaluation

The Burrows, Abadi, and Needham (BAN) logic of authentication is a methodology to validate and analyze security protocols [160]. The BAN logic is based on the idealization of protocols using the specific symbols shown in Table 6.1.

Tableau 6.1 Symbols of the BAN Logic

Symbol	Description
$A \mid\equiv X$	A believes that X is true
$A \triangleleft M$	A received a message containing M
$A \mid\sim K$	A once used the key K to encrypt a message
$A \mid\sim M$	A once sent a message containing M
$A \Rightarrow M$	A has control over M and can be trusted on M
$A \overset{K}{\leftrightarrow} B$	A and B can use the secret key K to communicate
$\#(M)$	M has not been sent before in any run of the protocol (fresh)
$\overset{K}{\rightarrow} A$	A has a public key K and a private key K^{-1}
$\{M\}_K$	M is encrypted using the key K

Moreover, the BAN logic presents three rules to help evaluate protocols: the *message meaning* rule, the *nonce verification* rule, and the *jurisdiction* rule.

Message Meaning

if A believes in a secret key K between A and B and A sees a message M encrypted with K, then A believes that B sent M, as illustrated in (6.1).

$$\frac{A \mid\equiv \left(A \overset{K}{\leftrightarrow} B\right), A \triangleleft \{M\}_K}{A \mid\equiv (B \mid\sim M)} \tag{6.1}$$

Nonce Verification

if A believes N is fresh and A believes B once said N, then A believes that B believes N, as illustrated in (6.2).

$$\frac{A \mid\equiv (\#(N)), A \mid\equiv (B \mid\sim N)}{A \mid\equiv (B \mid\equiv N)} \tag{6.2}$$

Jurisdiction

if A believes B has control over X and A believes B believes X, then A believes X, as illustrated in (6.3).

$$\frac{A \mid\equiv (B \Rightarrow X), A \mid\equiv (B \mid\equiv X)}{A \mid\equiv X} \tag{6.3}$$

6.3 Related Work

In our previous work, we showed the efficiency of a cryptographic architecture to protect data against outsider attacks [85] and insider attacks [109,161]. However, this approach is specific to localized databases and the challenges exposed in Section 6.1 are still to be addressed.

Therefore, we review in this section the existing work related to distributed database protection, distributed databases based on blockchain, and blockchain security.

6.3.1 Distributed Database Protection

Standard symmetric encryption algorithms, such as the Advanced Encryption Standard (AES) algorithm [23], can be used in distributed databases to ensure confidentiality. This approach would require either a common key or individual keys.

The complexity of having individual keys is highlighted in [162] since there will be an unmanageable amount of keys to maintain, store, revoke, and decommission in case of a breach. Moreover, the management of those keys is therefore left to the nodes which can end up compromised or malicious.

The possibility of having a preshared key for all nodes allows better node communication and reduces the time and effort required in backup-and-restore operations, key revocation, and overall key management [67]. Indeed, if a new database node joins the network or one node needs to reset, any node can send its data files to this new node. The new node does not need to decrypt and re-encrypt data because they already share the same key.

In both scenarios, the concept of acccess revocation needs to be studied thoroughly to prevent a node from further accessing data after it has been marked as malicious or compromised.

When dealing with individual or pre-shared keys, it's sometimes impossible to make sure the node effectively surrendered its keys, which poses a leak risk.

6.3.2 Distributed Database Based on Blockchain

Some approaches in literature use blockchain technology to answer the integrity and concurrency issues of distributed databases. Indeed, a Blockchain-Based Distributed Database leverages the idea of a consensus to maintain order and consistency in the database.

Nathan *et al.* propose a blockchain relational database in [103]. Their approach consists of replicating database nodes as blockchain nodes, for which blockchain consensus will determine the transactions. This approach improves the synchronization process by delegating it to the blockchain paradigm, thus solving the concurrency issue. However, the consensus algorithm is a critical component in establishing an efficient system, as some consensus algorithms have very poor scalability performance.

Similarly, McConaghy *et al.* propose in [104] another blockchain-based database: BigchainDB. Once again, this approach combines the consensus processes of a blockchain to manage a distributed database. However, the authors use a consensus algorithm based on Raft [106], which could be costly in terms of network traffic.

Both approaches lack a harmonious encryption mechanism that would allow the nodes to communicate effectively using encrypted data.

6.3.3 Blockchain Security

When we consider blockchain as a storage solution, it is essential to ensure the security of the information stocked. For instance, if some sensitive information is stored on the blockchain, we need to ensure its confidentiality, which is not inherent to the blockchain.

A commonly used method is "On-Chain Encryption" [67], which is simple in principle: the confidential information is encrypted using a symmetric encryption key, then the digest is uploaded on the blockchain via a smart contract. However, in the application, this encryption key must be shared with whoever requires access to the information. This brings up the problem of a key-sharing mechanism. In addition, the problem of access revocation should also be noted since we have no way of forcing a node to surrender its access to the information.

On the network access layer, typical network security methods, such as DNSSEC, VPNs, and TFA, can be used to authenticate new nodes joining the network [61]. However, the authors note that these protocols can induce overhead and reduce the dynamism of the blockchain,

which is not ideal.

Finally, smart contracts can detect insider attacks on the blockchain [74]. However, collaborating nodes can circumvent the proposed protocol in [74] and conduct the attack. Moreover, without specialized hardware, the authentication protocol, relying on cryptography, is vulnerable to side-channel attacks [49].

6.4 Proposed Architecture

The proposed architecture for BlockDBMS combines two smart contracts and introduces an HSM-Based Security Node (HSM-BSN) with its Distributed Database Management System (DDBMS) Middleware. The general architecture is described in Fig. 6.1.

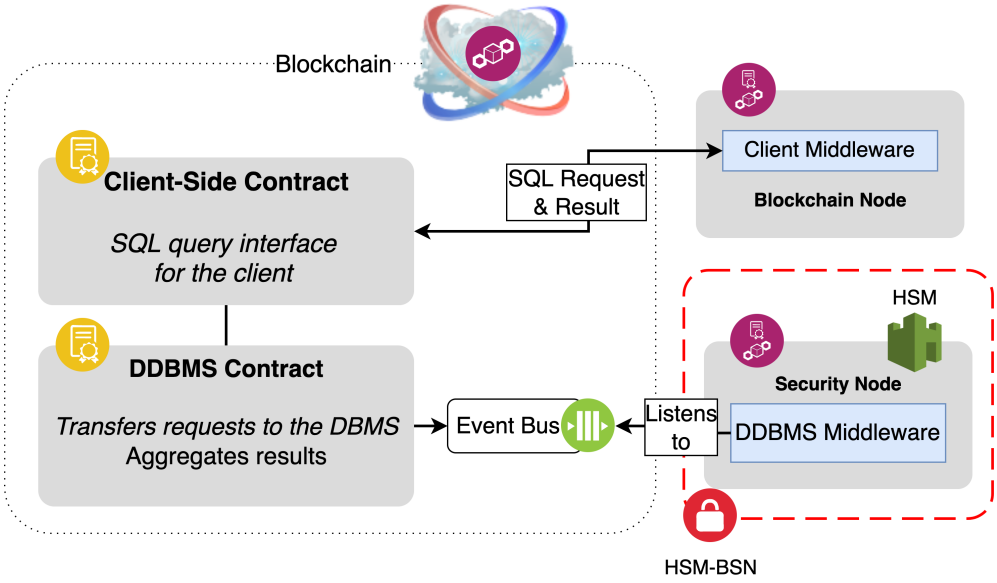

Figure 6.1 General Architecture for BlockDBMS

This architecture is tailored for private and semi-private blockchains. The nodes joining this blockchain must belong to the concerned organizations to be authorized to join. For example, the blockchain runs on a private network on which the nodes need to be to join the blockchain. Traditional authentication mechanisms can be used here to verify the identity of joining nodes.

To improve the efficiency of the blockchain, we choose a Proof-of-Authority (PoA) consensus algorithm. This PoA algorithm, as described in Section 6.2.2, assigns the right to add new blocks to the chain to a limited number of nodes. This algorithm prevents malicious nodes from adding malicious blocks to the chain. In our architecture, we decide that each

HSM-Based Security Node (Section 6.4.1) is an Authorized Signer. Moreover, we create an HSM-BSN Authentication protocol for any node that wishes to be promoted to Authorized Signer (Section 6.4.2). This protocol proves, without divulging the Application Key (AK), that the concerned node is indeed connected to an HSM with the same AK as the others. The Client-Side Contract (Section 6.4.3) serves as an interface for the client to access the database. The DDBMS Smart Contract (Section 6.4.4) handles the secure data acquisition using the DDBMS Middleware.

6.4.1 HSM-Based Security Node

Only the HSM-Based Security Nodes (HSM-BSN) can be Authorized Signers for the PoA consensus algorithm in our architecture. An HSM-BSN is a combination of a regular blockchain node connected directly to an HSM and the database. This allows the node to access the HSM's cryptographic functionalities and act as a gateway between the blockchain and the database.

The HSM-BSN runs the DDBMS Middleware, which securely treats queries in a decentralized database. The middleware uses an Object Relational Mapping (ORM) library to map programming classes to database structures. This middleware first needs to be initialized by creating the correct database structure. When a new table is created, the JSON skeleton of the database is encrypted using the AK of the HSM and uploaded on the blockchain. In other cases, the JSON skeleton is fetched from the blockchain to initialize the ORM classes. This process is described in Algorithm 1.

Algorithm 1 DDBMS Middleware Initialization

Require: CreateTable {Boolean Value}

```
1:
2: if CreateTable is TRUE then
3:     skeleton ← MakeDatabaseSkeleton()
4:     skeleton ← HSMEncrypt(skeleton)
5:     Transact("SetSkeleton", skeleton)
6: else
7:     skeleton ← Call("GetSkeleton")
8:     skeleton ← HSMDecrypt(skeleton)
9:     SetORMClasses(skeleton)
10: end if
```

In Algorithm 1, we define the following methods:

— *MakeDatabaseSkeleton* creates a JSON structure containing all the parameters to recreate the existing tables for other nodes. Each entry contains the name of the ORM class,

the name of the table, and the name of the columns with their types and optional parameters (i.e., Primary Key, nullable);

— *HSMEncrypt* uses the HSM to symmetrically encrypt a payload with the AK;

— *HSMDecrypt* uses the HSM to decrypt a symmetrically encrypted payload with the AK;

— *Transact* transacts with a Smart Contract function. The first argument is the name of the function, and the other arguments are the parameters of this function;

— *Call* calls a Smart Contract function. The first argument is the name of the function, and the other arguments are the parameters of this function;

— *SetORMClasses* creates the ORM Classes from the skeleton and the corresponding tables in the database.

Moreover, the middleware is programmed to listen to Request Notification events on the blockchain event bus. When the DDBMS Smart Contract emits such an event, the middleware runs the callback method to process the event as described in Algorithm 2.

Algorithm 2 DDBMS Middleware Callback Method

Require: RequestId {The ID of this request}
Require: PrivateKey {The Private Key of this node}
Require: Query {The SQL query}
1: *JSONQuery* ← PreprocessQuery(*Query*)
2:
3: **if** *JSONQuery.method* is "SELECT" **then**
4: *result* = Select(*JSONQuery*)
5: Transact("SetResult", *result*, *RequestID*)
6: **else if** *JSONQuery.method* is "INSERT" **then**
7: Insert(*JSONQuery*)
8: **else if** *JSONQuery.method* is "UPDATE" **then**
9: Update(*JSONQuery*)
10: **else if** *JSONQuery.method* is "DELETE" **then**
11: Delete(*JSONQuery*)
12: **end if**

In this algorithm, we define the following:

— *PreprocessQuery* parses a regular SQL query (i.e. `SELECT id, name, salary FROM seedings WHERE name="John"`) into a JSON version of the query. The result for this query is illustrated in Fig. 6.2. This JSON query is used to construct the ORM query.

— *Select* uses the JSON query to select the relevant information from the database according to the given query;

— *Insert* uses the JSON query to create new entries in the database;

— *Update* uses the JSON query to update entries in the database;

— *Delete* uses the JSON query to delete entries from the database.

```
{
  "method": "SELECT",
  "columns": ["id", "name", "salary"],
  "table": "seedings",
  "where": [
    "name='John'"
  ]
}
```

Figure 6.2 JSON Transformation of the query `SELECT id, name salary FROM seedings WHERE name="John"`

During the creation of the ORM Classes in Algorithm 1, we flag the sensitive database fields to encrypt them. In our previous research work [85], we implemented the Enhanced StealthDB (ESDB), which allowed a DBMS to encrypt sensitive data in a database while maintaining the regular SQL features. In this work, we reuse the ESDB to encrypt sensitive data in our distributed database. Hence, the Select, Insert, Update, and Delete functions use ESDB to process sensitive values securely.

6.4.2 HSM-BSN Authentication Protocol

Since all HSM-BSN must act as Authorized Signers in the blockchain, we require an authentication protocol for any new HSM-BSN joining the network. Without loss of realism, we suppose the PoA-based blockchain is equipped with the following:

— A voting protocol allowing the Authorized Signers to vote to add a new signer or to revoke an existing signer;

— A function to get the blockchain addresses of all the Authorized Signers.

In our context, a node qualifies as HSM-BSN if it is connected to an HSM sharing the same Application Key (AK) as the others in the network. Therefore, our protocol must be able to prove that property for the challenged node. The HSM-BSN Authentication Protocol is illustrated in Fig. 6.3.

The protocol steps are as follows:

1. The new node (requestor) sends an Authentication request on a designated TCP port of the challenger node. The request is `AuthRequest:{addr}` where `{addr}` is replaced by the blockchain address of the requestor node;

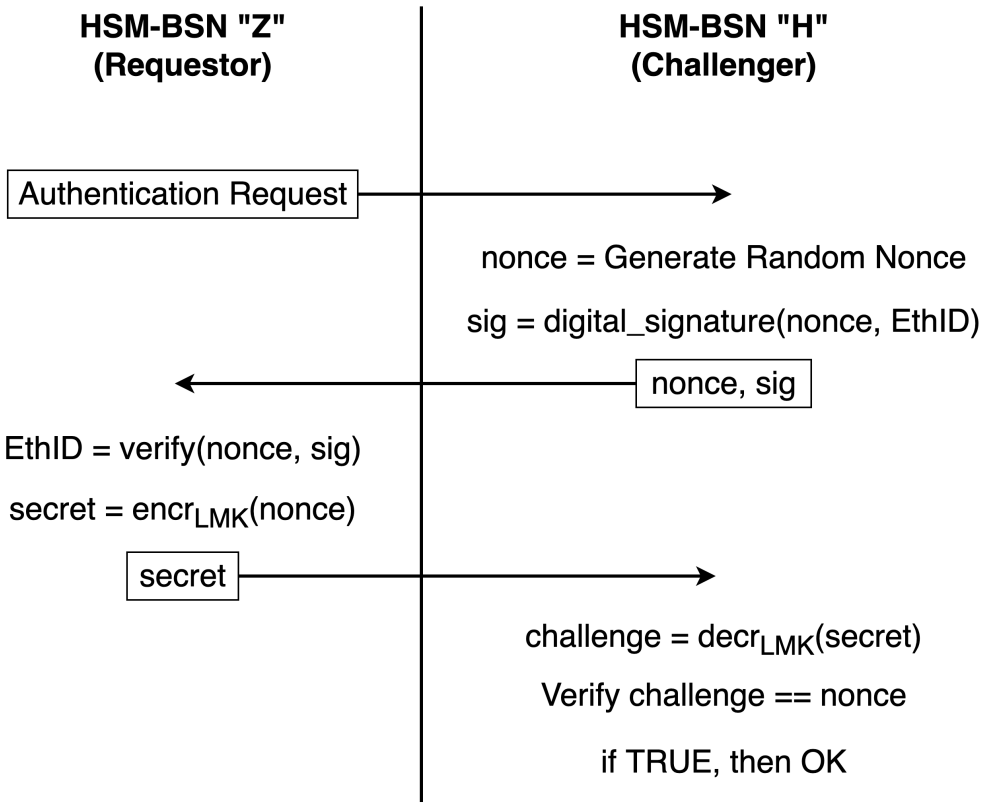

Figure 6.3 HSM-BSN Authentication Protocol

2. The challenger node generates a random nonce;
3. The challenger signs the nonce with its private key;
4. The challenger sends the nonce and its signature to the requestor;
5. The requestor uses the nonce and the signature to verify the identity of the challenger;
6. The requestor encrypts the nonce using the AK;
7. The requestor sends the encrypted nonce to the challenger;
8. The challenger decrypts the encrypted nonce and compares it to the original value;
9. If the nonce passes the validation, the challenger accepts the requestor and votes favorably.

Step #1 of this protocol assumes that the requestor knows the IP address of the challenger to send the Authentication Request. However, in real cases, the PoA-based blockchain can only provide the blockchain address of the nodes, not their IP addresses. Hence, we propose an HSM-BSN Identification Protocol, which identifies the IP Addresses of existing HSM-BSNs. The HSM-BSN Identification Protocol is illustrated in Fig. 6.4.

The protocol steps are as follows:

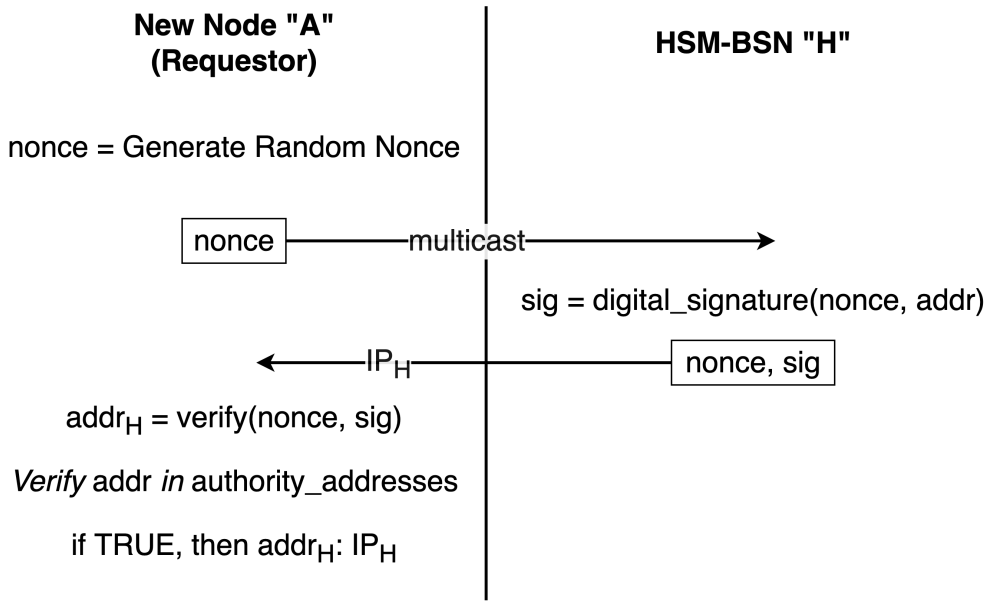

Figure 6.4 HSM-BSN Identification Protocol

1. The new node (requestor) generates a random nonce;
2. The requestor sends the nonce through a multicast packet to the entire blockchain network;
3. Each HSM-BSN receiving the request signs the nonce with their private key;
4. The HSM-BSN returns the nonce, its public IP address, and its signature to the requestor;
5. For each response, the requestor obtains the address of the responder from the signature;
6. The requestor verifies if the address belongs to the list of Authority nodes from the blockchain;
7. The requestor also verifies that the recorded IP address (step 5) matches the IP address sent by the responder;
8. If the addresses are verified, the requestor associates the IP address to the verified node.

6.4.3 Client-Side Contract

The Client-Side Smart Contract is the interface through which the client sends database-related operations on the blockchain and receives its results. This contract is accessible to all nodes and never contains sensitive information in plain text. There are four essential features in this contract: the *execution* of an SQL query, the *reading* of a decrypted result, the *emission* of a notification, and the *clearing* of the secure buffer. Some of these features

create access to the DDBMS Smart Contract. When deploying the Client-Side Contract, the address of the DDBMS Contract should be hard-coded instead of programmed in a variable to make access to the contract harder.

Execution of an SQL query

This feature connects to the DDBMS Contract and forwards the SQL query. The SQL query is asymmetrically encrypted for additional security using the HSM public key.

Reading of a decrypted result

After a query, the results should be read from the secure buffer. This feature connects to the DDBMS Contract to fetch the values from the buffer.

Emission of a notification

The client nodes wait after sending a query to receive a notification when the query has been properly executed. Therefore, when a query is completed, the contract emits a notification on the event bus. While these notifications are public and accessible to all, we specify the concerned node address in the event properties. This node can then use the reading function to access the result of its query.

Clearing of the secure buffer

After reading a decrypted value, this feature connects to the DDBMS Contract to clear the value from the secure buffer.

6.4.4 Distributed Database Management System Smart Contract

The DDBMS Smart Contract is reserved for the HSM-BSN to use via the DDBMS Middleware. The functions of this contract require identification before being executed so only HSM-BSNs can call or transact with the contract. In this contract, a list of requests keeps track of the different requests, their owners, and their answers. Moreover, a buffer contains the results of the requests to be read by the client. In this section, we denote by b the number of HSM-BSNs in our blockchain.

Execute an SQL query

The Client-Side Contract calls this feature and contains the SQL query its owner. It creates a new request with the query and its owner and sends a Request Notification on the event bus. This notification will trigger the callback of the DDBMS Middleware (Section 6.4.1).

Set a result

Since the database is distributed, a fault or an attack might modify one database instance. The DDBMS Middleware calls this function to record the answer of each HSM-BSN. When the number of answers reaches $\left\lceil \frac{b}{2} + 1 \right\rceil$ of the number of existing HSM-BSN, we start counting the answers. If an answer has been given $\left\lceil \frac{b}{2} + 1 \right\rceil$ times, we accept this as the majority answer and save it in the buffer. The client is then notified that his request is finalized.

Fetch from the buffer

This feature can only be called by the Client-Side Contract. It fetches a result from the buffer according to its owner. This allows the result to be accessible only to the node that ordered the query.

Clear from the buffer

After fetching from the buffer, the Client-Side Contract transacts with the DDBMS Contract to delete the value from the buffer. This prevents a malicious entity from picking data residue from the current state of the contract.

Setting and getting the database skeleton and the public key

The HSM-BSN also has access to contract functions, allowing it to set or get the encrypted database skeleton for the ORM setup and the HSM public key. The client uses this public key to encrypt the query.

6.5 Results and Discussion

6.5.1 Implementation

To facilitate the testing and evaluation phase of our work, we implemented the proposed architecture using a simulated network of nodes. Using the Docker technology [123], we

created a virtual private network on a host machine. On this private network, a predefined number of nodes is created, and the nodes are divided into two categories: regular blockchain nodes and HSM-BSNs.

All nodes are equipped with the Go-Ethererum Client, which allows access to a private blockchain and the necessary middleware source codes to access features of the proposed architecture. In addition, the HSM-BSN is equipped with an HSM-like source code to simulate the cryptographic abilities of an HSM and a PostgreSQL instance to manage the local database instance.

It is critical to note that the recorded performances in Section 6.5.3 are affected by the sharing of resources during the simulation. To the least, a single simulation comprises 3 HSM-BSN and 1 regular node, which equates to 4 blockchain clients (3 miners and 1 light client), 3 HSMs, 3 PostgreSQL databases, 3 HSM-BSN Middlewares, and 1 Client Middleware.

6.5.2 Results

The evaluation of this work is separated into two distinct categories: *qualitative* results and *quantitative* results. The qualitative results aim to assert the good functioning of our proposed architecture and its security properties. The quantitative results determine the performance of the implemented architecture for standard metrics such as network occupancy, latency, and throughput.

Qualitative Results

The qualitative results of this work comprise of the following: we formally evaluate the HSM-BSN Identification and Authentication Protocol (IAP) using the BAN Logic [160], we analyze the security properties of the HSM-BSN IAP, and we analyze the security properties of BlockDBMS.

HSM-BSN Identification Protocol BAN Logic Evaluation The HSM-BSN Identification Protocol pings and verifies an existing HSM-BSN to obtain its IP address. The authentication has to be challenged by a regular node that knows the blockchain address of the HSM-BSNs and can verify their identity using the signature verification process.

The protocol consists of 2 messages:

$$(M_1)\ A \rightarrow \text{all} : N_A$$
$$(M_2)\ H \rightarrow A : \left\{N_A, \xrightarrow{K} H\right\}_{K^{-1}}$$

The goal of this protocol is to prove to A that H is the one who replied to the request and that H used the private key K^{-1} to sign the nonce, as shown in G_1 and G_2.

$$(G_1)\ A \mid\equiv (H \mid\sim N_A)$$
$$(G_2)\ A \mid\equiv \left(H \mid\sim K^{-1}\right)$$

In this protocol, we make the two assumptions that A believes that H believes in his own public/private key and that A believes in the freshness of its own nonce, as shown in A_1 and A_2.

$$(A_1)\ A \mid\equiv \left(H \mid\equiv \overset{K}{\rightarrow} H\right)$$
$$(A_2)\ A \mid\equiv (\#\,(N_A))$$

With these assumptions, we can emit the following assertions for the validation:

$$\begin{aligned}
&(D_1)\ A \triangleleft \left\{N_A, \overset{K}{\rightarrow} H\right\}_{K^{-1}}, && M_2\\
&(D_2)\ A \mid\equiv \left(H \Rightarrow \overset{K}{\rightarrow} H\right), && D_1\\
&(D_3)\ A \mid\equiv \overset{K}{\rightarrow} H, && D_2 + A_1\\
&(D_4)\ A \mid\equiv (H \mid\sim N_A), && D_3 + D_1\\
&(D_5)\ A \mid\equiv \left(H \mid\sim K^{-1}\right), && D_3 + D_4 + A_2
\end{aligned}$$

The above assertions can be further explained as follows:

— In D_1, A receives a message containing the nonce signed by the private key K^{-1}. It seems the key comes from H. This is a consequence of M_2;

— In D_2, A verifies the signed nonce and asserts that the private/public key used to sign is under the jurisdiction of H;

— In D_3, since A believes that H believes in his own key K (assumption A_1) and that K is in the jurisdiction of H (assertion D_2), the "jurisdiction" rule says that A believes that K belongs to H;

— In D_4, since K (and by extension K^{-1}) belong to H (assertion D_3) and A received N_A signed by K^{-1}, the "message meaning" rule says that A believes H once said N_A;

HSM-BSN Authentication Protocol BAN Logic Evaluation The HSM-BSN Authentication Protocol ensures that the new node in the network (Z) is indeed an HSM-BSN. The authentication has to be challenged by an HSM-BSN.

The protocol consists of 3 messages:

$$
\begin{aligned}
&(M_1)\ Z \rightarrow H : Z \\
&(M_2)\ H \rightarrow Z : N_H, \{N_H\}_{K_H^{-1}} \\
&(M_3)\ Z \rightarrow H : \{N_H\}_{AK}
\end{aligned}
$$

The goal of this protocol is to prove to H that Z knows the common pre-shared key AK to encrypt and decrypt values.

$$
(G_1)\ H \mid\equiv \left(Z \mid\equiv \left(Z \overset{AK}{\longleftrightarrow} H\right)\right)
$$

In this protocol, we make the two assumptions that H believes in the pre-shared key AK and that H believes in the freshness of his own nonce.

$$
\begin{aligned}
&(A_1)\ H \mid\equiv \left(Z \overset{AK}{\longleftrightarrow} H\right) \\
&(A_2)\ H \mid\equiv (\#(N_H))
\end{aligned}
$$

With these assumptions, we can emit the following assertions for the validation:

$$
\begin{aligned}
&(D_1)\ H \triangleleft \left\{N_A, Z \overset{AK}{\longleftrightarrow} H\right\}_{AK}, && M_3 \\
&(D_2)\ H \mid\equiv \left(Z \mid\sim \left(N_A, Z \overset{AK}{\longleftrightarrow} H\right)\right), && A_1 + D_1 \\
&(D_3)\ H \mid\equiv \left(Z \mid\equiv \left(Z \overset{AK}{\longleftrightarrow} H\right)\right), && A_2 + D_2
\end{aligned}
$$

The above assertions can be further explained as follows:

— In D_1, H receives a message containing a nonce N_H encrypted by AK, supposedly shared between Z and H;

— In D_2, since H believes that AK is shared between Z and H (from A_1), and after receiving N_H encrypted by AK (from D_1), the "message meaning" rule says that H believes Z once sent N_H using AK;

— In D_3, since H believes the freshness of N_H (from A_2) and that Z sent N_H using AK (from D_2), the "nonce verification" rule says that Z must know the AK, reaching goal G_1.

Security of the HSM-BSN Identification Protocol To determine the security of the HSM-BSN Identification Protocol, we conduct a step-by-step analysis against known attacks:

— **Steps 1&2:** The use of a Nonce value by the requestor prevents an attacker from replaying an intercepted response from a previous round of the protocol;

— **Step 3:** The digital signature of the response prevents an attacker from tampering with an intercepted response, for example, by modifying the IP address;

— **Step 4:** In this step, an attacker (man-in-the-middle) can intercept the message and perform various attacks. However, the other steps of the protocol are designed to reduce the possible attacks;

— **Steps 5&6:** The digital signature verification algorithm is designed to verify the message's integrity and obtain the signer's Blockchain Address. Therefore, if a malicious node (not an HSM-BSN) attempts to be identified as an HSM-BSN, his response will be ignored because its address is not in the recorded authority addresses of the blockchain;

— **Step 7:** An attacker (man-in-the-middle) can intercept a response and send it to the requestor. If this attack succeeds, the Blockchain Address will be associated with the attacker's IP. Including the HSM-BSN's IP address in the signed response prevents this attack.

Security of the HSM-BSN Authentication Protocol To determine the security of the HSM-BSN Authentication Protocol, we conduct a step-by-step analysis against known attacks. To successfully falsify an authentication, the attacker must provide the challenger with a properly encrypted Nonce value using the AK. Although there are several ways to obtain this value, our protocol aims to make this task impossible for the attacker.

One way to obtain this nonce would be to enter an authentication agreement with an HSM-BSN where the attacker poses as the requestor (Attempt A). At the same time, the attacker poses as a challenger in another agreement with a different HSM-BSN (Attempt B). The Challenger of Attempt A sends the nonce to the attacker, which uses it as a nonce in Attempt B. The HSM-BSN encrypts the nonce in Attempt B and returns it to the attacker, who uses the encrypted value in Attempt A. However, our protocol is designed to be **reactive** through the Authentication Request (Step 1). In other words, the attacker cannot force an HSM-BSN to authenticate itself and, therefore, cannot obtain an encrypted nonce from the HSM-BSN.

Moreover, nonce values prevent an attacker (man-in-the-middle) from intercepting an encrypted nonce (Step 7) and using it in a different run of the protocol.

Security of the BlockDBMS Our proposed BlockDBMS distinguishes itself with many security features.

— **Data Resilience**: The distributed aspect of the database, combined with data replication on all HSM-BSNs, guarantees data resilience. If one of the HSM-BSNs fails or is destroyed, the system remains active and functional.

— **Data Verification**: The implemented smart contract gathers the query results from all HSM-BSN and requires 51% of answers to be identical to be accepted as the correct answer. If one of the HSM-BSN is faulty or tampered with, its response will be ignored. This verification of data guarantees the truthfulness of the result.

— **Data Encryption**: To maintain data confidentiality, BlockDBMS encrypts all sensitive data. To maintain strong confidentiality, integrity, and availability, we suggest that each localized database is protected against attacks using the defense mechanisms proposed in our previous work [85, 161]. In addition, we encrypt the data using the same key on all sites to allow easy migration and synchronization in case data restoration is needed on one site. This key is securely stored inside the HSM secure partition.

— **Access Revocation**: In many distributed database situations, all the participating nodes run a database instance and must be synchronized. However, this poses a security threat as these nodes could be compromised. In our case, only the HSM-BSNs, considered secure by design, are allowed to decrypt sensitive information only through the HSM. Therefore, access revocation is implemented simply by blocking malicious nodes from the blockchain.

— **Node Authentication**: Regular nodes are authentified in the blockchain using their properties, such as a blockchain ID. HSM-BSN are authenticated in the blockchain using the HSM-BSN IAP.

Table 6.2 summarizes the security aspects of our proposed method and compares it to similar candidates found in the literature.

6.5.3 Quantitative Results

The quantitative evaluation requires the analysis of the following metrics. We analyze the network occupancy for the HSM-BSN Identification and Authentication Protocol (HSM-BSN IAP). For BlockDBMS, we analyze the execution time of SQL queries.

Tableau 6.2 Comparison of this Work to Other Blockchain Storage Solutions

Method	Data Resilience	Data Encryption	Data Verif.	Access Revoc.	Node Auth.
BlockDBMS	✓	✓	✓	✓	✓
Nathan *et al.* [103] (Blockchain Meets Database)	✓				
McConaghy *et al.* [104] (BigchainDB)	✓				
Muzammal *et al.* [75] (Renovating Blockchain)	✓	✓			
CSIRO [67] (On-chain Encryption)		✓			

HSM-BSN Identification and Authentication Protocol Network Occupancy

Considering that the HSM-BSN IAP is not often executed and is not time-critical for the application's functioning, we can safely disregard its execution time. However, the network occupancy indicates the network traffic generated by the proposed HSM-BSN IAP. A recurrent problem in distributed consensus protocols is their scalability in terms of generated traffic. Indeed, some of these protocols saturate the network when the number of nodes increases drastically. Typically, those protocols follow a n^2 tendency, where n is the number of nodes.

In our case, we divide the nodes into regular nodes and HSM-BSNs. Our protocols are designed to be transparent to the public. However, only HSM-BSNs are invited to participate in the HSM-BSN IAP. We denote by n the number of regular nodes and by b the number of HSM-BSN. Table 6.3 depicts the number of messages required for each step of the protocol. We quickly notice that the number of messages equals $6b$, which is independent of n. With the realistic assumption that $b \ll n$, we conclude that the protocol is easily scalable to big productions.

Tableau 6.3 Number of Messages for the HSM-BSN IAP

Protocol Step	**Description**	**# of Messages**
Step 1: HSM-BSN Identification Protocol	New Node sends b identification requests and receives b responses	$2 \times b$
Step 2: HSM-BSN Authentication Protocol	The challenger exchanges 4 messages with each HSM-BSN	$4 \times b$

Distributed Database Execution Time

The DDB queries are divided into two types: *modification queries* and *selection queries*. Each has a different logic and, therefore, a different execution time. In both cases, we study the tendency of the execution time with respect to the number of requests and the number of HSM-BSN in the network.

Modification queries, such as INSERT, UPDATE, or DELETE, can be executed in the background, with the safe assumption that query order will prevent a selection query on data that is bound to be modified. Also, they do not require a consensus agreement from the HSM-BSN; they only need to reach the database and be locally executed.

Selection queries, however, require a consensus agreement from the HSM-BSN before a result can be returned. To determine the performance of the proposed BlockDBMS for the selection queries, we measure the response time for a varying number of requests.

Table 6.4 details the breakdown of a sample selection query execution.

Tableau 6.4 Example of Selection Query Execution Breakdown

Time	**Client**	**HSM-BSN1**	**HSM-BSN2**	**HSM-BSN3**
00.000	Send Tr#1A974			
01.714		Seal Blk#CD393		
02.340			Import Blk#CD393	Import Blk#CD393
02.539		Recv. Query		
02.572		Send Tr#76A69		
02.624			Recv. Query	
02.636			Send Tr#E280A	

(Suite à la page suivante)

Tableau 6.4 Example of Selection Query Execution Breakdown (Suite)

Time	**Client**	**HSM-BSN1**	**HSM-BSN2**	**HSM-BSN3**
06.735				Seal Blk#CD803
06.741		Import Blk#CD803		
06.746			Import Blk#CD803	
08.047	Callback			
08.050	Read Result			
08.056	Clear Tr#106D7			
08.061	Result			

By breaking down the various steps, we notice some time gaps in long execution. The first one is between 00.000 and 02.539. Indeed, even after the client submits the query through a transaction, it is not available for the HSM-BSNs until the block containing the transaction (CD393) is sealed in the blockchain and imported by the other nodes. The second and most significant gap is between 02.636 and 06.735. During that time, the two transactions (76A69 and E280A) are waiting to be sealed in block CD803. These two transactions contain the computed results for HSM-BSN1 and HSM-BSN2. Once these two results are sent to the contract, the majority (2 out of 3) is reached, and the client receives a callback at 8.047.

This analysis allows us to understand the nature of the induced latency: most of the time is wasted waiting for signers to submit their blocks containing the transactions. Because the mining process in the blockchain is not exact, the execution time varies between similar experimentations. We estimated that the average execution time varies around 10 seconds.

The time gaps in the breakdown, illustrated in Table 6.4, suggest that, up to a certain value, the number of requests does not affect the execution time. Indeed, the query rate for each database instance is very high compared to the block mining speed. Therefore, many transactions can be included in the same block, and the execution time should not drastically increase with the number of requests. This property of our proposed solution is essential to showcase its scalability.

With our setup, we measure the mean processing time for a query on a local database at 20ms, whereas the mean block mining time is 4 seconds. The mean number of queries per block ($nqpb$), caluclated in (6.4), is approximately 200 queries per block.

$$
\begin{aligned}
nqpb &= \frac{\text{block mining time}}{\text{local processing time}} \\
&= \frac{4}{0.02} \\
&= 200
\end{aligned}
\tag{6.4}
$$

The expected execution time against the number of requests for a SELECT query using our proposed BlockDBMS is depicted in Fig. 6.5.

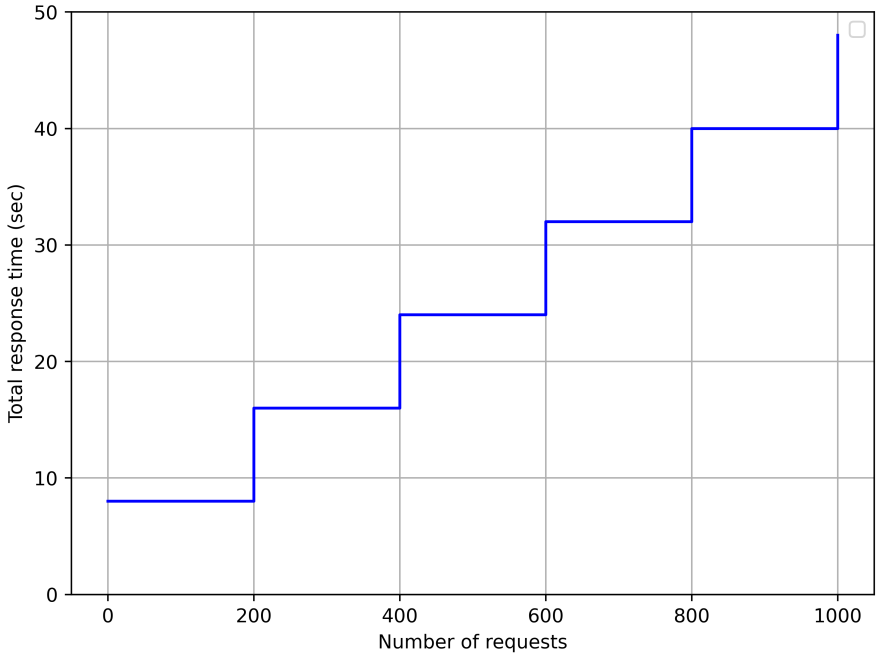

Figure 6.5 Selection query execution time against number of requests

6.5.4 Discussion

The implementation and evaluation of BlockDBMS highlighted its impact on the secure data distribution scheme, with an interesting impact on the performance.

Effective data distribution The main strength of BlockDBMS relies on its effective and robust data distribution scheme. As we noticed in the qualitative evaluation, we managed to achieve data resilience through replication, data verification through the aggregation mechanism, data encryption and access revocation through the HSM, and HSM-BSN authentication

through the IAP protocol. The access revocation component is of utmost importance because it forces malicious nodes to surrender their access to the database, preventing them from causing further damages. Indeed, in regular situations, it would be impossible to force a malicious node to delete their access keys. However, with the encryption being delegated to the trusted computing base, revoking the access becomes a simple matter of access control between the HSM and the different nodes. As shown in Table 6.2, our solution provides more security features than its contestants in literature.

Security and performance trade-off The quantitative evaluation of BlockDBMS shows an interesting step-like property of the execution time against the number of requests. Since the blockhain mining time which is far greater than the average processing time of the SQL request on the HSM-BSN, many requests are put on hold until the block is processed by the blockchain, which causes the delay.

6.6 Conclusion

In this paper, we proposed a Blockchain-Based Distributed Database Management System to efficiently distribute data in large systems while maintaining high data security. We introduced a new type of node, the HSM-Based Security Node (HSM-BSN), that manages the cryptographic aspect of the blockchain while also managing the database through a regular Database Management System (DBMS). Through smart contracts and their associated middleware, the blockchain nodes can transact and act like a standard distributed database where confidentiality and authenticity are prioritized.

This work was conducted in an effort to tackle the shortcomings of distributed databases in terms of concurrency, reliability, and harmonious encryption. To the extent of our knowledge, the existing methods in literature overlook the importance of trustworthy result aggregation, which constitutes a threat to data truthfulness. Indeed, in traditional methods, the closest or most available server could be compromised and return a false result.

BlockDBMS has shown to drastically increase data security in a distributed database, equating to the level of security expected in a localized database in regards to confidentiality, integrity, and availability. Indeed, by using the blockchain technology, every call and transaction is recorded and monitored, which strongly discourages tampering with the data in transit. Finally, using a private blockchain for the DDB shields the database instances on the HSM-BSN from outsiders. The regular server nodes, which are inside the blockchain, can freely query the DDB. For outside nodes, the regular server nodes must act as a proxy, and they constitute a first defense wall by filtering unauthorized queries. Moreover, the issues

of concurrency and data trustworthiness have been treated to be more robust through the aggregation smart contracts. We proposed a lightweight cryptography-based authentication protocol for joining HSM-BSN, which does not affect the performance of the blockchain.

Finally, our implementation shows that the execution time follows a large-step tendency caused by aggregating several queries in the same block on the chain. This gap can be reduced by lowering the block mining time. Therefore, the physical specifications of the system determine the width of the step.

However, the current architecture is designed to allow access revocation through authentication and access control of regular nodes to the HSM-BSNs. Such access control mechanisms have yet to be conceived and implemented as well to ensure access revocation.

In future work, we could focus on a realistic implementation of the proposed architecture, resulting in a more accurate performance evaluation. We should also complement this work with a smart access control mechanism for regular nodes.

Acknowledgment

The authors thank Dr. Franjieh El Khoury for her constructive comments and for proofreading this paper.

The authors would also like to thank the technical and engineering team at Entrust Corporation for providing valuable insights and documentation about HSM.

CHAPITRE 7 DISCUSSION GÉNÉRALE

Suite à l'exposition des travaux relatifs au modèle d'attaque internes, au modèle de détection des attaques internes et au modèle de distribution sécurisée des données, nous entamons dans ce chapitre une discussion générale sur ces travaux. Dans un premier temps, nous revenons sur les aspects méthodologiques qui nous ont permis d'atteindre les objectifs de recherche. Dans un deuxième temps, nous analysons l'ensemble des résultats obtenus.

7.1 Aspects méthodologiques

Cette thèse se focalise sur la sécurité des données entreposées sur les serveurs d'applications. En effet, dans un travail préliminaire, nous avons identifié dans la littérature scientifique les différents manques et vulnérabilités qui existent aujourd'hui concernant la sécurité des données. La revue de littérature a clairement souligné les faiblesses au niveau du traitement des attaques internes à un système informatique, en particulier leur identification et détection. De plus, la distribution des données apporte des défis supplémentaires comme la concurrence et le chiffrement distribué.

Au vu de ces lacunes, nous avons proposé trois volets qui permettent de traiter l'identification des attaques internes, la détection des attaques internes, et la distribution sécurisée des données.

La méthodologie adoptée dans chacun de ces volets suit les étapes suivantes :

1. Une revue critique de la littérature pour soulever les lacunes des solutions existantes et les points que nous pouvons améliorer ;
2. La conception d'une solution qui répond aux lacunes identifiées ;
3. L'implémentation d'un prototype pour la solution proposée ;
4. L'évaluation des performances de la solution proposée.

Ainsi, dans le premier volet, nous avons proposé un modèle d'attaques internes contre les architectures de sécurité basées sur HSM. Ce modèle d'attaques exhaustif permet d'identifier correctement les attaques internes pour ensuite les traiter.

Le second volet a été une continuation directe du premier, dans la mesure où nous avons proposé une architecture de sécurité pour détecter les attaques identifiées au premier volet.

Finalement, dans le troisième volet, nous avons proposé une méthodologie de distribution des bases de données grâce à la blockchain. Le but de ce volet était de permettre une distribution

efficace et sécuritaire des données, tout en assurant une non-concurrence des données entre les différentes requêtes.

7.2 Analyse des résultats

Les résultats obtenus au terme des travaux de cette thèse contribuent à améliorer la sécurité des données stockées sur les serveurs.

Tout d'abord, la définition d'un modèle d'attaques internes a permis d'identifier de façon exacte et précise les vulnérabilités d'une architecture de sécurité basée sur HSM. En effet, nous avons identifié 9 attaques contre la confidentialité, l'intégrité et la disponibilité des données. De plus, l'évaluation des attaques à partir de la métrique δ-score permet de classifier les attaques selon leur niveau de difficulté et la gravité de leur impact. Ainsi, le modèle d'attaques interne proposé est la première étape dans la méthodologie de traitement des attaques internes.

La seconde partie du travail a été de concevoir et implémenter une architecture de sécurité pour détecter les attaques internes. Ceci s'intègre dans la seconde étape dans la méthodologie de traitement des attaques internes. Pour cela, nous avons proposé 4 mécanismes de défense basés sur la cryptographie. Ces mécanismes ont été évalués sur leur efficacité en termes de sécurité et sur la latence introduite. Ainsi, le Nonce-Based Process Authentication (NBPA) détecte les attaques de l'homme du milieu et du réutilisation des mots-de-passe entre le serveur et le HSM. Cette détection se fait de façon presque immédiate, et le mécanisme implique un délai modeste de 14%. Le Hash-Based Field Integrity (HBFI) et Hash-Based Field Availability (HBFA) se basent sur les hachages cryptographiques pour détecter toute modification des champs des bases de données. La détection se fait dès que la donnée est sollicitée auprès de la base de données et ces mécanismes impliquent un délai de 30 à 50% pour HBFI, 25% pour HBFA et 43.74% pour l'ensemble des mécanismes combinés. Enfin, le Hash-Based Row Availability (HBRA) se base sur les hachages cryptographiques pour détecter une suppression d'entrée dans la base de données. Ce mécanisme fonctionne en arrière-plan et a un impact négligeable sur les performances de l'application. La détection de la suppression est de l'ordre de quelques minutes pour une base de données avec 1000 entrées, en fonction des paramètres du mécanisme.

La dernière partie du travail concerne la distribution sécurisée des données dans les bases de données. En particulier, nous avons proposé une architecture de SGBD distribué basé sur la blockchain et HSM pour faciliter la distribution tout en maintenant un haut niveau de sécurité. L'évaluation montre que le système proposé permet d'avoir la résilience des données

grâce à la distribution, le chiffrement des données, le partage sécuritaire de la clé grâce au HSM, la révocation d'accès par la séparation entre le serveur et le HSM, la non-concurrence des données grâce aux contrats intelligents, la validation des résultats grâce aux contrats intelligents, et l'authentification des HSM-BSN grâce au protocole d'authentification. De plus, l'évaluation qualitative montre un potentiel évolutif puisque plusieurs transactions peuvent se faire dans un même nœud.

En conclusion, nous avons atteint les objectifs de recherche énoncés à la section 1.3 à travers les trois volets de cette thèse. Chacun de ces volets a été traité dans un article de revue scientifique, présenté dans les chapitres 4, 5 et 6.

CHAPITRE 8 CONCLUSION

Ce chapitre final récapitule les travaux réalisés dans la présente thèse. Tout d'abord, nous mettons en évidence les contributions de la thèse au savoir scientifique. Ensuite, nous identifions les limitations des travaux réalisés. Finalement, nous énonçons des pistes de travaux futurs qui peuvent améliorer ou compléter le présent travail.

8.1 Synthèse des travaux

Notre objectif principal pour cette thèse est de proposer une architecture de sécurité basée sur une technologie adaptée et spécialisée pour protéger les données dans les architectures locales et distribuées. Cet objectif répond à la problématique de sécurité des données applicatives dans un système informatique. En effet, nos travaux s'attardent sur trois aspects différents de la sécurité des données qui se sont traduits par les différents volets de la thèse.

Dans le premier volet, nous cherchions à identifier les vulnérabilités d'un système vis-à-vis d'attaquants internes. Pour arriver à ce premier objectif, nous avons proposé un modèle d'attaques internes sur les architectures de sécurité basées sur HSM. Ce modèle a été obtenu en menant une étude exhaustive sur les plages d'attaque et les vulnérabilités de l'architecture initiale. Ainsi, nous avons identifié 9 attaques qui compromettent la sécurité des attaques : le vol ou le remplacement de la clé privée, l'accès à la Local Master Key (LMK) du HSM, l'accès à la base de données et l'altération, la corruption, l'échange, la nullification ou la suppression de données de la base de données. Les attaques ont également été classées selon une notre métrique δ-score pour déterminer leur impact sur la sécurité des données.

Dans le second volet, nous cherchions à protéger les données contre les menaces internes. Pour cela, nous avons proposé une architecture de sécurité basée sur HSM qui répond au modèle d'attaques du premier volet. Cette architecture se focalise sur la détection des attaques internes en utilisant la cryptographie. Ainsi, nous avons conçu et implémenté 4 mécanismes de défense qui interagissent avec les données pour détecter les attaques. Le mécanisme *Nonce-Based Process Authentication* (NBPA) améliore l'authentification des processus avec le HSM. Les mécanismes *Hash-Based Field Integrity* (HBFI) et *Hash-Based Field Availability* (HBFA) permettent de détecter une altération, un échange ou une nullification sur les champs de la base de données. Le mécanisme *Hash-Based Row Availability* (HBRA) permet de détecter une suppression d'entrées dans la base de données. Puisque ces mécanismes sont basés sur la cryptographie, ils permettent de détecter sans faute les attaques ciblées par le modèle

d'attaques internes. De plus, le délai introduit par ces nouveaux mécanismes est acceptable pour l'ajout en termes de sécurité de l'information.

Dans le troisième volet, nous cherchions à distribuer des données dans un large système informatique sans compromettre leur sécurité. Pour cela, nous avons proposé un système de gestion de bases de données (SGBD) distribué basé sur HSM et la blockchain. Il introduit au sein d'une blockchain privée un nouveau nœud de sécurité, le HSM-Based Security Node (HSM-BSN). Ce nœud est responsable à la fois de maintenir une instance locale de la base de données, mais également de chiffrer les données grâce au HSM. Ceci permet la révocation d'accès puisque le nœud blockchain ne possède pas la clé de chiffrement. De plus, l'utilisation de contrats intelligents permet d'ordonnancer les requêtes pour assurer une non-concurrence des données entre plusieurs requêtes. L'évaluation de ce SGBD a montré son ajout en termes de sécurité et d'efficacité de distribution des données, mais également son potentiel évolutif.

8.2 Contributions

Les travaux réalisés ont permis de contribuer à l'avancement scientifique dans le domaine de la sécurité des données. Ainsi, les contributions principales de cette thèse se résument comme suit :

Conception d'un modèle d'attaques internes : La revue de littérature au Chapitre 2 montre distinctement l'absence de méthodologie claire pour identifier les attaques internes dans un système informatique. Cette identification est nécessaire pour mener à une architecture de défense adéquate. En effet, les travaux de la littérature qui faisaient abstraction de cette étape se retrouvent parfois avec des mécanismes de défense qui ne correspondent pas à la situation décrite, par exemple en protégeant un aspect qui n'était pas vulnérable, ou encore en délaissant une vulnérabilité inaperçue.

Notre premier volet, à travers l'établissement d'un modèle d'attaques internes au Chapitre 4, pallie à ce manque dans la mitigation des attaques internes. En particulier, ce modèle d'attaques internes cible les architectures de sécurité basées sur HSM. Le choix de cette architecture initiale n'est pas anodin dans la mesure où les organisations qui incluent des HSM dans leurs architectures ont tendance à croire que la sécurité de leurs données est devenue absolue. Ainsi, le modèle d'attaques montre de façon tangible les vulnérabilités internes de cette architecture. Son importance se démarque également grâce à l'implémentation concrète des attaques et la métrique d'évaluation δ-score proposée qui permet de juger le risque posé par une attaque en fonction de sa difficulté et de son impact. Cette métrique permet donc de classer les attaques de façon quantitative et est essentielle pour déterminer les priorités dans

le cas où des concessions doivent être faites.

Ainsi, le premier volet de cette thèse contribue à l'identification des attaques internes en proposant une méthodologie conception de modèles d'attaques internes et une métrique efficace pour classifier les attaques. Ce volet est également nécessaire pour tout travail subséquent relatif à la mitigation des attaques internes, comme la détection ou la protection.

Conception d'un modèle de détection des attaques internes : Le modèle de détection des attaques internes proposé au Chapitre 5 apporte une nouvelle dimension à la sécurité des données contre les attaques internes.

En effet, le premier aspect novateur se retrouve dans l'adaptation de l'interface PKCS #11 pour répondre aux attaques sur l'authentification des sessions. En effet, la nouvelle méthode proposée assure une authentification continue plus robuste pour détecter de façon immédiate les usurpations de sessions PKCS #11. De plus, notre modèle de détection des attaques internes propose une restructuration des données invisible à l'utilisateur tout en assurant l'intégrité et la disponibilité. Cette particularité est notable car elle permet d'intégrer facilement et rapidement les mécanismes de défense sans avoir à modifier la façon dont l'utilisateur interagit avec l'application. Les mécanismes de défense relatifs à cette restructuration sont basés sur la cryptographie et, contrairement aux modèles basés sur l'apprentissage, permettent une détection sans faute et quasi-immédiate des attaques ciblées. Enfin, le mécanisme de validation des entrées continu et optimisé assure la disponibilité des données en arrière-plan. Contrairement aux approches « sur demande » proposées dans la littérature, ce mécanisme vient répondre à la problématique de latence sur les opérations de lecture des données. En effet, un impact inévitable de la sécurité se retrouve au niveau des performances des requêtes lorsque plusieurs protocoles de validation doivent être enchaînés avant de retourner le résultat à l'utilisateur. Dans le but de minimiser cet impact, nous avons proposé ce mécanisme continu, ce qui permet de profiter des temps morts du système pour valider certaines données.

Les revendications en terme de sécurité de ce modèle de détection des attaques internes ont bien été montrées lors de l'évaluation qualitative. En effet, notre modèle est unique dans son habileté à garantir à la fois la confidentialité, l'intégrité, la disponibilité et la résistance aux attaques side-channel dans le contexte des attaques internes, en éliminant toute charge supplémentaire à l'utilisateur et en minimisant le rayon de confiance requis.

Conception d'un SGBD distribué basé sur HSM et la Blockchain : Le modèle BlockDBMS proposé au Chapitre 6, soit un SGBD distribué sécurisé par la blockchain et HSM, a apporté des nouveautés dans le domaine des données distribuées.

Tout d'abord, il a été nécessaire de concevoir des nouveaux éléments pour la blockchain qui permettent le chiffrement partagé d'informations sur la blockchain. Grâce aux HSM-BSN et à des contrats intelligents, nous avons réussi à stocker et partager des données critiques sur la blockchain même. Contrairement à la méthodologie standard de chiffrement sur la blockchain [67], notre méthodologie respecte également la révocation d'accès et ne contient pas de vulnérabilité pour le partage de la clé.

L'utilisation des contrats intelligents également mené à une technologie innovante de SGBD distribué grâce à HSM et la blockchain. En effet, le modèle proposé permet d'effectuer les opérations attendues pour une base de données tout en répondant aux problèmes les plus majeurs du paradigme des bases de données distribuées, soit la concurrence et la confidentialité. En opposition aux travaux existants, notre modèle permet efficacement la *distribution*, le *chiffrement*, la *révocation d'accès*, la *validation*, et l'*authentification des nœuds*.

L'originalité de ce volet se base sur la restructuration efficace entre les acteurs de la base de données distribuée. En effet, bien que la blockchain et les HSM semblent être deux paradigmes très disjoints, nous avons obtenu une interopérabilité robuste qui tire profit des deux paradigmes pour améliorer la sécurité dans un milieu distribué où la confiance est mince, sans pour autant limiter l'accès régulier aux données.

8.3 Limitations des travaux réalisés

Les travaux exposés dans cette thèse ont certes positivement contribué à la sécurité des données sur les serveurs d'applications. Cependant, certaines limitations existent toujours.

Paradigme du modèle d'attaques internes : Le modèle d'attaques internes proposé au Chapitre 4 est relié à un paradigme bien spécifique d'architecture de données basée sur HSM. Il suppose la présence ou l'absence de certains nœuds particuliers au réseau, ainsi qu'une interaction bien spécifique entre ces nœuds. En effet, certaines hypothèses de sécurité ont été prises pour acquis, ce qui n'est pas toujours le cas. En effet, une architecture moins stricte contiendrait davantage de vulnérabilités et donc aurait un modèle d'attaques internes différent. De plus, ce modèle d'attaques ne s'applique pas sur les autres paradigmes tels que le cloud, l'internet des objets, ou les réseaux *peer-to-peer*.

Continuité du modèle de détection des attaques internes : L'architecture de sécurité proposée au Chapitre 5 se focalise sur la détection des attaques internes. Ceci fait référence à la deuxième étape parmi les 5 étapes de la méthodologie de gestion des attaques internes. La prévention, la réponse et la récupération ne sont pas traitées par cette architecture de

sécurité. Par exemple, le mécanisme de défense NBPA n'empêche pas l'attaquant de voler le nonce pour communiquer illégalement avec le HSM. Cependant, la désynchronisation est imminente. Ainsi, le NBPA détecte effectivement l'attaque, sans la prévenir. De même, les mécanismes HBFI, HBFA et HBRA permettent de détecter rapidement une attaque sur les données, mais sans pour autant l'empêcher ou permettre un retour à l'état correct des données.

Désynchronisation des instances du SGBD distribué : La méthodologie de SGBD distribué basé sur la blockchain et HSM, proposé au Chapitre 6, permet la synchronisation de toutes les instances de la base de données grâce à l'ordonnancement des requêtes dans les contrats intelligents. Cependant, si une attaque ou une erreur désynchronise le système, notre SGBD distribué ne permet pas de détecter et de régler le problème.

Évaluation du SGBD distribué : L'implémentation du SGBD distribué a été menée sur un petit ensemble de machines virtuelles dans un milieu à ressources informatiques réduites. Bien que cette implémentation ait été suffisante pour évaluer la sécurité apportée par le SGBD distribué, elle n'a permis qu'une évaluation quantitative préliminaire. En effet, nous avons émis la conjecture que la tendance du temps de traitement suit une fonction en escalier à cause du temps d'inscription des blocs qui est bien supérieur au temps de traitement des requêtes. Cependant, à cause de l'interférence entre notre solution et le partage des ressources pour maintenir plusieurs nœuds blockchain, base de données, et HSMs, nous n'avons pas été en mesure d'observer la tendance voulue.

8.4 Travaux futurs

Pour conclure cette thèse, nous recommandons les directions suivantes pour des travaux futurs. Ces directions sont directement liées aux limitations exposées à la Section 8.3.

Autres modèles d'attaques internes : La méthodologie qui nous a permis d'obtenir le modèle d'attaques internes peut être utilisée pour concevoir des modèles d'attaques internes reliés à d'autres paradigmes tels que le cloud, l'internet des objets, les réseaux *peer-to-peer*, ou les réseaux véhiculaires. Ainsi, les vulnérabilités de chacune de ces architectures doivent être identifiées, implémentées, et classées selon la métrique δ-score du rapport impact/difficulté. Pour améliorer la qualité de cette étude, les modèles doivent être représentatifs de la réalité dans la meilleure situation permise par l'état de l'art actuel. En d'autres termes, l'architecture initiale sur laquelle le modèle d'attaque est basé doit comprendre les avancées les plus

modernes pour protéger les données dans le paradigme choisi.

Modèle d'attaques internes automatisé : Éventuellement, avec la conception de plusieurs modèles d'attaques internes, il sera possible de déceler un lien direct entre certaines caractéristiques du réseau et la présence d'une vulnérabilité. Ainsi, un outil informatique basé sur l'intelligence artificielle et l'apprentissage machine pourrait permettre la conception automatique d'un modèle d'attaques internes spécifique à n'importe quelle architecture donnée en entrée. Pour cela, une première partie se chargera d'étudier les métriques utiles qui seront isolées et pré-traitées pour pouvoir être consommable par l'algorithme d'apprentissage. Par la suite, le modèle intelligent doit être choisi, implémenté, et entraîné avec le jeu de données existant. Ceci permettra d'obtenir des modèles d'attaques internes pour des architecture quelconques, y-compris des architectures qui combinent plusieurs paradigmes, comme des architectures IoT véhiculaires, ou des architectures IoT cloud.

Prévention des attaques internes : En revenant à la méthodologie en 5 étapes de gestion des attaques internes, la prochaine étape est le volet de protection contre les attaques internes. Ce volet fait l'objet de nouveaux mécanismes de défense qui empêchent activement un attaquant interne de mener son attaque. Ces mécanismes de défense peuvent être de nature variée (architectures, modèles ou nouveaux algorithmes de chiffrement) et peuvent se localiser sur trois sites : le serveur, le terminal client, et le canal de communication. L'utilisation d'éléments de sécurité matérielle (i.e., carte SIM) et la conception d'une architecture de chiffrement propriétaire des données de l'utilisateur pourrait contribuer à la prévention des attaques internes sur le terminal. Ceci impliquerait alors, sur le serveur applicatif, une nouvelle méthodologie pour traiter des données chiffrées par l'utilisateur tout en maintenant l'utilisabilité des données.

Protection des données dans les environnements distribués Les modèles de distribution des données sont de plus en plus utilisés dans la littérature et l'industrie. Par exemple, l'utilisation des réseaux blockchain contribue de façon intrinsèque à l'intégrité et la disponibilité des données. Dans la philosophie de distribution sécurisée des données, la fragmentation des données (contrairement à la réplication que nous avons adoptée dans cette thèse) apporte de nouveaux défis en terme de validation des données, car les informations ne sont pas réparties de façon identiques entre les différents sites. De plus, il est nécessaire de proposer des mécanismes de resynchronisation efficaces pour répondre à des attaques sur l'intégrité ou la disponibilité des données. L'aspect des performances d'un tel système ne doit pas être négligé, et les limitations montrent qu'il y a encore matière à progression au niveau du temps

de traitement des requêtes. Finalement, le « droit à l'oubli » lorsqu'il s'agit d'informations personnelles est un enjeu sérieux qui freine l'utilisation de la blockchain pour les bases de données distribuées. Ainsi, un modèle de protection des données dans les environnements distribués doit prendre en compte ce droit pour protéger la vie privée des utilisateurs.

RÉFÉRENCES

[1] "Cisco Annual Internet Report - Cisco Annual Internet Report (2018–2023) White Paper." [En ligne]. Disponible : https://www.cisco.com/c/en/us/solutions/collateral/executive-perspectives/annual-internet-report/white-paper-c11-741490.html

[2] Centro de Innovación BBVA, "Mobile Banking – New Experience in the Post PC Era," *Innovation Edge*, vol. 2, avr. 2012. [En ligne]. Disponible : https://issuu.com/cibbva/docs/innovation_edge_mobile_banking

[3] M. Correa, J. Ramón García et A. Tabanera, "E-commerce and consumption habits in Spain : the importance of online banking," *Digital Economy Watch*, janv. 2015. [En ligne]. Disponible : https://www.bbvaresearch.com/en/tag/e-commerce-2/

[4] M. De Soete, "Mobile Payments," dans *Encyclopedia of Cryptography and Security*, H. C. A. van Tilborg et S. Jajodia, édit. Boston, MA : Springer US, 2011, p. 788–789. [En ligne]. Disponible : http://link.springer.com/10.1007/978-1-4419-5906-5_292

[5] S. Karnouskos, "Mobile payment : A journey through existing procedures and standardization initiatives," *IEEE Communications Surveys Tutorials*, vol. 6, nº. 4, p. 44–66, 2004, conference Name : IEEE Communications Surveys Tutorials.

[6] S. Pierre et O. Italis, "Les systèmes de paiement mobile à l'ère de la Covid-19 : sécurité, vie privée et confiance numérique," *Technologie et innovation*, vol. 6, nº. 1, 2021. [En ligne]. Disponible : https://www.openscience.fr/Les-systemes-de-paiement-mobile-a-l-ere-de-la-Covid-19-securite-vie-privee-et

[7] R. Tiwari et S. Buse, *The Mobile Commerce Prospects : A Strategic Analysis of Opportunities in the Banking Sector*. Hamburg University Press, 2007. [En ligne]. Disponible : https://library.oapen.org/handle/20.500.12657/27668

[8] D. Dennehy et D. Sammon, "Trends in mobile payments research : A literature review," *Journal of Innovation Management*, vol. 3, mars 2015.

[9] J. T. Isaac et Z. Sherali, "Secure Mobile Payment Systems," *IT Professional*, vol. 16, nº. 3, p. 36–43, mai 2014, conference Name : IT Professional.

[10] J. Müthing, R. Brüngel et C. M. Friedrich, "Server-Focused Security Assessment of Mobile Health Apps for Popular Mobile Platforms," *Journal of Medical Internet Research*, vol. 21, nº. 1, p. e9818, janv. 2019. [En ligne]. Disponible : https://www.jmir.org/2019/1/e9818/

[11] M. Perry et L. J. Fennelly, "Data Center and Server Security," dans *Physical Security : 150 Things You Should Know*. Oxford, UNITED STATES :

Elsevier Science & Technology, 2016, p. 68–69. [En ligne]. Disponible : http://ebookcentral.proquest.com/lib/polymtl-ebooks/detail.action?docID=4730474

[12] L. D. Tsobdjou, S. Pierre et A. Quintero, "A New Mutual Authentication and Key Agreement Protocol for Mobile Client—Server Environment," *IEEE Transactions on Network and Service Management*, vol. 18, nº. 2, p. 1275–1286, 2021.

[13] C. Rodrigo, S. Pierre, R. Beaubrun et F. El Khoury, "BrainShield : A Hybrid Machine Learning-Based Malware Detection Model for Android Devices," *Electronics*, vol. 10, nº. 23, 2021.

[14] Y. Wang, C. Hahn et K. Sutrave, "Mobile payment security, threats, and challenges," dans *2016 Second International Conference on Mobile and Secure Services (MobiSecServ)*, févr. 2016, p. 1–5.

[15] S. Samonas et D. Coss, "The CIA Strikes Back : Redefining Confidentiality, Integrity and Availability in Security," *Journal of Information System Security*, vol. 10, nº. 3, p. 25.

[16] S. Keykhaie et S. Pierre, "Mobile Match on Card Active Authentication Using Touchscreen Biometric," *IEEE Transactions on Consumer Electronics*, vol. 66, nº. 4, p. 376–385, nov. 2020, conference Name : IEEE Transactions on Consumer Electronics.

[17] A. Fournier, "Détection de programmes malveillants dédiée aux appareils mobiles," Mémoire de maîtrise, École Polytechnique de Montréal, déc. 2018. [En ligne]. Disponible : https://publications.polymtl.ca/3700/

[18] A. Fournier, F. El Khoury et S. Pierre, "Classification Method for Malware Detection on Android Devices," dans *Proceedings of the Future Technologies Conference (FTC) 2020, Volume 3*, ser. Advances in Intelligent Systems and Computing, K. Arai, S. Kapoor et R. Bhatia, édit. Cham : Springer International Publishing, 2021, p. 810–829.

[19] R. L. Rivest, "Cryptography," dans *Handbook of Theoretical Computer Science (Vol. A) : Algorithms and Complexity*. Cambridge, MA, USA : MIT Press, 1991, p. 617–755.

[20] "Definition of BINARY NOTATION." [En ligne]. Disponible : https://www.merriam-webster.com/dictionary/binary+notation

[21] S. Pierre, *Réseaux et systèmes informatiques mobiles : fondements, architectures et applications*. Montréal, QC : Presses internationales Polytechnique, 2011, oCLC : 794300589.

[22] B. Kaliski, "Symmetric Cryptosystem," dans *Encyclopedia of Cryptography and Security*, H. C. A. van Tilborg et S. Jajodia, édit. Boston, MA : Springer US, 2011, p. 1271–1271. [En ligne]. Disponible : https://doi.org/10.1007/978-1-4419-5906-5_438

[23] National Institute of Standards and Technology, “Advanced Encryption Standard (AES),” National Institute of Standards and Technology, FIPS 197, nov. 2001. [En ligne]. Disponible : https://csrc.nist.gov/publications/detail/fips/197/final

[24] “AES,” dans *Encyclopedia of Cryptography and Security*, H. C. A. van Tilborg et S. Jajodia, édit. Boston, MA : Springer US, 2011, p. 26–26. [En ligne]. Disponible : https://doi.org/10.1007/978-1-4419-5906-5_1207

[25] P. S. . D. M. Turner (guests), “Symmetric Key Encryption - why, where and how it’s used in banking.” [En ligne]. Disponible : https://www.cryptomathic.com/news-events/blog/symmetric-key-encryption-why-where-and-how-its-used-in-banking

[26] B. Kaliski, “Public-Key Cryptography Standards (PKCS) #8 : Private-Key Information Syntax Specification Version 1.2,” EMC, RFC 5208, mai 2008. [En ligne]. Disponible : https://tools.ietf.org/html/rfc5208

[27] ——, “Asymmetric Cryptosystem,” dans *Encyclopedia of Cryptography and Security*, H. C. A. van Tilborg et S. Jajodia, édit. Boston, MA : Springer US, 2011, p. 49–50. [En ligne]. Disponible : https://doi.org/10.1007/978-1-4419-5906-5_394

[28] E. Rescorla, “The Transport Layer Security (TLS) Protocol Version 1.3,” Internet Engineering Task Force (IETF), RFC 8446, août 2018. [En ligne]. Disponible : https://tools.ietf.org/html/rfc8446

[29] T. Dierks et E. Rescorla, “The Transport Layer Security (TLS) Protocol Version 1.2,” Internet Engineering Task Force (IETF), RFC 5246, août 2008. [En ligne]. Disponible : https://tools.ietf.org/html/rfc5246

[30] ——, “The Transport Layer Security (TLS) Protocol Version 1.1,” Internet Engineering Task Force (IETF), RFC 4346, avr. 2006. [En ligne]. Disponible : https://tools.ietf.org/html/rfc4346

[31] E. Rescorla, “Diffie-Hellman Key Agreement Method,” RTFM Inc., RFC 2631, juin 1999. [En ligne]. Disponible : https://tools.ietf.org/html/rfc2631

[32] B. Kaliski, J. Jonsson, K. Moriarty et A. Rush, “PKCS #1 : RSA Cryptography Specifications,” RSA Laboratories, East, RFC 8017, nov. 2016. [En ligne]. Disponible : https://tools.ietf.org/html/rfc8017

[33] J. Lopez et R. Dahab, “An overview of elliptic curve cryptography,” Institute of Computing, State University of Campinas, Brazil, Technical Report, 2000, publisher : Citeseer. [En ligne]. Disponible : https://citeseerx.ist.psu.edu/viewdoc/summary?doi=10.1.1.37.2771

[34] M. Wiener, “Cryptanalysis of short RSA secret exponents,” *IEEE Transactions on Information Theory*, vol. 36, n°. 3, p. 553–558, mai 1990, conference Name : IEEE

Transactions on Information Theory.

[35] RSA Laboratories, “PKCS #3 : Diffie-Hellman Key- Agreement Standard,” RSA Laboratories, East, Technical Note, nov. 1993. [En ligne]. Disponible : https://www.teletrust.de/fileadmin/files/oid/oid_pkcs-3v1-4.pdf

[36] M. Just, “Diffie–Hellman Key Agreement,” dans *Encyclopedia of Cryptography and Security*, H. C. A. van Tilborg et S. Jajodia, édit. Boston, MA : Springer US, 2011, p. 341–342. [En ligne]. Disponible : https://doi.org/10.1007/978-1-4419-5906-5_75

[37] D. Boneh, “Digital Signature Standard,” dans *Encyclopedia of Cryptography and Security*, H. C. A. van Tilborg et S. Jajodia, édit. Boston, MA : Springer US, 2011, p. 347–347. [En ligne]. Disponible : https://doi.org/10.1007/978-1-4419-5906-5_145

[38] B. Preneel, “Hash Functions,” dans *Encyclopedia of Cryptography and Security*, H. C. A. van Tilborg et S. Jajodia, édit. Boston, MA : Springer US, 2011, p. 543–553. [En ligne]. Disponible : http://link.springer.com/10.1007/978-1-4419-5906-5_580

[39] D. Eastlake et T. Hansen, “US Secure Hash (SHA and SHA-based HMAC and HKDF),” AT&T Labs, Huawei, RFC 6234, mai 2011. [En ligne]. Disponible : https://www.ietf.org/rfc/rfc1321.txt

[40] R. Rivest, “The MD5 Message-Digest Algorithm,” MIT Laboratory for Computer and RSA Data Security, Inc., RFC 1321, avr. 1992. [En ligne]. Disponible : https://www.ietf.org/rfc/rfc1321.txt

[41] B. Preneel, “Preimage Resistance,” dans *Encyclopedia of Cryptography and Security*, H. C. A. Tilborg, édit. Springer US, 2005, p. 465–466. [En ligne]. Disponible : http://link.springer.com/10.1007/0-387-23483-7_309

[42] ——, “Second preimage resistance,” dans *Encyclopedia of Cryptography and Security*, H. C. A. Tilborg, édit. Springer US, 2005, p. 543–544. [En ligne]. Disponible : http://link.springer.com/10.1007/0-387-23483-7_372

[43] ——, “Collision resistance,” dans *Encyclopedia of Cryptography and Security*, H. C. A. Tilborg, édit. Springer US, 2005, p. 81–82. [En ligne]. Disponible : http://link.springer.com/10.1007/0-387-23483-7_69

[44] I. B. Damgård, “A design principle for hash functions,” dans *Conference on the Theory and Application of Cryptology*. Springer, 1989, p. 416–427.

[45] S. Turner, “Using SHA2 Algorithms with Cryptographic Message Syntax,” IECA, RFC 5754, janv. 2010. [En ligne]. Disponible : https://tools.ietf.org/html/rfc5754

[46] T. Lange, “Hash-Based Signatures,” dans *Encyclopedia of Cryptography and Security*, H. C. A. van Tilborg et S. Jajodia, édit. Boston, MA : Springer US, 2011, p. 540–542. [En ligne]. Disponible : http://link.springer.com/10.1007/978-1-4419-5906-5_413

[47] X. Wang, “Computational Puzzles,” dans *Encyclopedia of Cryptography and Security*, H. C. A. van Tilborg et S. Jajodia, édit. Boston, MA : Springer US, 2011, p. 244–245. [En ligne]. Disponible : http://link.springer.com/10.1007/978-1-4419-5906-5_264

[48] B. Singhal, G. Dhameja et P. S. Panda, *Beginning Blockchain : A Beginner’s Guide to Building Blockchain Solutions*. Berkeley, CA, UNITED STATES : Apress L. P., 2018. [En ligne]. Disponible : http://ebookcentral.proquest.com/lib/polymtl-ebooks/detail.action?docID=5448970

[49] T. Caddy, “Side-Channel Attacks,” dans *Encyclopedia of Cryptography and Security*, H. C. A. van Tilborg et S. Jajodia, édit. Boston, MA : Springer US, 2011, p. 1204–1204. [En ligne]. Disponible : https://doi.org/10.1007/978-1-4419-5906-5_227

[50] S. Bhunia et M. Tehranipoor, “Side-Channel Attacks,” dans *Hardware Security*. Elsevier, 2019, p. 193–218. [En ligne]. Disponible : https://linkinghub.elsevier.com/retrieve/pii/B9780128124772000137

[51] M. Pudovkina, “A Related-Key Attack on Block Ciphers with Weak Recurrent Key Schedules,” dans *Foundations and Practice of Security*, J. Garcia-Alfaro et P. Lafourcade, édit. Berlin, Heidelberg : Springer Berlin Heidelberg, 2012, p. 90–101.

[52] B. Zhao, L. Wang, K. Jiang, X. Liang, W. Shan et J. Liu, “An Improved Power Attack on Small RSA Public Exponent,” dans *2016 12th International Conference on Computational Intelligence and Security (CIS)*, déc. 2016, p. 578–581.

[53] M. Joye et F. Olivier, “Side-Channel Analysis,” dans *Encyclopedia of Cryptography and Security*, H. C. A. van Tilborg et S. Jajodia, édit. Boston, MA : Springer US, 2011, p. 1198–1204. [En ligne]. Disponible : https://doi.org/10.1007/978-1-4419-5906-5_516

[54] T. Caddy, “Differential Power Analysis,” dans *Encyclopedia of Cryptography and Security*, H. C. A. van Tilborg et S. Jajodia, édit. Boston, MA : Springer US, 2011, p. 336–338. [En ligne]. Disponible : https://doi.org/10.1007/978-1-4419-5906-5_196

[55] O. Benot, “Fault Attack,” dans *Encyclopedia of Cryptography and Security*, H. C. A. van Tilborg et S. Jajodia, édit. Boston, MA : Springer US, 2011, p. 452–453. [En ligne]. Disponible : https://doi.org/10.1007/978-1-4419-5906-5_505

[56] F. Koeune, “Timing Attack,” dans *Encyclopedia of Cryptography and Security*, H. C. A. van Tilborg et S. Jajodia, édit. Boston, MA : Springer US, 2011, p. 1303–1305. [En ligne]. Disponible : https://doi.org/10.1007/978-1-4419-5906-5_138

[57] M. N. M. Bhutta, A. A. Khwaja, A. Nadeem, H. F. Ahmad, M. K. Khan, M. A. Hanif, H. Song, M. Alshamari et Y. Cao, “A Survey on Blockchain Technology : Evolution, Architecture and Security,” *IEEE Access*, vol. 9, p. 61 048–61 073, 2021.

[58] "What Is Cryptocurrency ?" 2022. [En ligne]. Disponible : https://www.investopedia.com/terms/c/cryptocurrency.asp

[59] D. Mingxiao, M. Xiaofeng, Z. Zhe, W. Xiangwei et C. Qijun, "A review on consensus algorithm of blockchain," dans *2017 IEEE International Conference on Systems, Man, and Cybernetics (SMC)*. Banff, AB : IEEE, oct. 2017, p. 2567–2572.

[60] "Smart Contracts : What You Need to Know." [En ligne]. Disponible : https://www.investopedia.com/terms/s/smart-contracts.asp

[61] I. Homoliak, S. Venugopalan, D. Reijsbergen, Q. Hum, R. Schumi et P. Szalachowski, "The Security Reference Architecture for Blockchains : Toward a Standardized Model for Studying Vulnerabilities, Threats, and Defenses," *IEEE Communications Surveys Tutorials*, vol. 23, n°. 1, p. 341–390, 2021.

[62] "Merkle-Hash-Trees Signatures," dans *Encyclopedia of Cryptography and Security*, H. C. A. van Tilborg et S. Jajodia, édit. Boston, MA : Springer US, 2011, p. 774–774. [En ligne]. Disponible : http://link.springer.com/10.1007/978-1-4419-5906-5_1132

[63] Z. Zheng, S. Xie, H. Dai, X. Chen et H. Wang, "An Overview of Blockchain Technology : Architecture, Consensus, and Future Trends," dans *2017 IEEE International Congress on Big Data (BigData Congress)*, juin 2017, p. 557–564.

[64] J. Clark, "Recent Developments in Information Systems Security," Concordia University. [En ligne]. Disponible : https://users.encs.concordia.ca/~clark/courses/1803-6630/index.html

[65] D. Tosh, S. S. Shetty, X. Liang, L. Njilla, C. A. Kamhoua et K. Kwiat, "Blockcloud Security Analysis," dans *Blockchain for Distributed Systems Security*. Newark : IEEE Computer Society Press, 2019, p. 161–192.

[66] Y. Xiao, N. Zhang, J. Li, W. Lou et Y. T. Hou, "Distributed Consensus Protocols and Algorithms," dans *Blockchain for Distributed Systems Security*. Newark : IEEE Computer Society Press, 2019, p. 25–50.

[67] CSIRO, "Encrypting On-Chain Data." [En ligne]. Disponible : https://research.csiro.au/blockchainpatterns/general-patterns/data-management-patterns/encrypting-on-chain-data/

[68] E. Heilman, N. Narula, G. Tanzer, J. Lovejoy, M. Colavita, M. Virza et T. Dryja, "Cryptanalysis of Curl-P and Other Attacks on the IOTA Cryptocurrency," *IACR Transactions on Symmetric Cryptology*, p. 367–391, sept. 2020. [En ligne]. Disponible : https://tosc.iacr.org/index.php/ToSC/article/view/8707

[69] M. T. Özsu, "Distributed Database Management Systems," University of Waterloo. [En ligne]. Disponible : https://cs.uwaterloo.ca/~tozsu/courses/cs856/F05/outline.html

[70] M. Ozsu et P. Valduriez, “Distributed database systems : where are we now ?” *Computer*, vol. 24, nº. 8, p. 68–78, août 1991. [En ligne]. Disponible : http://ieeexplore.ieee.org/document/84879/

[71] Centre canadien pour la cybersécurité, “Comment protéger votre organisation contre les menaces internes,” Centre canadien pour la cybersécurité, Canada, Rapport technique ITSAP.10.003, févr. 2020. [En ligne]. Disponible : https://cyber.gc.ca/fr/orientation/comment-proteger-votre-organisation-contre-les-menaces-internes-itsap10003-0

[72] R. Grimmick, “What is an Insider Threat ? Definition and Examples | Varonis,” janv. 2021. [En ligne]. Disponible : https://www.varonis.com/blog/insider-threats/

[73] K. A. Scarfone, W. Jansen et M. Tracy, “Guide to general server security,” National Institute of Standards and Technology, Gaithersburg, MD, NIST SP 800-123, 2008, edition : 0. [En ligne]. Disponible : https://nvlpubs.nist.gov/nistpubs/Legacy/SP/nistspecialpublication800-123.pdf

[74] O. Ajayi et T. Saadawi, “Detecting Insider Attacks in Blockchain Networks,” dans *2021 International Symposium on Networks, Computers and Communications (ISNCC)*. Dubai, United Arab Emirates : IEEE, oct. 2021, p. 1–7. [En ligne]. Disponible : https://ieeexplore.ieee.org/document/9615799/

[75] M. Muzammal, Q. Qu et B. Nasrulin, “Renovating blockchain with distributed databases : An open source system,” *Future Generation Computer Systems*, vol. 90, p. 105–117, janv. 2019. [En ligne]. Disponible : https://www.sciencedirect.com/science/article/pii/S0167739X18308732

[76] W. Fan, S.-Y. Chang, S. Emery et X. Zhou, “Blockchain-based Distributed Banking for Permissioned and Accountable Financial Transaction Processing,” dans *2020 29th International Conference on Computer Communications and Networks (ICCCN)*. Honolulu, HI, USA : IEEE, août 2020, p. 1–9. [En ligne]. Disponible : https://ieeexplore.ieee.org/document/9209687/

[77] A. Fournier, F. El Khoury et S. Pierre, “A Client/Server Malware Detection Model Based on Machine Learning for Android Devices,” *IoT*, vol. 2, nº. 3, 2021.

[78] S. Keykhaie et S. Pierre, “Lightweight and Secure Face-based Active Authentication for Mobile Users,” *IEEE Transactions on Mobile Computing*, p. 1–1, 2021.

[79] ——, “A Generic Model for Privacy-Preserving Authentication on Smartphones,” dans *2021 IEEE International Systems Conference (SysCon)*, 2021, p. 1–7.

[80] L. D. Tsobdjou, S. Pierre et A. Quintero, “An Online Entropy-based DDoS Flooding Attack Detection System with Dynamic Threshold,” *IEEE Transactions on Network and Service Management*, p. 1–1, 2022. [En ligne]. Disponible : https://ieeexplore.ieee.org/document/9678955/

[81] N. Provos, “Firewall,” dans *Encyclopedia of Cryptography and Security*, H. C. A. van Tilborg et S. Jajodia, édit. Boston, MA : Springer US, 2011, p. 471–474. [En ligne]. Disponible : https://doi.org/10.1007/978-1-4419-5906-5_506

[82] P. S. Murthy et V. Nagalakshmi, “Database Forensics and Security Measures to Defend from Cyber Threats,” dans *2020 3rd International Conference on Intelligent Sustainable Systems (ICISS)*, déc. 2020, p. 1302–1307.

[83] R. Fleiner, R. Hubert, A. Bánáti et L. Erdôdi, “Security threats based on critical database system privileges,” dans *2022 IEEE 10th Jubilee International Conference on Computational Cybernetics and Cyber-Medical Systems (ICCC)*, 2022, p. 000 117–000 122.

[84] A. Mousa, M. Karabatak et T. Mustafa, “Database Security Threats and Challenges,” dans *2020 8th International Symposium on Digital Forensics and Security (ISDFS)*, 2020, p. 1–5.

[85] M. Dib, “Architecture basée sur HSM pour sécuriser les données sur un serveur d’applications mobiles,” Mémoire de maîtrise, Polytechnique Montreal, Department of Computer and Software Engineering, Montréal, QC, 2020.

[86] R. Anderson, M. Bond, J. Clulow et S. Skorobogatov, “Cryptographic Processors-A Survey,” *Proceedings of the IEEE*, vol. 94, n°. 2, p. 357–369, févr. 2006. [En ligne]. Disponible : http://ieeexplore.ieee.org/document/1580505/

[87] Entrust Corporation, “The nShield Security World architecture : Optimizing security and operational efficiency in Entrust nShield HSM environments,” Entrust Corporation, White Paper, 2023. [En ligne]. Disponible : https://images.go.entrust.com/Web/EntrustInc/%7Bed70caee-8d96-470c-b141-69401fc7ff22%7D_nCipher-Security-World-Architecture-wp.pdf?_gl=1%2Ajdfjth%2A_ga%2AMjA1MjY2NzAxNS4xNzAxODExOTUx%2A_ga_6QRW66BW5T%2AMTcwMTgxMTk1MS4xLjEuMTcwMTgxMjE1Ni4yNi4wLjA.&_ga=2.134694258.260465634.1701811954-2052667015.1701811951

[88] M. Dahmane et S. Foucher, “Combating Insider Threats by User Profiling from Activity Logging Data,” dans *2018 1st International Conference on Data Intelligence and Security (ICDIS)*. South Padre Island, TX : IEEE, avr. 2018, p. 194–199. [En ligne]. Disponible : https://ieeexplore.ieee.org/document/8367763/

[89] M. Mylrea, S. N. G. Gourisetti, C. Larimer et C. Noonan, “Insider Threat Cybersecurity Framework Webtool & Methodology : Defending Against Complex Cyber-Physical Threats,” dans *2018 IEEE Security and Privacy Workshops (SPW)*. San Francisco, CA : IEEE, mai 2018, p. 207–216. [En ligne]. Disponible : https://ieeexplore.ieee.org/document/8424652/

[90] "CIS Community Defense Model," CIS Center for Internet Security, New York, Rapport technique. [En ligne]. Disponible : http://www.cisecurity.org/controls/

[91] C. Joshi, J. R. Aliaga et D. R. Insua, "Insider Threat Modeling : An Adversarial Risk Analysis Approach," *IEEE Transactions on Information Forensics and Security*, vol. 16, p. 1131–1142, 2021.

[92] Y. M. Tukur et Y. S. Ali, "Demonstrating the Effect of Insider Attacks on Perception Layer of Internet of Things (IoT) Systems," dans *2019 15th International Conference on Electronics, Computer and Computation (ICECCO)*. Abuja, Nigeria : IEEE, déc. 2019, p. 1–6. [En ligne]. Disponible : https://ieeexplore.ieee.org/document/9043248/

[93] R. N. Kocsis, édit., *Applied criminal psychology : a guide to forensic behavioral sciences*, second edition éd. Springfield, Ill : Charles C Thomas, Publisher, Ltd, 2018.

[94] W. R. Claycomb, C. L. Huth, B. Phillips, L. Flynn et D. McIntire, "Identifying indicators of insider threats : Insider IT sabotage," dans *2013 47th International Carnahan Conference on Security Technology (ICCST)*, 2013, p. 1–5.

[95] L. Reinerman-Jones, G. Matthews, R. Wohleber et E. Ortiz, "Scenarios using situation awareness in a simulation environment for eliciting insider threat behavior," dans *2017 IEEE Conference on Cognitive and Computational Aspects of Situation Management (CogSIMA)*. Savannah, GA : IEEE, mars 2017, p. 1–3. [En ligne]. Disponible : https://ieeexplore.ieee.org/document/7929611/

[96] A. Duncan, S. Creese et M. Goldsmith, "A Combined Attack-Tree and Kill-Chain Approach to Designing Attack-Detection Strategies for Malicious Insiders in Cloud Computing," dans *2019 International Conference on Cyber Security and Protection of Digital Services (Cyber Security)*. Oxford, United Kingdom : IEEE, juin 2019, p. 1–9. [En ligne]. Disponible : https://ieeexplore.ieee.org/document/8885401/

[97] C. Egbunike et S. Rajendran, "The implementation of negative database as a security technique on a generic database system," dans *2017 International Conference on Circuit ,Power and Computing Technologies (ICCPCT)*, 2017, p. 1–8.

[98] D. Vinayagamurthy, A. Gribov et S. Gorbunov, "StealthDB : a Scalable Encrypted Database with Full SQL Query Support," *Proceedings on Privacy Enhancing Technologies*, vol. 2019, n°. 3, p. 370–388, juill. 2019. [En ligne]. Disponible : https://content.sciendo.com/doi/10.2478/popets-2019-0052

[99] B. H. Chen, P. Y. Cheung, P. Y. Cheung et Y.-K. Kwok, "CypherDB : A Novel Architecture for Outsourcing Secure Database Processing," *IEEE Transactions on Cloud Computing*, vol. 6, n°. 2, p. 372–386, avr. 2018, conference Name : IEEE Transactions on Cloud Computing.

[100] M. Okada, T. Suzuki, N. Nishio, H. Waidyasooriya et M. Hariyama, “FPGA-accelerated Searchable Encrypted Database Management Systems for Cloud Services,” *IEEE Transactions on Cloud Computing*, p. 1–1, 2020, conference Name : IEEE Transactions on Cloud Computing.

[101] L.-x. Xu, D. Sun et D. Liu, “Study on methods for data confidentiality and data integrity in relational database,” dans *2010 3rd International Conference on Computer Science and Information Technology*, vol. 1, 2010, p. 292–295.

[102] “CIS Controls Guide,” CIS Center for Internet Security, New York, Rapport technique, mai 2021. [En ligne]. Disponible : http://www.cisecurity.org/controls/

[103] S. Nathan, C. Govindarajan, A. Saraf, M. Sethi et P. Jayachandran, “Blockchain meets database : Design and implementation of a blockchain relational database,” *arXiv preprint arXiv :1903.01919*, 2019.

[104] T. McConaghy, R. Marques, A. Müller, D. De Jonghe, T. McConaghy, G. McMullen, R. Henderson, S. Bellemare et A. Granzotto, “Bigchaindb : a scalable blockchain database,” *white paper, BigChainDB*, p. 53–72, 2016.

[105] M. Castro et B. Liskov, “Practical byzantine fault tolerance and proactive recovery,” *ACM Transactions on Computer Systems*, vol. 20, nº. 4, p. 398–461, nov. 2002. [En ligne]. Disponible : https://dl.acm.org/doi/10.1145/571637.571640

[106] D. Ongaro et J. Ousterhout, “In search of an understandable consensus algorithm,” dans *2014 USENIX annual technical conference (USENIX ATC 14)*, 2014, p. 305–319.

[107] D. Georgiev, “22 Insider Threat Statistics to Look Out For in 2023,” août 2023. [En ligne]. Disponible : https://techjury.net/blog/insider-threat-statistics/

[108] ID Watchdog, “Insider Threats and Data Breaches.” [En ligne]. Disponible : https://www.idwatchdog.com/insider-threats-and-data-breaches/

[109] M. Dib et S. Pierre, “Insider Attack Model Against HSM-Based Architecture,” *IEEE Access*, vol. 11, p. 86 848–86 858, 2023. [En ligne]. Disponible : https://ieeexplore.ieee.org/document/10216991/

[110] H. Pishro-Nik, “Linear MMSE Estimation of Random Variables,” dans *Introduction to probability, statistics, and random processes*. Blue Bell, PA : Kappa Research, LLC, 2014.

[111] M. Burrows, M. Abadi et R. Needham, “A logic of authentication,” *Proceedings of the Royal Society of London. A. Mathematical and Physical Sciences*, vol. 426, nº. 1871, p. 233–271, déc. 1989. [En ligne]. Disponible : https://royalsocietypublishing.org/doi/10.1098/rspa.1989.0125

[112] A. Drapkin, “Data Breaches That Have Happened in 2022 So Far - Updated List,” nov. 2022. [En ligne]. Disponible : https://tech.co/news/data-breaches-2022-so-far

[113] The Canadian Press, “Quebec court approves $200.9M settlement against Desjardins over data breach | CBC News.” [En ligne]. Disponible : https://www.cbc.ca/news/canada/montreal/quebec-court-settlement-desjardins-data-breach-1.6493397

[114] “Our security measures.” [En ligne]. Disponible : https://www.desjardins.com/ca/security/our-security-measures/index.jsp

[115] Cisco, “Implementing Cisco Adaptive Security Appliance (ASA),” dans *CCNA Security*, 2013. [En ligne]. Disponible : http://sclabs.blogspot.com/2013/01/chapter-10-implementing-cisco-adaptive.html

[116] L. Sustek, “Hardware Security Module,” dans *Encyclopedia of Cryptography and Security*, H. C. A. van Tilborg et S. Jajodia, édit. Boston, MA : Springer US, 2011, p. 535–538. [En ligne]. Disponible : http://link.springer.com/10.1007/978-1-4419-5906-5_509

[117] P. Smirnoff, “Understanding Hardware Security Modules (HSMs).” [En ligne]. Disponible : https://www.cryptomathic.com/news-events/blog/understanding-hardware-security-modules-hsms

[118] T. Gunasekhar, K. T. Rao et M. T. Basu, “Understanding insider attack problem and scope in cloud,” dans *2015 International Conference on Circuits, Power and Computing Technologies [ICCPCT-2015]*. Nagercoil, India : IEEE, mars 2015, p. 1–6. [En ligne]. Disponible : http://ieeexplore.ieee.org/document/7159380/

[119] Canadian Centre for Cyber Security, “How to protect your organization from insider threats,” Canadian Centre for Cyber Security, Canada, Rapport technique ITSAP.10.003, févr. 2020. [En ligne]. Disponible : https://cyber.gc.ca/en/guidance/how-protect-your-organization-insider-threats-itsap10003-0

[120] M. Johnson, “What is a pull request ?” nov. 2013. [En ligne]. Disponible : http://oss-watch.ac.uk/resources/pullrequest

[121] A. Ronacher, “Flask : Web Development, one drop at a time,” Austria, avr. 2020. [En ligne]. Disponible : https://palletsprojects.com/p/flask/

[122] “PostgreSQL Database Management System,” 2020. [En ligne]. Disponible : www.postgresql.org

[123] “Docker,” 2023. [En ligne]. Disponible : https://www.docker.com/

[124] L. Luchini, “ASN.1 JavaScript Decoder,” 2023. [En ligne]. Disponible : https://lapo.it/asn1js/

[125] B. Kaliski, “PKCS #5 : Password-Based Cryptography Specification Version 2.0,” RSA Laboratories, East, RFC 2898, sept. 2000. [En ligne]. Disponible : https://www.ietf.org/rfc/rfc2898.txt

[126] ——, “PKCS #8 : Private Key Information Syntax Specification Version 1.2,” EMC, RFC 5208, mai 2008. [En ligne]. Disponible : https://datatracker.ietf.org/doc/html/rfc5208

[127] A. J. Cabrera-Gutierrez, E. Castillo, A. Escobar-Molero, J. A. Alvarez-Bermejo, D. P. Morales et L. Parrilla, “Integration of Hardware Security Modules and Permissioned Blockchain in Industrial IoT Networks,” *IEEE Access*, vol. 10, p. 114 331–114 345, 2022. [En ligne]. Disponible : https://ieeexplore.ieee.org/document/9931706/

[128] V. Costan et S. Devadas, “Intel SGX Explained,” *IACR Cryptol. ePrint Arch.*, vol. 2016, p. 86, 2016.

[129] D. G. Kendall, “Stochastic Processes Occurring in the Theory of Queues and their Analysis by the Method of the Imbedded Markov Chain,” *The Annals of Mathematical Statistics*, vol. 24, n°. 3, p. 338–354, sept. 1953. [En ligne]. Disponible : https://projecteuclid.org/journals/annals-of-mathematical-statistics/volume-24/issue-3/Stochastic-Processes-Occurring-in-the-Theory-of-Queues-and-their/10.1214/aoms/1177728975.full

[130] A. Ravindran, “Queuing Theory,” dans *Operations research methodologies*. Boca Raton : CRC Press, 2009, oCLC : 288039187. [En ligne]. Disponible : http://www.crcnetbase.com/isbn/9781420091823

[131] V. Sundarapandian, *Probability, statistics and queuing theory*. New Delhi : PHI Learning, 2009, oCLC : 739482193.

[132] D. A. Menascé, V. A. F. Almeida et L. Dowdy, *Performance by design : computer capacity planning by example*. Upper Saddle River, NJ : Prentice Hall PTR, 2004, oCLC : ocm54410490.

[133] “PKCS #11 : Cryptographic Token Interface Base Specification,” OASIS Standard, Rapport technique, mai 2016. [En ligne]. Disponible : http://docs.oasis-open.org/pkcs11/pkcs11-base/v2.40/pkcs11-base-v2.40.html

[134] C. Priebe, K. Vaswani et M. Costa, “EnclaveDB : A Secure Database Using SGX,” dans *2018 IEEE Symposium on Security and Privacy (SP)*, mai 2018, p. 264–278, iSSN : 2375-1207. [En ligne]. Disponible : https://ieeexplore.ieee.org/document/8418608/

[135] D. Caputo, M. Maloof et G. Stephens, “Detecting Insider Theft of Trade Secrets,” *IEEE Security & Privacy*, vol. 7, n°. 6, p. 14–21, nov. 2009. [En ligne]. Disponible : https://ieeexplore.ieee.org/document/5210090/

[136] S. Yuan et X. Wu, “Deep learning for insider threat detection : Review, challenges and opportunities,” *Computers & Security*, vol. 104, p. 102221, mai 2021. [En ligne]. Disponible : https://linkinghub.elsevier.com/retrieve/pii/S0167404821000456

[137] M. Li, X. Zhao, L. Chen, C. Tan, H. Li, S. Wang, Z. Mi, Y. Xia, F. Li et H. Chen, “Encrypted Databases Made Secure Yet Maintainable,” 2023, p. 117–133. [En ligne]. Disponible : https://www.usenix.org/conference/osdi23/presentation/li-mingyu

[138] K. A. Scarfone, M. P. Souppaya, A. Cody et A. D. Orebaugh, “Technical guide to information security testing and assessment.” National Institute of Standards and Technology, Gaithersburg, MD, Rapport technique NIST SP 800-115, 2008, edition : 0. [En ligne]. Disponible : https://nvlpubs.nist.gov/nistpubs/Legacy/SP/nistspecialpublication800-115.pdf

[139] R. Shirey, “Internet Security Glossary, Version 2,” RFC 4949, août 2007. [En ligne]. Disponible : https://tools.ietf.org/html/rfc4949

[140] D. D. Clark, “Window and Acknowledgement Strategy in TCP,” MIT Laboratory for Computer Science, RFC 813, juill. 1982. [En ligne]. Disponible : https://datatracker.ietf.org/doc/html/rfc813

[141] “Cisco Annual Internet Report (2018–2023),” Cisco, White Paper, mars 2020. [En ligne]. Disponible : https://www.cisco.com/c/en/us/solutions/collateral/executive-perspectives/annual-internet-report/white-paper-c11-741490.html

[142] S. Sinha, “State of IoT 2023 : Number of connected IoT devices growing 16% to 16.7 billion globally,” mai 2023. [En ligne]. Disponible : https://iot-analytics.com/number-connected-iot-devices/

[143] “Distributed relational database,” avr. 2023. [En ligne]. Disponible : https://www.ibm.com/docs/en/i/7.5?topic=concepts-distributed-relational-database

[144] Akashkumar17, “Advantages of Distributed database,” mai 2023. [En ligne]. Disponible : https://www.geeksforgeeks.org/advantages-of-distributed-database/

[145] C. Stoll, L. Klaaßen et U. Gallersdörfer, “The Carbon Footprint of Bitcoin,” *Joule*, vol. 3, n°. 7, p. 1647–1661, juill. 2019. [En ligne]. Disponible : https://linkinghub.elsevier.com/retrieve/pii/S2542435119302557

[146] “What is distributed database ? | Definition from TechTarget,” 2024. [En ligne]. Disponible : https://www.techtarget.com/searchoracle/definition/distributed-database

[147] “Distributed Database - Apache Ignite.” [En ligne]. Disponible : https://ignite.apache.org/

[148] “Couchbase Server.” [En ligne]. Disponible : https://www.couchbase.com/products/server/

[149] "AWS | Amazon SimpleDB – Simple Database Service." [En ligne]. Disponible : https://aws.amazon.com/simpledb/

[150] "Bitcoin - Open source P2P money." [En ligne]. Disponible : https://bitcoin.org/en/

[151] "Home | ethereum.org," sept. 2023. [En ligne]. Disponible : https://ethereum.org/en/

[152] Hyperledger Foundation, "Hyperledger Fabric." [En ligne]. Disponible : https://hyperledger.org/projects/fabric

[153] S. Kaur, S. Chaturvedi, A. Sharma et J. Kar, "A Research Survey on Applications of Consensus Protocols in Blockchain," *Security and Communication Networks*, vol. 2021, p. 1–22, janv. 2021. [En ligne]. Disponible : https://www.hindawi.com/journals/scn/2021/6693731/

[154] A. De Vries, "Bitcoin's growing energy problem," *Joule*, vol. 2, n°. 5, p. 801–805, 2018, publisher : Elsevier.

[155] P. Szilágyi, "EIP-255 : Clique proof-of-authority consensus protocol," Ethereum Improvement Proposals 255, mars 2017. [En ligne]. Disponible : https://eips.ethereum.org/EIPS/eip-225

[156] C. N. Samuel, S. Glock, F. Verdier et P. Guitton-Ouhamou, "Choice of Ethereum Clients for Private Blockchain : Assessment from Proof of Authority Perspective," dans *2021 IEEE International Conference on Blockchain and Cryptocurrency (ICBC)*. Sydney, Australia : IEEE, mai 2021, p. 1–5. [En ligne]. Disponible : https://ieeexplore.ieee.org/document/9461085/

[157] "What are smart contracts on blockchain ? | IBM." [En ligne]. Disponible : https://www.ibm.com/topics/smart-contracts

[158] Solidity Team, "Home | Solidity Programming Language," 2023. [En ligne]. Disponible : https://soliditylang.org/

[159] C. Smith, "Oracles," sept. 2023. [En ligne]. Disponible : https://ethereum.org/en/developers/docs/oracles/

[160] M. Burrows, M. Abadi et R. Needham, "A logic of authentication," *ACM Transactions on Computer Systems (TOCS)*, vol. 8, n°. 1, p. 18–36, 1990, publisher : ACM New York, NY, USA. [En ligne]. Disponible : https://www.cs.cmu.edu/~dga/15-712/F07/papers/Burrows90.pdf

[161] M. Dib et S. Pierre, "HSM-Based Architecture to Detect Insider Attacks on Server-Side Data," *IEEE Transactions on Information Forensics and Security (unpublished)*.

[162] T. M. Zaw, M. Thant et S. V. Bezzateev, "Database Security with AES Encryption, Elliptic Curve Encryption and Signature," dans *2019 Wave Electronics and its Application in Information and Telecommunication Systems (WECONF)*.

Saint-Petersburg, Russia : IEEE, juin 2019, p. 1–6. [En ligne]. Disponible : https://ieeexplore.ieee.org/document/8840125/