Titre: Title:	Modèles de prévention et de détection des cyberattaques dans un environnement périphérique mobile/serveur
Auteur: Author:	Loïc Tsobdjou Dongmo
Date:	2022
Type:	Mémoire ou thèse / Dissertation or Thesis
Référence: Citation:	Tsobdjou Dongmo, L. (2022). Modèles de prévention et de détection des cyberattaques dans un environnement périphérique mobile/serveur [Thèse de doctorat, Polytechnique Montréal]. PolyPublie. https://publications.polymtl.ca/10465/

URL de PolyPublie: PolyPublie URL:	https://publications.polymtl.ca/10465/
Directeurs de recherche: Advisors:	Samuel Pierre, & Alejandro Quintero
Programme: Program:	Génie informatique

POLYTECHNIQUE MONTRÉAL

affiliée à l'Université de Montréal

Modèles de prévention et de détection des cyberattaques dans un environnement périphérique mobile/serveur

LOÏC TSOBDJOU DONGMO

Département de génie informatique et génie logiciel

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

Génie informatique

Août 2022

POLYTECHNIQUE MONTRÉAL

affiliée à l'Université de Montréal

Cette thèse intitulée :

Modèles de prévention et de détection des cyberattaques dans un environnement périphérique mobile/serveur

présentée par **Loïc TSOBDJOU DONGMO**

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 et codirecteur de recherche
Soumaya CHERKAOUI, membre
Abderrezak RACHEDI, membre

DÉDICACE

À mon papa, Martin Tsobdjou, parti si tôt.

À ma maman, Sabine Tsague, pour son amour inconditionnel.

REMERCIEMENTS

J'exprime ma gratitude au Professeur Samuel Pierre, pour m'avoir donné l'occasion d'effectuer ma thèse au sein du laboratoire de recherche en réseautique et informatique mobile (LARIM). Je lui adresse mes remerciements pour sa disponibilité, son encadrement et ses conseils avisés, tout au long de mon projet doctoral. Je remercie également mon codirecteur de recherche, Professeur Alejandro Quintero, pour sa disponibilité et ses critiques constructives, ayant servi à améliorer mon travail.

Je remercie les professeurs Martine Bellaïche, Soumaya Cherkaoui et Abderrezak Rachedi pour avoir accepté d'évaluer cette thèse.

J'adresse également mes remerciements à l'endroit de Docteur Franjieh El Khoury, la coordonnatrice du LARIM, pour sa supervision, la relecture de mes différents travaux, ses conseils et son dévouement à la tâche. Par la même occasion, j'exprime ma gratitude à mes collègues du LARIM, pour la bonne ambiance et les discussions chaleureuses.

Je dis merci à la compagnie Flex Group qui m'a permis d'allier recherche académique et immersion en milieu industriel durant mes études.

Je tiens également à remercier les personnes qui me soutiennent au quotidien et me donnent la force d'avancer malgré les difficultés. Dans un premier temps, je pense à mes frères et sœurs : Harold Kenfack, Raïssa Chiozem, Carmen Ngouadjio et Cherel Tsobdjou. Dans un deuxième temps, je pense à ma belle-sœur Aristide Tematio et à mon beau-frère Docteur Frank Amabo. Dans un troisième temps, je pense à mes amis : Docteur Yves Mbeutcha, Docteur Stéphanie Kamgnia, Michèle Ngom, Marina Kossinga, Charlaine Mbounga, Danièle Weulassagou et Miranda Tsague.

Merci à Bertin Nankeng pour m'avoir aidé à monter mon projet d'études au Canada et pour son soutien multiforme. J'adresse aussi mes remerciements à ceux et celles qui m'ont accueilli au Canada et ont facilité mon intégration. Une pensée particulière à Patricia Akiobe, Thierry Bayi et Géraldine Nguekeng.

À toutes ces personnes que je côtoie au quotidien et qui, d'une façon ou d'une autre, sont des sources d'inspiration, je vous dis merci.

RÉSUMÉ

De nos jours, nous observons une prolifération des périphériques mobiles, tels que les téléphones intelligents, les tablettes, les montres connectées et les capteurs corporels.

D'une part, certains de ces périphériques mobiles sont caractérisés par des ressources limitées, dont la capacité de calcul, la mémoire et la durée de vie de la batterie. Pour réduire la consommation des ressources des périphériques mobiles, le paradigme *« Mobile Cloud Computing » (MCC)* a vu le jour. Il consiste à transférer l'exécution des opérations gourmandes en ressources vers le nuage doté de ressources quasi illimitées.

D'autre part, la plupart des services destinés aux périphériques mobiles, à l'instar de la banque mobile, le paiement mobile et la télémédecine, requiert une communication sécurisée entre les périphériques mobiles et les serveurs distants, dans un modèle client mobile/serveur.

La communication entre les périphériques mobiles et les serveurs distants à travers des canaux de communication non sécurisés, soulève le problème de sécurité des données, en termes de confidentialité, d'intégrité et de disponibilité. Par conséquent, des mécanismes de sécurité sont nécessaires pour protéger les données des utilisateurs, tout en considérant les contraintes de ressources liées à l'usage de périphériques mobiles.

Au regard de tout ce qui précède, l'objectif principal de cette thèse est de concevoir des modèles robustes pour améliorer la sécurité des communications dans un environnement client mobile/serveur. La revue de littérature effectuée nous a permis d'identifier les menaces de sécurité dans un environnement client mobile/serveur. Ces menaces de sécurité sont réparties en trois catégories, en fonction de leurs zones d'action, à savoir le périphérique mobile, l'infrastructure réseau et le serveur. Dans le même ordre d'idées, et en vue d'atteindre l'objectif principal de cette thèse, nous avons divisé le travail en trois volets.

Le premier volet traite une menace de sécurité qui opère au niveau de l'infrastructure réseau, plus précisément l'attaque de l'homme du milieu. Dans cette attaque, un cybercriminel peut, soit usurper l'identité du périphérique mobile ou du serveur, soit intercepter et/ou modifier les messages échangés entre le périphérique mobile et le serveur. La solution proposée à cette menace est un protocole d'authentification mutuelle et d'échange de clé entre un périphérique mobile et un serveur. Ce protocole s'appuie sur la cryptographie sur les courbes elliptiques afin d'être à la fois

sécuritaire et léger. De plus, l'adoption de la carte *« Subscriber Identity Module » (SIM)* permet de stocker les données secrètes et d'exécuter les opérations cryptographiques de façon sécuritaire. L'évaluation de la sécurité est faite de façon informelle et formelle, via la logique *« Burrows-Abadi-Needham » (BAN)* et le logiciel *« Scyther »*. Cette évaluation considère les capacités d'un adversaire, qui peut par exemple intercepter, supprimer, modifier ou renvoyer les messages qui transitent sur le canal de communication. Il en ressort que, le protocole proposé possède des propriétés de sécurité, telles que la résistance à l'attaque de l'homme du milieu, la résistance à l'usurpation d'identité, la résistance à la perte ou au vol de la carte *SIM*, la confidentialité persistante et l'anonymat. L'évaluation des performances, à travers le *coût de calcul*, le *coût de mémoire* et le *coût de communication*, montre que le protocole proposé consomme peu de ressources. Par conséquent, il est adapté pour les périphériques mobiles.

Le deuxième volet de cette thèse aborde la menace de déni de service qui agit au niveau du serveur. La conséquence de cette menace est l'indisponibilité du serveur pour répondre aux requêtes des périphériques mobiles. Pour faire face à cette menace, nous proposons un système de détection des attaques de déni de service distribué *« Distributed Denial of Service » (DDoS)* par inondation. Dans ce type d'attaque, plusieurs périphériques malveillants envoient une grande quantité de trafic à haut débit au serveur, afin de réduire sa capacité à répondre aux requêtes des périphériques légitimes, voire le rendre indisponible. Le système de détection proposé utilise une approche basée sur l'anomalie et les statistiques, plus précisément l'entropie de l'information. L'entropie permet de détecter une attaque en utilisant peu de caractéristiques du trafic réseau. Dans le système proposé, nous utilisons uniquement l'*adresse « Internet Protocol » (IP) source* contenue dans l'entête des paquets du trafic réseau. Une attaque est détectée lorsque l'entropie est en dessous d'un certain seuil. Nous proposons également un algorithme qui permet de fixer le seuil de manière dynamique, en s'adaptant aux changements observés dans le trafic réseau légitime. Une première évaluation des performances du système de détection proposé est faite à l'aide des simulations sur le simulateur réseau *« Graphical Network Simulator-3 » (GNS3)*. Une deuxième évaluation des performances est faite à l'aide de l'ensemble de données public *CICIDS2017* qui contient à la fois le trafic réseau légitime et le trafic d'attaque *DDoS*. Les résultats montrent un taux de détection de 100 %, un taux de faux positifs de 0 % et un délai de détection de 7,89 secondes.

Le troisième volet a pour objet la menace de vulnérabilité des applications mobiles qui intervient au niveau du périphérique mobile. Cette menace met en péril la confidentialité et l'intégrité des

données des utilisateurs, ainsi que le respect de leur vie privée. Dans ce volet, nous nous focalisons sur les applications de banque mobile et sur le système d'exploitation mobile Android. La banque mobile est un moyen de plus en plus utilisé pour accéder aux services bancaires et aux données financières très sensibles. Android est le leader mondial du marché des systèmes d'exploitation mobiles. La solution proposée est un cadre *« Framework »* pour l'évaluation de la sécurité des applications Android de banque mobile. Ce cadre est un moyen pour les chercheurs et les banques d'évaluer la sécurité des applications de banque mobile, afin de limiter les vulnérabilités qu'elles peuvent présenter. Le cadre proposé est un ensemble de vingt-six critères divisés en cinq catégories : *sécurité du périphérique mobile*, *données en transit*, *stockage des données*, *mauvaise utilisation cryptographique* et *autres*. L'évaluation du cadre proposé montre qu'il satisfait aux requis de *non-redondance*, de *non-ambiguïté* et de *complétude*. Comme étude de cas, nous avons évalué la sécurité de sept applications Android de banque mobile du marché canadien, sur la base des critères de la catégorie *données en transit*. Le résultat montre que les applications évaluées protègent adéquatement les données lors de la communication avec les serveurs distants. Ce résultat peut motiver les usagers à faire confiance aux applications de banque mobile du marché canadien.

En résumé, les différents travaux réalisés dans le cadre de cette thèse, contribuent à améliorer la sécurité des communications entre un périphérique mobile et un serveur distant.

ABSTRACT

Nowadays, we are witnessing a rapid increase in numbers of mobile devices, such as smartphones, tablets, smartwatches and body sensors.

On one hand, some of these mobile devices are characterized by limited resources, including computing capacity, memory and battery life. To reduce the consumption of mobile device resources, the *Mobile Cloud Computing (MCC)* paradigm has emerged. It consists of transferring the execution of resource-intensive operations to the cloud endowed with almost unlimited resources.

On the other hand, most services for mobile devices, such as mobile banking, mobile payment and telemedicine, require secure communication between mobile devices and remote servers, in a mobile client/server model.

The communication between mobile devices and remote servers through unsecured communication channels raises the issue of data security, in terms of confidentiality, integrity and availability. Therefore, security mechanisms are needed to protect users' data, while considering the resource constraints associated with the use of mobile devices.

To this end, the main objective of this thesis is to design robust models to improve communication security in a mobile client/server environment. The performed literature review allowed us to identify security threats in a mobile client/server environment. These security threats are divided into three categories, depending on their areas of operation, namely the mobile device, the network infrastructure and the server. In the same vein, to achieve the main objective of this thesis, we have divided the work into three parts.

The first part deals with a security threat that operates at the network infrastructure level, specifically the man-in-the-middle attack. In this attack, a cybercriminal can either impersonate the mobile device or server, or intercept and/or modify the messages exchanged between the mobile device and the server. The proposed solution to this threat is a mutual authentication and key exchange protocol between a mobile device and a server. This protocol relies on elliptic curve cryptography to be both secure and lightweight. In addition, the adoption of the *Subscriber Identity Module (SIM)* card makes it possible to store secret data and execute cryptographic operations in a secure manner. The security evaluation of the proposed protocol is done informally and formally,

via *Burrows-Abadi-Needham (BAN)* logic and *Scyther* software. This evaluation considers the capabilities of an adversary, who can, for example, intercept, delete, modify or resend messages that pass through the communication channel. The results show that the proposed protocol has security properties, such as resistance to a man-in-the-middle attack, resistance to impersonation, resistance to SIM card loss or theft, perfect forward secrecy and anonymity. The performance evaluation, through *computation cost*, *memory cost* and *communication cost*, shows that the proposed protocol consumes few resources. Therefore, it is suitable for mobile devices.

The second part of this thesis addresses the denial-of-service threat that acts at the server level. The consequence of this threat is the unavailability of the server to respond to requests from mobile devices. To address this threat, we propose a *Distributed Denial of Service (DDoS)* flooding attack detection system. In this type of attack, several malicious devices send a large amount of high-rate traffic to the server, reducing its ability to respond to requests from legitimate devices or even making it unavailable. The proposed detection system uses an anomaly and statistics-based approach, specifically information entropy. Entropy makes it possible to detect an attack using few network traffic characteristics. In the proposed system, we only use the *source Internet Protocol (IP) address* contained in the header of the network traffic packets. An attack is detected when the entropy is below a certain threshold. Moreover, we propose an algorithm to set the threshold dynamically, adapting to the observed changes in legitimate network traffic. A first performance evaluation of the proposed detection system is done using simulations on the network simulator *Graphical Network Simulator-3 (GNS3)*. A second performance evaluation is done using the *CICIDS2017* public dataset, which contains both legitimate network traffic and *DDoS* attack traffic. The results show a 100% detection rate, a 0% false positive rate and a detection time of 7.89 seconds.

The third part focuses on the mobile applications vulnerability threat that occurs at the mobile device level. This threat jeopardizes the confidentiality and integrity of users' data, as well as their privacy. In this part, we focus on mobile banking applications and the Android mobile operating system. Mobile banking is an increasingly used means of accessing banking services and it handles highly sensitive financial data. Android is the global market leader in mobile operating systems. The proposed solution is a framework for security assessment of Android mobile banking applications. This framework is a mean for researchers and banks to assess the security of mobile banking applications, in order to limit the vulnerabilities they may present. The proposed

framework is a set of twenty-six criteria divided into five categories: *mobile device security*, *data in transit*, *data storage*, *cryptographic misuse* and *others*. The evaluation of the proposed framework shows that it satisfies the requirements of *no redundancy*, *no ambiguity* and *comprehensiveness*. As a case study, we evaluate the security of seven Android mobile banking applications from the Canadian market, based on the criteria of the *data in transit* category. The result shows that the reviewed applications adequately protect data when communicating with remote servers. This result may motivate users to trust mobile banking applications in the Canadian market.

In summary, the various pieces of work conducted in this thesis contribute to improving the security of communications between a mobile device and a remote server.

TABLE DES MATIÈRES

LISTE DES TABLEAUX

LISTE DES FIGURES

LISTE DES SIGLES ET ABRÉVIATIONS

AES	Advanced Encryption Standard
AH	Authentication Header
API	Application Programming Interface
BAN	Burrows-Abadi-Needham
CA	Certificate Authority
CBC	Cipher Block Chaining
CPU	Central Processing Unit
DDoS	Distributed Denial of Service
DFS	Digital Financial Services
DH	Diffie-Hellman
DNS	Domain Name System
ECB	Electronic Code Book
ECC	Elliptic Curve Cryptography
ECDHP	Elliptic Curve Diffie-Hellman Problem
ECDLP	Elliptic Curve Discrete Logarithm Problem
ECIES	Elliptic Curve Integrated Encryption Scheme
eSIM	embedded Subscriber Identity Module
ESP	Encapsulating Security Payload
FE	Flash Events
FTP	File Transfer Protocol
GNS3	Graphical Network Simulator-3
HCE	Host Card Emulation
HTTP	HyperText Transfer Protocol

HTTPS	HyperText Transfer Protocol Secure
ICC	Inter Component Communication
ICMP	Internet Control Message Protocol
IETF	Internet Engineering Task Force
IKE	Internet Key Exchange
IP	Internet Protocol
IPsec	Internet Protocol Security
IV	Initialization Vector
KOAD	Kernel-based Online Anomaly Detection
LAN	Local Area Network
LOIC	Low Orbit Ion Canon
MAC	Message Authentication Code
MCC	Mobile Cloud Computing
NFC	Near-Field Communication
NIST	National Institute of Standards and Technology
NS-2	Network Simulator 2
OSI	Open Systems Interconnection
OTP	One-Time Password
OWASP	Open Web Application Security Project
PBE	Password-Based Encryption
PKI	Public Key Infrastructure
QoS	Quality of Service
QR	Quick Response
RSA	Rivest, Shamir, Adleman

SCG	Smart Card Generator
SDN	Software-Defined Networking
SHA-256	Secure Hash Algorithm – 256
SIM	Subscriber Identity Module
SMS	Short Message Service
SPDL	Security Protocol Description Language
SQL	Structured Query Language
SSH	Secure Shell
SSL	Secure Sockets Layer
SVM	Support Vector Machine
TCP	Transmission Control Protocol
TLS	Transport Layer Security
TTP	Trusted Third Party
UDP	User Datagram Protocol
UI	User Interface
USIM	Universal Subscriber Identity Module
XOR	eXclusive OR

LISTE DES ANNEXES

CHAPITRE 1 INTRODUCTION

Les périphériques mobiles, tels que les téléphones intelligents, les tablettes et les montres connectées, font désormais partie de notre quotidien. À titre d'illustration, Statistique Canada révèle qu'en 2018, 88 % des Canadiens possédaient un téléphone intelligent [1]. En outre, Cisco prévoit que le nombre de périphériques mobiles dans le monde passera de 8,8 milliards en 2018 à 13,1 milliards d'ici 2023 [2]. Ces périphériques mobiles permettent aux utilisateurs d'avoir accès à divers services, tels que la télémédecine, le paiement mobile et la banque mobile. Ceci est rendu possible grâce aux avancées technologiques dans le domaine des réseaux de communication mobiles. Ces avancées sont entre autres la gestion de la mobilité de façon transparente pour les utilisateurs et la réduction de la latence avec l'avènement des réseaux de quatrième et de cinquième génération.

Certains services auxquels les utilisateurs ont accès via les périphériques mobiles, nécessitent la communication entre le périphérique mobile et un serveur distant dans un modèle client/serveur. Dès lors, la communication peut se faire à travers des réseaux étendus comme Internet.

La prolifération des périphériques mobiles, combinée à la vulgarisation d'Internet facilite la génération, l'accès, l'échange et le stockage de l'information. En revanche, cette facilité donne lieu à des défis de sécurité, d'autant plus que des informations sensibles, telles que les informations financières et les dossiers médicaux des utilisateurs peuvent être mises en jeu. En février 2022, le service de santé aux États-Unis *ARcare* a subi un accès non autorisé aux informations personnelles et médicales des individus [3].

Les mécanismes de sécurité sur Internet ne sont pas toujours adaptés pour des périphériques mobiles. Ceux-ci nécessitent des méthodes qui utilisent peu de ressources. Il est donc nécessaire de proposer de nouvelles solutions appropriées aux périphériques mobiles, qui ont par exemple une faible durée de vie de batterie. De plus, ces solutions doivent garantir la confidentialité et l'intégrité des données échangées entre le périphérique mobile et le serveur, la disponibilité du service et le respect de la vie privée des utilisateurs. Pour faire face à ces défis, les travaux de cette thèse se focalisent sur la problématique de sécurité dans un environnement client mobile/serveur.

Ce chapitre introductif définit les concepts de base que sont le *modèle client/serveur*, le *client mobile*, la *sécurité* et la *cryptographie*. Par la suite, nous présentons les éléments de problématique

en rapport avec la sécurité dans un environnement client mobile/serveur. Ensuite, nous énonçons les objectifs de recherche qui découlent de ces éléments de problématique. Par la suite, les principales contributions de cette thèse et leur originalité sont mises en exergue. Nous terminons ce chapitre par la présentation d'un aperçu sommaire des chapitres subséquents.

1.1 Définitions et concepts de base

Cette section est consacrée aux définitions et aux concepts de base permettant de situer notre travail dans un cadre théorique. Nous présentons d'abord le *modèle client/serveur* et le *client mobile* qui sont utilisés tout au long de cette thèse. Ensuite, nous introduisons les termes clés liés à la *sécurité des données*. Enfin, nous abordons des notions de *cryptographie* qui permettent d'assurer la sécurité des données. Plus précisément, nous définissons le chiffrement symétrique, le chiffrement à clé publique, la signature numérique, le certificat numérique, la cryptographie sur les courbes elliptiques et la fonction de hachage.

1.1.1 Modèle client/serveur

En informatique, un modèle client/serveur désigne une approche dans laquelle une entité appelée client, fait appel aux services d'une autre entité appelée serveur [4]. Les entités peuvent être aussi bien des ordinateurs que des applications qui s'exécutent sur ces ordinateurs.

La Figure 1.1 représente un exemple de modèle client/serveur [5]. Les clients peuvent être des ordinateurs de bureau, des ordinateurs portables ou des téléphones intelligents. Les serveurs peuvent être des serveurs Web qui fournissent des services Web ou des serveurs de base de données qui fournissent des services d'accès aux bases de données. Un client peut faire appel aux services de plusieurs serveurs simultanément. De même, un serveur peut répondre aux requêtes de plusieurs clients simultanément. Les clients et les serveurs communiquent entre eux à travers un réseau qui peut être soit local ou étendu.

Au début de la communication, le client est en mode actif, c'est-à-dire qu'il initie la communication en envoyant une requête au serveur. Cependant, le serveur est en mode passif, c'est-à-dire qu'il est en attente d'une requête provenant du client. Des exemples de services qu'un serveur peut fournir sont : le partage de fichiers, le partage d'imprimantes, l'accès aux bases de données et d'autres [5].

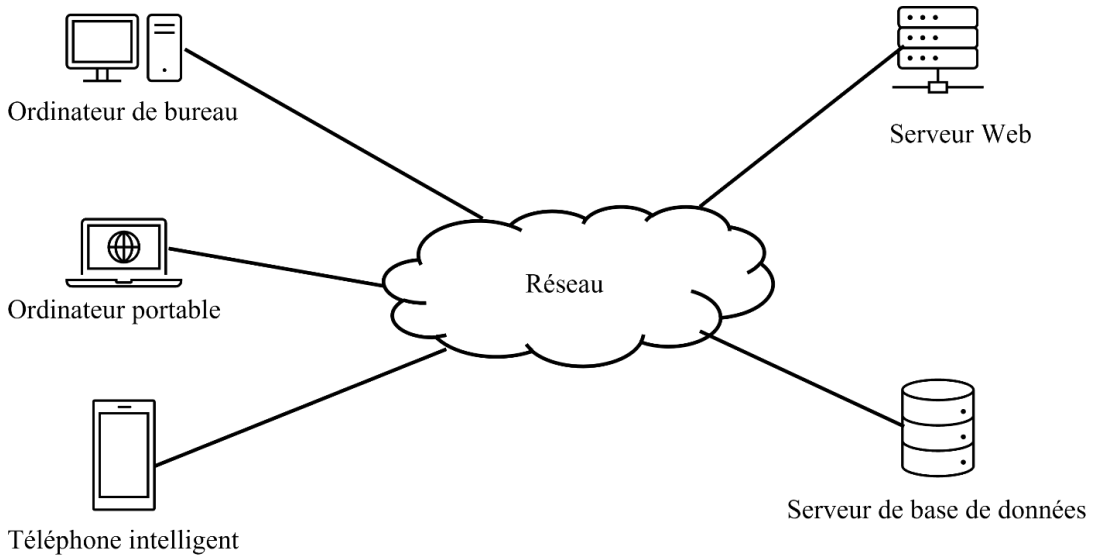

Figure 1.1 Modèle client/serveur

1.1.2 Client mobile

Dans les réseaux informatiques, la mobilité fait référence, à la possibilité pour l'usager d'avoir accès aux différents services indépendamment de sa localisation [6]. Ces services incluent la possibilité de recevoir du courrier électronique ou d'avoir accès à une page Web à bord d'une voiture [6].

Un périphérique mobile est un appareil électronique facilement transportable, qui est capable d'accéder aux services de réseau à partir de n'importe quelle position. Comme exemple, nous avons les téléphones intelligents, les tablettes et les montres connectées.

Les périphériques mobiles se caractérisent par une durée de vie de batterie relativement faible, une capacité de calcul et un espace mémoire limités pour certaines applications gourmandes en ressources [7] [8].

Dans cette thèse, le concept de *client mobile* désigne un périphérique mobile qui joue le rôle de client dans un modèle client/serveur.

1.1.3 Sécurité

Les réseaux informatiques en général et Internet en particulier favorisent l'accès, l'échange et le stockage de l'information. Cependant, cette information est exposée à plusieurs risques, tels que l'accès non autorisé, l'altération, la suppression et l'indisponibilité [9], d'où l'importance de la sécurité.

La *sécurité* désigne la protection de l'information. Les objectifs fondamentaux des solutions de sécurité sont la confidentialité, l'intégrité et la disponibilité [10]. La confidentialité garantit que l'information ne peut être accessible qu'aux personnes autorisées. Elle est intimement liée au service d'authentification qui consiste à vérifier l'identité d'une entité. L'intégrité quant à elle, garantit que l'information n'a pas été altérée. La disponibilité à son tour, assure l'accès à l'information aux personnes autorisées.

1.1.4 Notions de cryptographie

La *cryptographie* renvoie à un ensemble de méthodes dont le but est de garantir la confidentialité des informations en utilisant le chiffrement [11]. Elle peut également être utile dans d'autres aspects de la sécurité comme l'intégrité et l'authentification [9]. Le chiffrement consiste à assurer la confidentialité d'une information en la rendant inintelligible aux entités qui ne possèdent pas la clé de déchiffrement.

La Figure 1.2 montre un modèle de système de chiffrement servant à garantir la confidentialité du message lors d'une communication entre les entités A et B [9]. Ainsi, un pirate qui intercepte le message, ne peut pas déterminer son contenu. Le message clair est le message brut qui doit être protégé au cours de la communication, tandis que le message chiffré est la représentation inintelligible du message clair. L'algorithme de chiffrement permet de passer du message clair au message chiffré, tandis que celui de déchiffrement effectue le chemin inverse, c'est-à-dire du message chiffré au message clair. La clé de chiffrement est l'information qui permet à l'algorithme de chiffrement de transformer le message clair en message chiffré. De même, la clé de déchiffrement est l'information qui permet à l'algorithme de déchiffrement de transformer le message chiffré en message clair.

Il existe deux types de chiffrement : le chiffrement symétrique et le chiffrement asymétrique appelé chiffrement à clé publique.

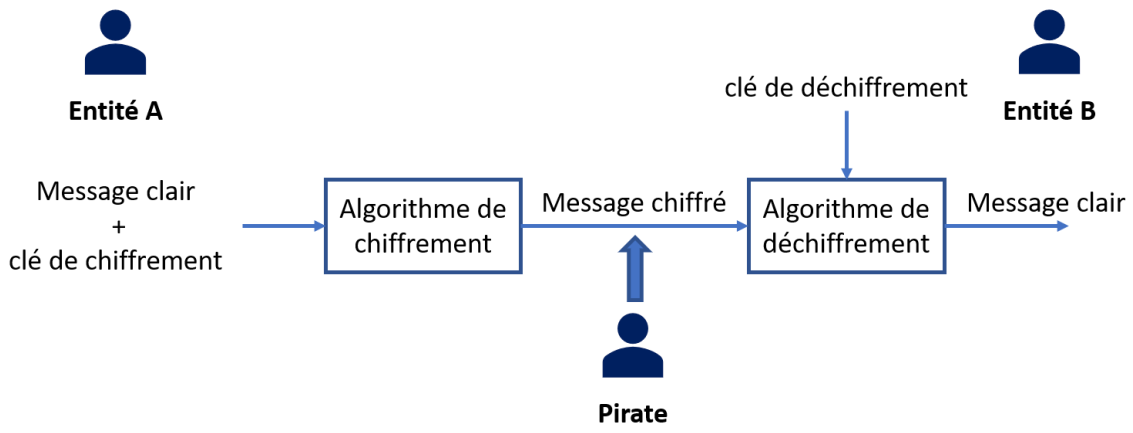

Figure 1.2 Modèle de système de chiffrement

1.1.4.1 Chiffrement symétrique

Dans le chiffrement symétrique [11], la clé de chiffrement est identique à celle de déchiffrement. Cependant, la clé doit rester secrète entre les entités de la communication. La taille de la clé est un paramètre crucial pour la sécurité du système de chiffrement. En effet, le pirate a accès à l'algorithme de déchiffrement et au message chiffré. Il ne lui manque plus que la clé de déchiffrement pour pouvoir retrouver le message clair. De ce fait, la clé doit être suffisamment grande pour qu'elle ne puisse être vulnérable à l'attaque par force brute. Cette attaque consiste à essayer toutes les valeurs possibles de la clé, dans le but de retrouver celle qui déchiffre correctement le message chiffré. Pour une clé de taille n bits, nous avons 2^n valeurs possibles. Le *« National Institute of Standards and Technology » (NIST)* recommande l'utilisation d'une clé de taille minimale de 128 bits, pour l'algorithme de chiffrement *« Advanced Encryption Standard » (AES)* [12].

Le problème avec le chiffrement symétrique est le partage de la clé entre les entités de la communication. Étant donné que la clé de chiffrement est identique à la clé de déchiffrement, comment ces entités peuvent-elles échanger la clé de chiffrement sur un canal non sécurisé?

1.1.4.2 Chiffrement à clé publique

Afin de résoudre le problème d'échange de clé dans le chiffrement symétrique, les chercheurs Diffie et Hellman ont proposé en 1976 un protocole d'échange de clé sur un canal non sécurisé

[13]. Ce protocole est à la base de la cryptographie à clé publique et utilise les propriétés mathématiques de la fonction exponentielle et son inverse, le logarithme discret [11].

Dans le chiffrement à clé publique, la clé de chiffrement est différente de la clé de déchiffrement, bien que les deux soient liées par une relation [11]. Cependant, cette relation ne permet pas de retrouver la clé de déchiffrement à partir de la clé de chiffrement. La clé de chiffrement est appelée clé publique et la clé de déchiffrement est appelée clé privée. Chaque entité de la communication possède une paire de clés publique et privée.

En considérant le schéma de la Figure 1.2, l'entité A va chiffrer le message clair avec la clé publique de l'entité B. Cette dernière, pourra par la suite, déchiffrer le message chiffré avec sa clé privée qu'elle doit garder confidentielle.

Le chiffrement à clé publique tel que défini soulève deux interrogations [11] :

1. L'entité B n'a aucune certitude sur l'identité de l'émetteur du message chiffré qu'elle reçoit. En effet, le fait que la clé publique soit connue de tous, donne la possibilité à n'importe qui de chiffrer un message et l'envoyer à l'entité B.
2. L'entité A n'a aucune garantie que la clé publique qu'elle utilise pour chiffrer le message est bien celle de l'entité B. En effet, un pirate peut substituer la clé publique de l'entité B par la sienne, ce qui lui permettra par la suite de déchiffrer les messages destinés à l'entité B en utilisant sa clé privée.

1.1.4.3 Signature numérique

La signature numérique apporte une solution au problème numéro 1 énoncé à la section précédente. Elle permet d'apporter une preuve que l'émetteur du message est bien celui qu'il prétend être : c'est l'authentification [11].

Un système de signature numérique est défini par deux fonctions [11] : une fonction de création de signature d'un message et une fonction de vérification de la signature d'un message.

- La fonction de création permet de générer la signature du message. Elle prend en entrée la clé privée du signataire ainsi que le message et retourne en sortie la signature. Ainsi, avant d'envoyer le message à l'entité B, l'entité A va le signer avec sa clé privée et envoyer la signature obtenue et le message à l'entité B.

- La fonction de vérification permet de s'assurer que le message a été signé par la bonne entité. Elle prend en entrée le message, la signature ainsi que la clé publique du signataire et retourne en sortie *vrai* si la signature correspond au message et à la clé publique fournis; et *faux* dans le cas contraire. De ce fait, à la réception du message et de la signature en provenance de l'entité A, l'entité B va utiliser la clé publique de l'entité A, afin de vérifier si le message a bien été signé par l'entité A.

Un système de signature numérique doit avoir trois propriétés de base à savoir [9] : la facilité à créer la signature d'un message, la facilité à vérifier la signature d'un message et la difficulté à falsifier la signature d'un message.

En plus de l'authentification, la signature numérique garantit également le principe de non-répudiation dans la mesure où, le signataire signe un message avec sa clé privée, qu'il est le seul à connaître [11].

1.1.4.4 Certificat numérique

Le certificat numérique apporte une solution au problème 2 énoncé à la section 1.1.4.2. Considérons le scénario suivant inspiré de [11] :

- Lorsque l'entité B envoie sa clé publique à l'entité A, le pirate l'intercepte et la remplace par sa propre clé publique;
- Lorsque l'entité A envoie sa clé publique à l'entité B, le pirate l'intercepte également et la remplace par sa clé publique.

Afin d'envoyer un message à l'entité B, l'entité A va chiffrer le message avec la clé publique du pirate, qu'elle croit être la clé publique de l'entité B. Le pirate peut ainsi intercepter le message chiffré et le déchiffrer avec sa clé privée. De plus, il peut chiffrer le message clair avec la clé publique de l'entité B et lui transmettre le message chiffré. L'entité B peut ainsi déchiffrer le message chiffré avec sa clé privée et retrouver le message clair. Cependant, elle ignore totalement que le message a été lu par un pirate.

De même, l'entité A peut signer un message avec sa clé privée. Le pirate peut intercepter le message, le modifier et le signer avec sa propre clé privée. L'entité B qui reçoit le message, va vérifier la signature avec la clé publique du pirate, qu'elle croit être la clé publique de l'entité A. Ainsi, la vérification de la signature sera valide.

Comme conséquences, la confidentialité et l'intégrité du message ont été compromises.

Pour éviter tout cela, il est important d'utiliser un certificat numérique qui permet d'associer une entité et sa clé publique. Pour qu'un certificat soit accepté ou pris en considération, il doit être signé par une autorité de confiance appelée autorité de certification [11].

Les certificats sont gérés par une infrastructure à clé publique « *Public Key Infrastructure » (PKI)*. Les utilisateurs d'une PKI font confiance aux certificats signés par l'autorité de certification. La PKI contient également une base de tous les certificats signés par son autorité de certification et maintient une liste des certificats révoqués.

1.1.4.5 Cryptographie sur les courbes elliptiques

La *cryptographie sur les courbes elliptiques « Elliptic Curve Cryptography » (ECC)* est un cas particulier de cryptographie à clé publique, qui utilise des courbes elliptiques définies sur un corps fini $\mathbb{F}$ [14] [15].

Une courbe elliptique E sur un corps $\mathbb{F}$ est l'ensemble des points de coordonnées $(x, y) \in \mathbb{F} * \mathbb{F}$ vérifiant l'équation ci-dessous [14] [15] :

$$y^2 = x^3 + ax + b, \text{ avec } a, b \in \mathbb{F} \text{ et } 4a^3 + 27b^2 \neq 0.$$

À ces points, se rajoute un point dit *à l'infini* noté O.

La Figure 1.3 représente la courbe elliptique d'équation $y^2 = x^3 - 15x + 18$ sur le corps des nombre réels $\mathbb{R}$.

Une relation d'addition notée $+$ est définie sur les points de E. Elle se décrit comme suit :

Soient $P, Q \in E$ deux points de la courbe et (D) la droite passant par les points P et Q. La droite (D) rencontre la courbe E en un troisième point T. Soit R le symétrique de T par rapport à l'axe des abscisses.

Nous avons : $P + Q = R$.

Dans cette relation d'addition, il existe deux cas particuliers :

- Si $P = Q$, alors la droite (D) est la tangente à la courbe E au point P;
- Si Q est le symétrique de P par rapport à l'axe des abscisses, alors la droite (D) rencontre la courbe E à l'infini et nous avons : $P + Q = O$.

La relation d'addition ainsi définie sur les points de E possède les propriétés suivantes :

- Commutativité : $P + Q = Q + P$, pour tout $P, Q \in E$;
- Associativité : $(P + Q) + R = P + (Q + R)$, pour tout $P, Q, R \in E$;
- Identité : $P + O = O + P = P$, pour tout $P \in E$;
- Inverse : $P + (-P) = O$, pour tout $P \in E$;
- Fermeture : $P + Q = R \in E$, pour tout $P, Q \in E$.

L'ensemble E muni de la relation d'addition $(E, +)$ forme un groupe abélien.

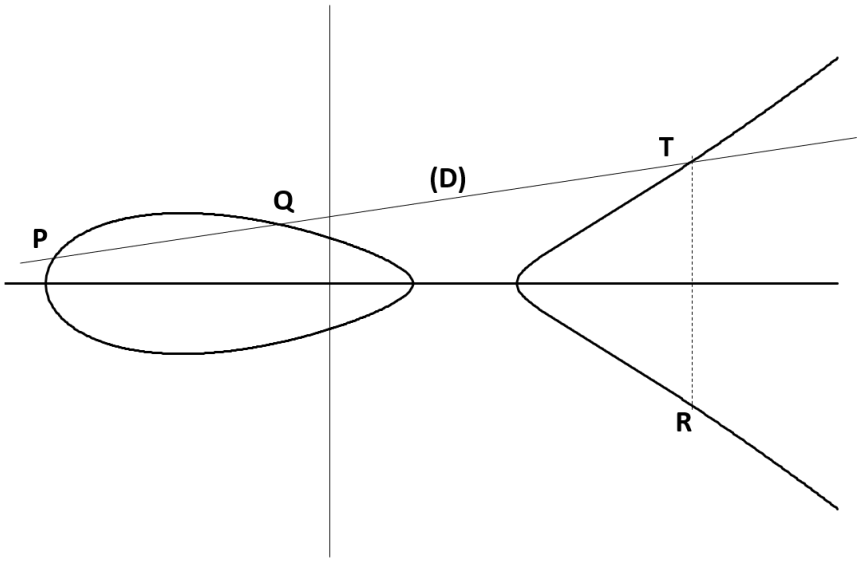

Figure 1.3 Courbe elliptique d'équation $y^2 = x^3 - 15x + 18$ sur $\mathbb{R}$

Pour des applications cryptographiques, les courbes utilisées sont des courbes elliptiques définies sur un corps fini $\mathbb{F}_p$, où $\mathbb{F}_p$ est l'ensemble des entiers modulo p, dont p représente un grand nombre premier.

La sécurité de l'ECC repose sur la difficulté du *problème du logarithme discret sur les courbes elliptiques « Elliptic Curve Discrete Logarithm Problem » (ECDLP)* [14] [15]. Ce problème est défini comme suit :

Soient E une courbe elliptique sur le corps fini $\mathbb{F}_p$ et P, Q deux points de E. Le problème du logarithme discret sur la courbe elliptique consiste à retrouver un entier n, tel que $Q = nP$.

En considérant le modèle de la Figure 1.2, l'entité A choisit de façon aléatoire un entier n qu'elle considère comme sa clé privée. Elle calcule ensuite $Q = nP$, qui est sa clé publique. Du fait de la difficulté du ECDLP, il est impossible de retrouver la clé privée n à partir de P et Q.

1.1.4.6 Fonction de hachage

Une fonction de hachage est une fonction mathématique qui prend en entrée une donnée de taille quelconque et fournit en sortie une donnée de taille fixe, appelée *haché* ou *empreinte* [11]. Une fonction de hachage cryptographique possède deux propriétés pratiques importantes et trois propriétés de sécurité [9].

- Propriété pratique 1 : compression. La fonction de hachage génère une donnée de taille fixe à partir d'une donnée de taille arbitraire. En général, la taille de la donnée passée en entrée est beaucoup plus grande que la taille de la donnée en sortie, d'où la propriété de compression.
- Propriété pratique 2 : facile à calculer. La fonction de hachage doit être facile à calculer en termes d'efficacité et de vitesse. En se basant sur la théorie de la complexité des algorithmes, une fonction de hachage doit être calculée en un temps polynomial [9].
- Propriété de sécurité 1 : résistance à la préimage. Cela signifie qu'il doit être difficile de calculer l'inverse d'une fonction de hachage. Étant donné une fonction de hachage h et un haché z, il doit être difficile de trouver une valeur x, telle que $h(x) = z$. La fonction de hachage est dite fonction à sens unique.
- Propriété de sécurité 2 : résistance à la seconde préimage. Étant donné une entrée x et son haché $z = h(x)$, il doit être difficile de trouver une autre entrée y, telle que $h(y) = h(x) = z$.
- Propriété de sécurité 3 : résistance à la collision. Cela signifie qu'il doit être difficile de trouver deux valeurs différentes ayant le même haché. Étant donné une fonction de hachage h, il est difficile de trouver deux valeurs différentes x et y ($x \neq y$), telles que $h(x) = h(y)$.

1.2 Éléments de la problématique

Avec la vulgarisation des technologies de l'information et d'Internet, les utilisateurs de périphériques mobiles ressentent le besoin d'accéder en tout temps et en tout lieu via leurs

périphériques, à des services évolués, tels que la télémédecine, le paiement mobile et la banque mobile. Ces services nécessitent pour la plupart, une communication entre les périphériques mobiles et des serveurs distants dans un modèle client/serveur. De plus, les périphériques mobiles se caractérisent par une durée de vie de batterie relativement faible, une capacité de calcul et un espace mémoire limités pour des applications gourmandes en ressources [7] [8]. Une nouvelle tendance est de transférer l'exécution d'une partie ou la totalité des applications gourmandes en ressources vers des serveurs distants ayant des ressources quasi illimitées. Dans le modèle client/serveur, l'échange des données des utilisateurs entre le périphérique mobile et le serveur à travers des réseaux sans fil pose un enjeu fondamental en ce qui concerne la sécurité des données, en termes de confidentialité, d'intégrité, de disponibilité et de respect de la vie privée des utilisateurs.

Les solutions proposées pour faire face à cet enjeu de sécurité doivent satisfaire les trois exigences ci-dessous [16] :

- Une faible charge computationnelle due au fait que les périphériques mobiles sont limités en ressources;
- Une forte sécurité, vu que les messages sont transmis via des réseaux sans fil non sécurisés;
- La protection de la vie privée des utilisateurs.

Les exigences d'une faible charge computationnelle et d'une forte sécurité sont en quelque sorte contradictoires. En effet, plus il y a des mécanismes de sécurité, plus il y a consommation des ressources. Un défi de recherche serait de proposer un modèle qui combine un niveau de sécurité élevé et une faible utilisation des ressources du périphérique mobile. Il faut donc faire un compromis entre le niveau de sécurité et la consommation des ressources.

La problématique de la sécurité dans un environnement client mobile/serveur se situe à plusieurs niveaux, dépendamment du point de vue par lequel elle est analysée : le point de vue du client mobile, le point de vue du serveur ou encore le point de vue de l'infrastructure réseau. Le Tableau 1.1 présente les problèmes de sécurité dans un environnement client mobile/serveur en fonction de leurs zones d'action.

Tableau 1.1 Problèmes de sécurité dans un environnement client mobile/serveur

Zone d'action	**Client mobile**	**Infrastructure réseau**	**Serveur**
Problème de sécurité	API non fiable Déni de service Vol ou perte Logiciel malveillant Vulnérabilité des applications Vulnérabilité du système d'exploitation Injections de code et SQL	Déni de service Attaque de l'homme du milieu Point d'accès malveillant	Fuites de données Administration des accès Déni de service Intégrité et confidentialité des données Non-respect de la vie privée Serveur malveillant Vulnérabilité des machines virtuelles

Le client mobile qui est le principal acteur du système est sujet à des menaces de sécurité. Les principales menaces sont les logiciels malveillants, les vulnérabilités des applications et les vulnérabilités du système d'exploitation. Les logiciels malveillants peuvent faire adopter au client mobile un comportement non désiré par l'utilisateur. Le client mobile peut par exemple exécuter un paiement sans le consentement de l'utilisateur. Des méthodes de détection de logiciels malveillants utilisent des algorithmes d'apprentissage machine pour réaliser une analyse statique ou une analyse dynamique [17]. Un défi de recherche serait d'explorer une approche hybride, combinant l'analyse statique et l'analyse dynamique, afin de tirer profit des avantages inhérents à chacune des méthodes d'analyse.

En ce qui concerne les vulnérabilités des applications, elles peuvent être dues au non-respect des bonnes pratiques en matière de développement des applications et à l'absence de critères concrets pour évaluer la sécurité des applications mobiles. Chen *et al.* [18] ont découvert des vulnérabilités,

comme l'utilisation de primitives cryptographiques non sécurisées dans des applications de banque mobile.

De nombreuses applications mobiles sauvegardent des données confidentielles comme les mots de passe et la clé de chiffrement dans la mémoire interne des périphériques mobiles. Un défaut de conception du système d'exploitation du périphérique peut donner suite à des attaques visant à accéder aux données confidentielles. Muhseen et Elameer [19] suggèrent aux utilisateurs de mettre régulièrement à jour le système d'exploitation de leurs périphériques mobiles pour faire face à cette menace. En revanche, une mise à jour automatique serait beaucoup plus appropriée, vu que les êtres humains sont exposés à des erreurs. Toutefois, les mises à jour ne corrigent que les défauts identifiés par les développeurs de logiciels, par conséquent, les logiciels demeurent avec des défauts qui ne sont pas encore identifiés. Une solution à explorer est l'utilisation de cartes à puce pour le stockage d'informations confidentielles comme les mots de passe et la clé de chiffrement.

Lorsque nous considérons le point de vue du serveur, les problématiques de sécurité auxquelles nous sommes confrontés incluent les fuites de données, la confidentialité, l'intégrité, le déni de service et le respect de la vie privée. Les données doivent être stockées sur des serveurs, de façon qu'elles ne puissent pas être interprétées par des personnes malveillantes en cas de fuite. Une méthode classique pour garantir la confidentialité des données consiste à les chiffrer avant de les stocker. Toutefois, la technique de chiffrement utilisée doit permettre d'effectuer des recherches d'information sur les données chiffrées, pour ne pas avoir à déchiffrer toutes les données sauvegardées sur le serveur avant d'effectuer la recherche. La sécurité du serveur implique également la mise en place d'un système de sauvegarde pour recouvrir les données des utilisateurs en cas d'attaque. La conception d'un nouveau système de défense face aux attaques de déni de service, constitue un défi de recherche dans la mesure où, dans le réseau Internet, ces attaques peuvent provenir de plusieurs sources. Nous gardons en mémoire l'attaque de déni de service *Mirai* qui a utilisé de milliers d'objets connectés pour paralyser des sites Web comme *Amazon*, *Spotify* et *Twitter* [20].

L'infrastructure réseau qui permet au client mobile et au serveur d'échanger entre eux, est également exposée aux problèmes de sécurité. La communication entre le client mobile et le serveur s'effectue via des réseaux de communication sans fil qui ne sont pas toujours sécurisés. Ceci expose la communication aux multiples attaques, à l'instar de l'attaque de l'homme du milieu

[21]. Dans l'attaque de l'homme du milieu, l'attaquant peut intercepter les communications et manipuler les messages entre le client mobile et le serveur, sans que ces derniers ne puissent se douter que le canal de communication a été compromis. Cet attaquant peut, par la même occasion, réaliser une usurpation d'identité. Il s'agit de se faire passer pour une des entités de la communication (i.e., le client mobile ou le serveur), afin d'accéder frauduleusement aux informations détenues par l'autre entité.

Pour aborder la problématique de sécurité liée à l'infrastructure réseau, des protocoles d'authentification mutuelle ont été proposés dans la littérature. En revanche, des lacunes récurrentes liées à ces protocoles sont la supposition d'un canal de communication sécurisé lors de la phase d'enregistrement, la présence d'une autorité de confiance, la non prise en compte du lieu de stockage des paramètres secrets et la gestion des clés. Au regard de ce qui précède, un défi de recherche est de concevoir un protocole d'authentification mutuelle entre un périphérique mobile et un serveur, qui assure la sécurité dans toutes les phases du protocole, limite l'intervention d'une autorité de confiance dans le protocole, gère le stockage sécurisé des paramètres secrets du protocole et consomme peu de ressources.

La problématique de la sécurité dans un environnement client mobile/serveur ainsi étudiée nous amène à la question de recherche principale qui est :

Comment améliorer la sécurité des communications dans un environnement client mobile/serveur?

Les questions de recherche secondaires qui découlent de cette question de recherche principale sont :

- Comment concevoir un protocole d'authentification mutuelle qui consomme peu de ressources tout en ayant un niveau de sécurité élevé?
- Sommes-nous en mesure de proposer un système de détection des attaques de déni de service auxquelles un serveur est exposé pour assurer la disponibilité du service?
- Quels sont les moyens dont nous disposons pour évaluer la sécurité des applications de banque mobile?

1.3 Objectifs de recherche

L'objectif principal de cette thèse est de concevoir des modèles robustes pour améliorer la sécurité des communications dans un environnement client mobile/serveur.

De manière plus spécifique, les objectifs que nous visons sont les suivants :

1. Concevoir un protocole d'authentification mutuelle entre un périphérique mobile et un serveur distant, qui prend en compte les caractéristiques des périphériques mobiles, dans le but de renforcer la sécurité des échanges entre les deux entités;
2. Concevoir un système en ligne de détection des attaques de déni de service effectuées à l'endroit des serveurs pour garantir la disponibilité du service aux clients mobiles légitimes;
3. Proposer un cadre pour l'évaluation de la sécurité des applications de banque mobile, afin de permettre aux banques d'intégrer la sécurité lors de la conception et du développement des applications mobiles;
4. Implémenter les différents modèles proposés pour établir leur faisabilité et évaluer leur performance en regard des caractéristiques propres à chacun d'entre eux.

1.4 Principales contributions de la thèse et leur originalité

Cette thèse contribue principalement à améliorer la sécurité dans un environnement client mobile/serveur. Ci-dessous les principales contributions de cette thèse :

1. La première contribution de cette thèse est de concevoir un nouveau protocole d'authentification mutuelle et d'échange de clé entre un client mobile et un serveur distant dans un environnement client mobile/serveur. Ce protocole se veut léger, afin de consommer au minimum les ressources du client mobile. De plus, il doit garantir des propriétés de sécurité, telles que la confidentialité persistante et la résistance aux attaques par rejeu. Les lacunes récurrentes dans les protocoles existants sont : la supposition de l'existence d'un canal de communication sécurisé entre le client mobile et le serveur lors de la phase d'enregistrement, la nécessité d'un matériel supplémentaire et le problème de synchronisation. Une des originalités du protocole proposé est l'introduction de l'*« Elliptic Curve Integrated Encryption Scheme » (ECIES)* [22] pour éliminer la supposition de l'existence d'un canal de communication sécurisé entre le client mobile et le serveur lors

de la phase d'enregistrement. En effet, l'*ECIES* garantit la confidentialité et l'intégrité des données. La seconde originalité observée est l'adoption de la carte *« Subscriber Identity Module » (SIM)* pour résoudre le problème du matériel supplémentaire. La carte *SIM* est un élément sécurisé qui est directement intégré au client mobile. Nous avons validé la sécurité du protocole à l'aide d'une évaluation informelle, d'une évaluation formelle avec la logique *« Burrows-Abadi-Needham » (BAN)* [23] et d'une simulation avec le logiciel *« Scyther »* [24].

2. La deuxième contribution de cette thèse est la conception d'un système basé sur l'entropie de l'information pour la détection des attaques de *déni de service distribué « Distributed Denial of Service » (DDoS)* par inondation. Le but visé est de maintenir le serveur disponible pour répondre aux requêtes des clients mobiles légitimes. Le système proposé s'appuie sur l'entropie de l'information, particulièrement l'*entropie de Shannon* [25]. L'utilisation de l'entropie nous permet d'avoir un système en ligne capable de détecter les attaques en un laps de temps. Les lacunes récurrentes dans les systèmes de détection existants sont : la détection de la présence d'une attaque sans identifier les connexions responsables de l'attaque, l'utilisation de seuils statiques et la validation des systèmes avec du trafic légitime et du trafic d'attaque provenant de deux ensembles de données différents. En effet, fusionner deux ensembles de données issus de réseaux ayant des caractéristiques différentes peut influencer les résultats [26]. L'originalité du système de détection proposé est qu'il détecte la présence d'une attaque et identifie les connexions responsables de cette attaque. De plus, nous évaluons le système de détection avec deux méthodes. La première méthode est la simulation à l'aide du simulateur réseau *« Graphical Network Simulator-3 » (GNS3)* [27]. La seconde méthode est l'évaluation à partir de l'ensemble de données public *CICIDS2017* [28] qui contient à la fois le trafic légitime et le trafic d'attaque.

3. La troisième contribution de cette thèse est de proposer un algorithme de seuil dynamique qui est utilisé dans le système de détection illustré à la deuxième contribution. Cet algorithme vient résoudre le problème d'utilisation de seuils statiques. Le but de cet algorithme est de mettre à jour le seuil de détection des attaques DDoS en fonction de l'évolution du trafic légitime. En effet, le trafic légitime étant dynamique, il est nécessaire que le seuil de détection le soit également pour réduire le taux de faux positif.

4. La quatrième contribution de cette thèse consiste à évaluer la sécurité des applications de banque mobile tout en proposant une solution au problème de vulnérabilité dans les applications mobiles. Nous nous sommes focalisés sur les applications de banque mobile et sur le système d'exploitation mobile Android. La littérature manque un cadre largement adopté par les chercheurs pour l'évaluation de la sécurité des applications de banque mobile. Le cadre proposé est constitué d'un ensemble de vingt-six critères (e.g., rejet de certificat autosigné, transmission de données sensibles sous forme chiffrée, etc.) concrets pour l'évaluation de la sécurité des applications de banque mobile. Les banques peuvent ainsi évaluer leurs applications sur la base de ces critères, avant de les déployer sur le marché. Nous avons évalué le cadre proposé sur la base de requis prédéfinis que sont : la *non-redondance*, la *non-ambiguïté* et la *complétude*.
5. La cinquième contribution est l'évaluation de la sécurité des applications Android de banque mobile de sept banques canadiennes. Cette évaluation est faite sur la base d'un sous-ensemble de critères proposés à la quatrième contribution. Pour ce faire, nous adoptons l'analyse dynamique qui consiste à étudier le comportement des applications de banque mobile lorsqu'elles sont en fonctionnement.

1.5 Plan de la thèse

Dans ce chapitre introductif, il est question de planter le décor de la thèse en présentant le problème étudié et les objectifs poursuivis. Nous avons commencé par définir les concepts de base nécessaires à la compréhension de la thèse. Par la suite, nous avons décrit les éléments de la problématique, qui ont donné suite aux objectifs de recherche et aux principales contributions de cette thèse. La suite de notre travail de thèse est constituée des chapitres 2 à 8.

Le chapitre 2 est une revue critique de la littérature sur les mécanismes de sécurité dans un environnement client mobile/serveur. Nous abordons les standards en matière de sécurité des communications client/serveur sur Internet, les problèmes de sécurité et les mécanismes de sécurité dans un environnement client mobile/serveur.

Le chapitre 3 présente la démarche de l'ensemble du travail de recherche. Dans ce chapitre, nous indiquons également la cohérence entre les objectifs de recherche et les articles scientifiques résultant de cette thèse.

Dans le chapitre 4, nous présentons l'article 1 intitulé *A New Mutual Authentication and Key Agreement Protocol for Mobile Client – Server Environment*, publié dans le journal *IEEE Transactions on Network and Service Management*. Cet article propose un protocole d'authentification mutuelle et d'échange de clé entre un client mobile et un serveur. Le protocole se base sur l'ECC pour limiter l'utilisation des ressources du client mobile.

Dans le chapitre 5, nous présentons l'article 2 intitulé *An Online Entropy-based DDoS Flooding Attack Detection System with Dynamic Threshold*, publié dans le journal *IEEE Transactions on Network and Service Management*. Dans cet article, nous décrivons un système de détection des attaques de déni de service distribué par inondation. Ce système se base sur l'entropie de l'information pour détecter les attaques en un laps de temps.

Dans le chapitre 6, nous présentons l'article 3 intitulé *A Framework for Security Assessment of Android Mobile Banking Applications*, soumis pour publication dans le journal *IEEE Transactions on Dependable and Secure Computing*. Dans cet article, nous proposons un cadre constitué d'un ensemble de critères pour l'évaluation de la sécurité des applications Android de banque mobile. Par la suite, nous évaluons la sécurité de sept applications de banque mobile des banques majeures canadiennes.

Le chapitre 7 est consacré à une discussion générale de l'ensemble de la thèse.

Le chapitre 8 quant à lui clôt la thèse.

CHAPITRE 2 REVUE DE LA LITTÉRATURE

Ce chapitre est consacré à une revue critique de la littérature. Dans un premier temps, nous présentons les articles scientifiques qui identifient les problèmes de sécurité dans un environnement client mobile/serveur. Ensuite, nous passons en revue les standards de sécurité des communications client/serveur sur Internet. Enfin, nous explorons les solutions proposées dans la littérature pour pallier les problèmes de sécurité dans un environnement client mobile/serveur, en mettant l'accent sur les protocoles d'authentification mutuelle entre un client mobile et un serveur. Nous avons porté notre choix sur ces protocoles car, en plus d'authentifier le client mobile et le serveur, ils permettent d'échanger une clé de chiffrement pour sécuriser le canal de communication.

2.1 Problèmes de sécurité dans un environnement client mobile/serveur

De nos jours, de nombreuses applications sont déployées sur des périphériques mobiles. Cependant, ces périphériques possèdent des ressources limitées [7] [8] : batteries, espace mémoire et capacité de traitement. Ce qui favorise la pleine croissance du paradigme *« Mobile Cloud Computing » (MCC)*. Ce paradigme consiste à combiner l'informatique mobile et l'informatique en nuage, dans le but de transférer l'exécution d'une partie ou la totalité des applications mobiles vers le nuage qui possède de ressources plus larges [7] [29]. La communication entre un client mobile et un serveur distant donne naissance à de nouveaux défis, dont celui de la sécurité. Les problèmes de sécurité peuvent être liés au client mobile, au serveur ou au réseau de communication entre le client mobile et le serveur [19] [30].

Vishal *et al.* [31] relèvent que la transmission des données d'un périphérique mobile vers un serveur distant dans un environnement ouvert, expose ces données aux attaques de pirates informatiques. Il est nécessaire de proposer des mécanismes qui assurent une transmission sécurisée des données. Dans cet article, les auteurs font une évaluation de quelques menaces de sécurité rencontrées dans un environnement MCC. Il s'agit de :

- Fuites de données : des données sensibles ou privées sont accessibles aux personnes non autorisées;
- Administration des accès : il s'agit de contrôler les informations auxquelles les utilisateurs ont accès;

- Interfaces de programmation d'application *« Application Programming Interface » (API)* peu fiables : des API sont utilisées pour communiquer entre applications et serveurs, donc la fiabilité de l'API doit être prise en compte;
- Dénis de service : il s'agit de rendre le service indisponible, par exemple, en consommant frauduleusement l'énergie du serveur [32];
- Périphériques mobiles : les menaces liées aux périphériques mobiles incluent le vol ou la perte des périphériques mobiles ainsi que les logiciels malveillants.

Kumar *et al.* [8] font une revue des mécanismes qui permettent de transférer l'exécution d'une partie des applications mobiles vers le nuage. Ce transfert est dû aux ressources limitées des périphériques mobiles [33]. Les auteurs relèvent que les problèmes liés à la sécurité et à la vie privée demeurent des défis dans ces mécanismes. Par exemple, dans une application de localisation, la vie privée des utilisateurs n'est pas respectée dans la mesure où le serveur peut faire le suivi de la position de ces utilisateurs. Les auteurs soulignent également des problèmes d'authentification qui donnent l'accès aux données à des personnes non autorisées et les problèmes d'intégrité des données stockées dans des serveurs non fiables.

Vishal *et al.* [34] présentent les défis de sécurité rencontrés dans un environnement MCC sous deux aspects : la sécurité des données ou des fichiers et la sécurité des applications mobiles. La sécurité des données ou des fichiers doit garantir le non-accès à ces données ou ces fichiers aux personnes non autorisées. La sécurité des applications mobiles quant à elle, assure que les applications n'ont pas été compromises.

Muhseen et Elameer [19] font une revue des problèmes de sécurité dans un environnement MCC. Les menaces de sécurité sont divisées en fonction de leurs zones d'intervention : le périphérique mobile, le réseau mobile et le nuage. Les menaces identifiées au niveau du périphérique mobile sont les logiciels malveillants ainsi que les vulnérabilités des applications mobiles et du système d'exploitation. Les menaces liées aux réseaux mobiles sont dues au fait que plusieurs canaux de communication qui ne sont pas toujours sécurisés peuvent être utilisés, comme le *« Short Message Service » (SMS)*, le WI-FI et le Bluetooth. Les menaces au niveau du nuage sont la fiabilité de la plateforme qui peut par exemple subir des attaques de déni de service, l'intégrité et la confidentialité des données. Les auteurs proposent quelques approches pour remédier aux problèmes de sécurité, à savoir : la détection et l'élimination des logiciels malveillants, la mise à

jour des applications mobiles et du système d'exploitation, le chiffrement des données, le contrôle d'accès et l'authentification.

Juárez et Cedillo [35] relèvent que les thèmes les plus étudiés, relatifs à la sécurité dans un environnement MCC, sont l'attaque de l'homme du milieu et le risque de fuite de données. De plus, les caractéristiques de sécurité prises en compte sont la confidentialité, l'intégrité et l'authentification. Cependant, ils mentionnent que très peu d'études traitent des sujets liés à la responsabilité et à la non-répudiation.

Zhang *et al.* [36] affirment que le transfert de données des utilisateurs mobiles vers des serveurs distants, pose le problème de fuite de données, de confidentialité et d'intégrité. La sécurité de ces données demeure un problème fondamental. Quelques défis de sécurité identifiés sont : le déni de service, l'attaque de l'homme du milieu, les points d'accès réseau malveillants et la falsification des données. Les auteurs soulignent des axes de recherche ouverts parmi lesquels, les systèmes de chiffrement de données légers qui consomment peu de ressources, l'affectation d'une identité aux différentes entités du système et la mise en place d'une authentification mutuelle entre ces entités.

Les problèmes de sécurité présentés par Somula et Sasikala [7] sont : la fuite de données, les logiciels malveillants, les points d'accès réseau non fiables, les vulnérabilités des systèmes d'exploitation, les attaques d'injection *« Structured Query Language » (SQL)* et le déni de service. Les auteurs affirment la nécessité des protocoles sécurisés pour protéger le canal de communication entre les périphériques mobiles et le nuage, et garantir l'intégrité des données, l'authentification et le contrôle d'accès.

Roman *et al.* [30] classent les menaces de sécurité selon leurs zones d'action à savoir l'infrastructure réseau, les serveurs et les périphériques mobiles. Les menaces qui sont le plus souvent identifiées au niveau de l'infrastructure réseau sont : le déni de service, l'attaque de l'homme du milieu et les points d'accès malveillants. Quant aux serveurs, les menaces sont : la fuite de données, le non-respect de la vie privée, les serveurs malveillants et le déni de service. Au niveau des périphériques mobiles, les menaces identifiées sont les injections de fausses informations. Les auteurs proposent par la suite des mécanismes pour assurer la sécurité de l'environnement :

- L'affectation d'une identité aux différentes entités et l'authentification entre ces entités;

- Le contrôle d'accès : utile dans la mesure où les entités autorisées sont les seules à avoir accès aux ressources;
- La mise en place des protocoles de communication sécurisés.

Cependant, ils révèlent la nécessité de réduire la latence des mécanismes de sécurité.

Les défis de sécurité identifiés par Mollah *et al.* [37] sont : la fuite ou la perte de données, les vulnérabilités des machines virtuelles, les logiciels malveillants et la perte ou le vol du périphérique mobile. Les solutions proposées dans la littérature pour faire face aux problèmes de sécurité se basent principalement sur la cryptographie. Comme questions de recherche ouvertes, les auteurs identifient l'importance de sécuriser la communication entre le périphérique mobile et le nuage et la nécessité des mécanismes de sécurité légers du fait des ressources limitées des périphériques mobiles.

Nisha et Bhanu [38] effectuent une revue des attaques par injection de code présentes dans un environnement MCC. L'attaquant introduit un code malicieux dans un champ d'entrée qui sera exécuté par l'application, en vue d'avoir des accès non autorisés. Le code peut être transmis par des canaux tels que SMS, WI-FI, Bluetooth, *« Near-Field Communication » (NFC)* et la caméra du périphérique mobile. La validation des entrées est vue comme le mécanisme le plus efficace pour se prémunir de ce type d'attaque.

En dépit des efforts effectués pour sécuriser l'informatique en nuage, Khan *et al.* [29] affirment que des zones d'ombre demeurent, à savoir : la sécurité des données stockées dans des serveurs, les menaces de sécurité dues aux multiples machines virtuelles et la détection des intrusions. Le MCC hérite de ces problèmes de sécurité, puisqu'il est basé sur l'informatique en nuage. Les mécanismes de sécurité de l'informatique en nuage, ne peuvent pas directement être appliqués aux périphériques mobiles à cause de leurs ressources limitées. La conception des versions légères de ces mécanismes adaptées aux périphériques mobiles, est un défi qui doit être relevé. Les auteurs relèvent la nécessité d'un canal de communication sécurisé entre le périphérique mobile et le nuage. Quelques services de sécurité identifiés sont : la protection de la vie privée, la vérification de l'intégrité des données, la gestion des utilisateurs, la gestion des clés, le chiffrement des données, la détection des intrusions et l'authentification.

Le Tableau 1.1 fait un récapitulatif des problèmes de sécurité dans les communications entre un client mobile et un serveur distant.

2.2 Protocoles de sécurité des communications client/serveur sur Internet

Les protocoles les plus connus pour assurer la sécurité des communications client/serveur sur Internet sont *« Secure Shell » (SSH)* [39] [40] [41] [42], *« Internet Protocol Security » (IPsec)* [43] [44] [45] et *« Transport Layer Security » (TLS)* [46]. La standardisation de ces protocoles est effectuée par *« Internet Engineering Task Force » (IETF)*.

2.2.1 Protocole SSH

SSH [39] [40] [41] [42] est un protocole qui utilise le chiffrement pour sécuriser les communications entre un client et un serveur. Il est utilisé pour fournir un accès sécurisé à des utilisateurs aux serveurs distants, transférer des fichiers, envoyer des commandes à distance et gérer une infrastructure réseau.

SSH comprend trois composantes :

- Le protocole de la couche transport [40] qui fournit les services d'authentification du serveur, d'échange de clé, de confidentialité et d'intégrité;
- Le protocole d'authentification des utilisateurs [41] dont le rôle est d'authentifier le client SSH au niveau du serveur;
- Le protocole de connexion [42] qui permet d'établir des sessions de communication interactives entre le client et le serveur.

Dans le protocole *SSH*, l'établissement de la communication entre un client et un serveur utilise le chiffrement à clé publique. Un *client SSH* initie une connexion en contactant le *serveur SSH* et en retour, le serveur envoie sa clé publique au client. Ensuite, le client et le serveur négocient les paramètres qui seront utilisés lors de la communication, dont les algorithmes de chiffrement et les fonctions de hachage. Le client peut s'authentifier auprès du serveur avec la combinaison *nom d'utilisateur/mot de passe* ou avec une *clé privée*. Dans le cas de l'authentification avec une clé privée, la clé publique correspondante à la clé privée est configurée au niveau du serveur. Ainsi, seuls les clients possédant une clé privée valide pourront accéder au serveur.

Après l'établissement de la connexion, le protocole *SSH* utilise les algorithmes de chiffrement symétrique (e.g., *« Advanced Encryption Standard » (AES)*) et les fonctions de hachage (e.g., *« Secure Hash Algorithm - 256 » (SHA-256)*) négociés, pour garantir la confidentialité et l'intégrité des données échangées entre le client et le serveur.

2.2.2 Protocole IPsec

IPsec [43] est un ensemble de protocoles pour la sécurisation des données échangées sur les réseaux *« Internet Protocol » (IP)*, dont Internet. Sa particularité est qu'il fournit des services de sécurité au niveau de la couche réseau du modèle *« Transmission Control Protocol/Internet Protocol » (TCP/IP)*.

IPsec est constitué principalement de deux protocoles de sécurité, à savoir : *« Authentication Header » (AH)* [44] et *« Encapsulating Security Payload » (ESP)* [45]. Il est associé au protocole *« Internet Key Exchange » (IKE)* [47] pour l'échange et la gestion des clés.

AH [44] assure principalement deux rôles : l'authentification de l'origine des données et l'intégrité des données. Il vérifie que les datagrammes IP reçus, proviennent effectivement de l'hôte dont l'adresse IP est indiquée comme adresse source dans l'entête. Il contrôle également que certains champs du datagramme IP n'ont pas été modifiés, en utilisant des *« Message Authentication Code » (MAC)* ou des fonctions de hachage.

En plus des services fournis par le protocole *AH*, *ESP* [45] garantit la confidentialité des données en utilisant le chiffrement symétrique.

IPsec peut fonctionner suivant deux modes : le mode transport et le mode tunnel. Dans le mode transport, l'entête du datagramme IP n'est pas modifié et les services de sécurité s'appliquent uniquement aux données du datagramme. En revanche, dans le mode tunnel, tout le datagramme IP initial est encapsulé dans un datagramme *IPsec*, ayant comme adresse IP de destination une passerelle de sécurité qui implémente *IPsec*.

2.2.3 Protocole TLS

TLS [46] est un protocole utilisé pour sécuriser les communications client/serveur sur Internet. Il succède au protocole *« Secure Sockets Layer » (SSL)* et le rend de ce fait obsolète. *TLS* implémente les mécanismes suivants :

- Le certificat numérique pour authentifier le serveur;
- Le chiffrement à clé publique pour l'échange de clé de session;
- Le chiffrement symétrique pour garantir la confidentialité des données;
- Le code MAC pour assurer l'intégrité des données.

Le protocole *TLS* est subdivisé en deux phases : l'établissement de la liaison et l'échange de données. L'établissement de la liaison utilise la cryptographie à clé publique pour échanger la clé de session. Le client initie la connexion en envoyant une requête au serveur contenant les versions de *TLS* et les algorithmes de chiffrement qu'il est capable de supporter. Le serveur répond à la requête en envoyant son certificat numérique signé par une autorité de certification, la version de *TLS* et les algorithmes de chiffrement qu'il a choisis. Par la suite, le client vérifie la signature du certificat du serveur afin de se rassurer de la validité. Si le certificat n'est pas valide, la procédure prend fin. S'il est valide, le client génère la clé de session, la chiffre avec la clé publique du serveur et la transmet au serveur. Ce dernier déchiffre la clé de session avec sa clé privée, et il partage dorénavant une clé de chiffrement symétrique avec le client.

Au cours de l'échange de données, le client chiffre le message à envoyer avec la clé de session et calcule son code MAC. À la réception, le serveur déchiffre le message avec la clé de session et vérifie le code MAC. Il effectue la même procédure pour envoyer un message au client.

2.2.4 Lacunes des protocoles SSH, IPsec et TLS

Les protocoles classiques de sécurité des communications client/serveur sur Internet, à savoir *SSH*, *IPsec* et *TLS*, utilisent la cryptographie à clé publique pour établir un canal sécurisé entre le client et le serveur. Cependant, ces protocoles peuvent difficilement être déployés sur des périphériques mobiles qui ont des ressources limitées (i.e., capacité de traitement, espace mémoire et batterie). En effet, *SSH*, *IPsec* et *TLS* utilisent des méthodes comme les certificats numériques et des algorithmes comme *« Rivest, Shamir, Adleman » (RSA)* qui sont gourmands en ressources.

Le protocole *SSH* est vulnérable aux attaques de l'homme du milieu et d'usurpation d'identité [39]. En effet, *SSH* ne met en place aucun mécanisme permettant d'associer l'identité d'un serveur à sa clé publique. Ainsi, lors de l'établissement d'une session, un attaquant peut intercepter la requête de connexion d'un client et envoyer en retour sa clé publique. Il peut ainsi se faire passer pour le

serveur légitime, ou encore relayer les messages échangés entre le client et le serveur tout en observant la communication. Il est donc nécessaire d'associer à *SSH*, des moyens pour lier l'identité d'un serveur à sa clé publique.

IPsec implémente les certificats numériques par le biais du protocole IKE, afin de garantir l'association entre l'identité d'une entité et sa clé publique. Néanmoins, ce mécanisme qui consomme énormément de ressources le rend inadapté pour des périphériques mobiles.

De même que *IPsec*, le protocole *TLS* implémente les certificats numériques pour lier l'identité d'une entité à sa clé publique. De plus, l'authentification du client dans le protocole *TLS* n'est pas obligatoire [46], ce qui donne lieu à des failles possibles de sécurité.

Les lacunes des mécanismes classiques présentés ci-dessus ont donné l'occasion à des chercheurs de proposer des mécanismes de sécurité plus appropriés aux périphériques mobiles.

2.3 Mécanismes de sécurité dans un environnement client mobile/serveur

Des thématiques de recherche, liées à l'amélioration de la sécurité dans un environnement client mobile/serveur, ont déjà été traitées et existent dans la littérature. Ajay et Umamaheswari [48] proposent une méthode de chiffrement dans laquelle une clé utilisée pour chiffrer un paquet dans une session ne peut plus être réutilisée pour chiffrer d'autres paquets de la même session. Pour réaliser cette méthode, le protocole d'échange de clé *« Diffie-Hellman » (DH)* [13] est implémenté avant l'envoie de chaque paquet d'une session. Cependant, le problème de latence créé par la répétition du protocole d'échange de clé *DH* avant le transfert de chaque paquet n'est pas clairement étudié.

Subramaniam et Jeet [17] explorent la détection des logiciels malveillants dans un environnement mobile, en utilisant des algorithmes d'apprentissage machine. Les auteurs distinguent deux catégories de détection, à savoir l'analyse statique et l'analyse dynamique. L'analyse statique examine le code source de l'application sans l'exécuter, tandis que l'analyse dynamique contrôle le comportement dynamique de l'application lorsqu'elle est exécutée.

Fournier *et al.* [49] proposent un modèle de détection des logiciels malveillant pour les appareils Android. Les auteurs adoptent une approche client/serveur afin de réduire la charge de calcul au niveau du périphérique mobile. Le modèle de détection se base sur l'analyse statique et la méthode

de classification utilisant l'algorithme d'apprentissage automatique de régression *« Random Forest »*.

Rodrigo *et al.* [50] décrivent un modèle hybride de détection des logiciels malveillants pour des appareils Android, en utilisant à la fois l'analyse statique et l'analyse dynamique, tout en se basant sur une approche client/serveur. La prédiction se fait au niveau du serveur par un réseau de neurones.

Inani *et al.* [51] mettent en avant les problèmes de confidentialité, d'intégrité et de disponibilité des données mobiles stockées sur des serveurs. Dans le modèle proposé, les données sont divisées en deux classes : les données publiques et les données confidentielles, dans le but d'appliquer des mesures de sécurité appropriées. La classification des données utilise l'algorithme d'apprentissage machine *« Training dataset Filtration Key Nearest Neighbour » (TsF-KNN)* [52]. Par la suite, seules les données classées confidentielles sont chiffrées avec la cryptographie sur les courbes elliptiques. L'objectif est de réduire le processus de chiffrement et l'appliquer uniquement aux données confidentielles, dans l'optique d'accroître l'efficacité des périphériques mobiles.

Xie *et al.* [53] proposent un mécanisme de contrôle d'accès dans un environnement MCC utilisant le modèle *« Modified Hierarchical Attribute-Based Encryption » (M-HABE)*. Il s'agit d'une version modifiée du chiffrement par attributs [54]. Le but est de s'assurer que les données stockées sur des serveurs ne sont accessibles que par les seules personnes autorisées.

Sandip *et al.* [55] proposent un modèle de contrôle d'accès aux données et d'authentification des utilisateurs dans un environnement MCC, pour des applications de santé. Le but est de fournir l'accès aux données uniquement aux utilisateurs autorisés. Le modèle proposé utilise des primitives cryptographiques telles que les fonctions de hachage, le couplage bilinéaire et le chiffrement par attributs. De plus, les auteurs font appel à la biométrie pour améliorer la sécurité de leur système.

Les recherches de Keykhaie et Pierre [56] [57] [58] portent sur l'authentification des utilisateurs sur un téléphone intelligent. Dans [56], les auteurs présentent un modèle générique d'authentification, qui peut utiliser différentes biométries et s'exécuter sur une carte *« Subscriber Identity Module » (SIM)* pour respecter la vie privée des utilisateurs. Dans [57], les auteurs proposent une authentification active des utilisateurs basée sur leur comportement, tel que les gestes qu'ils effectuent sur les écrans tactiles des téléphones intelligents. Cette authentification active authentifie de façon continue et transparente les utilisateurs durant l'utilisation des

téléphones. La vérification de l'identité des utilisateurs se fait grâce à un modèle simplifié de réseau de neurones profond, qui s'exécute sur une carte SIM. Dans [58], les auteurs proposent une authentification active basée sur le visage. Ils distinguent deux types d'architecture. La première architecture utilise les ressources du nuage pour l'enregistrement des utilisateurs et la carte SIM ou la carte *« embedded Subscriber Identity Module » (eSIM)* pour la vérification. Dans la seconde architecture, l'enregistrement et la vérification se font sur la carte SIM/eSIM.

2.3.1 Protocoles d'authentification mutuelle entre client mobile et serveur

Dans l'optique d'apporter leur pierre à l'édifice, certains chercheurs ont manifesté leur intérêt en donnant des propositions de protocoles d'authentification mutuelle entre un client mobile et un serveur. Ces protocoles doivent garantir une faible charge computationnelle due au fait que les clients mobiles sont limités en ressources. De plus, une forte sécurité est nécessaire étant donné que les messages sont transmis via des réseaux sans fil non sécurisés. En outre, la protection de la vie privée des utilisateurs doit être prise en compte [16].

Les protocoles d'authentification mutuelle existant dans la littérature peuvent être catégorisés en fonction des techniques cryptographiques utilisées, à savoir : la cryptographie asymétrique, la cryptographie symétrique, la cryptographie sur les courbes elliptiques et la cryptographie à base de couplages.

Les protocoles d'authentification mutuelle proposés dans les articles [59] [60] [61] [62], utilisent la cryptographie asymétrique. Fan *et al.* [59] présentent un protocole d'authentification mutuelle basé sur le second facteur universel, afin de garantir la sécurité des systèmes de paiement mobile. Ce protocole est capable d'identifier et de rejeter les serveurs et les clients contrefaits. La cryptographie asymétrique est utilisée lors de l'authentification mutuelle entre le périphérique mobile et le serveur. Le protocole est divisé en deux phases : la phase d'enregistrement et la phase d'authentification. La carte *« Universal Subscriber Identity Module » (USIM)* du téléphone est utilisée pour le stockage de la clé privée et le traitement des données. Le protocole proposé est capable de faire face aux attaques par rejeu et de contrefaçon. Cependant, le second facteur d'authentification peut nécessiter un matériel additionnel qui n'est pas toujours pratique pour les utilisateurs.

Un autre mécanisme d'authentification mutuelle utilisant la cryptographie asymétrique est exposé dans [60]. Le protocole a pour but d'améliorer la sécurité des systèmes de paiement mobile. Cette approche s'appuie sur la *« Public Key Infrastructure » (PKI)* et les informations sont cryptées dans un code à réponse rapide *« Quick Response » (QR)*. Le marchand chiffre les informations de la transaction grâce à la clé publique du client et génère un code QR contenant les informations de la transaction chiffrées. Le client peut scanner le code QR et déchiffrer les informations grâce à sa clé privée. L'authentification mutuelle est faite par la PKI qui garantit l'authenticité du client et du marchand. La principale limite de cette solution est la totale confiance à la PKI pour l'authentification, qui représente un point de défaillance du système. De plus, le processus de transfert de clé privée de la PKI vers le client n'est pas spécifié.

Munivel et Kannammal [61] proposent un protocole d'authentification basé sur la technique *« Zero-knowledge proof »*, dans un environnement MCC. Ce protocole se veut résistant aux attaques d'un environnement MCC, telles que le rejeu, l'homme du milieu, le déni de service, l'usurpation d'identité, l'hameçonnage et d'autres. Sa particularité est que le mot de passe réel n'est jamais transmis à une entité de la communication durant tout le processus d'authentification. Le protocole fait appel à une autorité de confiance, *« Trusted Third Party » (TTP)*, qui joue le rôle de serveur d'authentification. Comme limite, les auteurs supposent la présence d'un canal sécurisé entre l'utilisateur et le serveur d'authentification lors de la phase d'enregistrement, ce qui n'est pas toujours le cas. La gestion des informations secrètes (e.g., mot de passe et clé) au niveau du périphérique mobile de l'utilisateur n'est pas spécifiée. De plus, le protocole utilise une autorité de confiance.

Abuarqoub [62] présente un modèle d'authentification à deux facteurs entre le client mobile et le serveur distant, basé sur le mot de passe et la carte à puce. Le modèle est divisé en trois phases : la phase d'enregistrement, la phase de connexion et la phase de vérification. L'objectif est de vérifier l'identité du client et celle du serveur.

Les mécanismes d'authentification mutuelle utilisant la cryptographie asymétrique présentent l'inconvénient de la consommation élevée des ressources computationnelles et mémoire, ce qui n'est pas convenable aux périphériques mobiles limités en ressource.

Les protocoles proposés dans les articles [21] [63] [64] [65] [66] utilisent la cryptographie symétrique pour réaliser le processus d'authentification mutuelle. Ahmed et Wendy [21] présentent

une revue de quelques protocoles d'authentification et par la suite, font une proposition de protocole. Il s'agit d'une authentification à plusieurs facteurs, dont le premier est la combinaison *nom d'utilisateur/mot de passe*, et le second est un *« One-Time Password » (OTP)* généré par le périphérique mobile. L'OTP est envoyé au serveur, qui le compare avec celui qu'il a calculé. Le lieu de stockage de la clé de chiffrement n'est pas pris en compte dans ce protocole. De plus, il ne s'agit pas réellement d'une authentification mutuelle, car le serveur n'est pas authentifié par le client.

Dey *et al.* [63] proposent un protocole d'authentification mutuelle basé sur la fonction de hachage, la localisation et l'horodatage. Le protocole permet de valider les identités d'un client mobile et d'un serveur qui communiquent entre eux. Ce protocole utilise la cryptographie à clé symétrique et emploie des clés de chiffrement dynamiques, afin de minimiser la prévisibilité de la clé. Cependant, ce protocole ne considère pas le lieu de stockage des paramètres secrets au niveau du client mobile, qui est vulnérable aux applications malveillantes.

Dey *et al.* [64] [65] proposent un protocole d'authentification mutuelle basé sur la fonction de hachage pour améliorer la sécurité dans un environnement MCC. Le protocole proposé utilise les fonctions de hachage et le traditionnel *nom d'utilisateur/mot de passe*. Dans un environnement MCC, le périphérique mobile doit au préalable s'enregistrer auprès du serveur. Comme lacunes, la clé est transmise en clair lors de la phase d'enregistrement et suppose la présence d'un protocole de sécurité comme SSH. Aussi, le lieu de stockage des clés au niveau du périphérique mobile n'est pas pris en compte.

Donald et Arockiam [66] abordent le problème d'authentification rencontré dans un environnement MCC. Le protocole est divisé en deux processus : le processus d'enregistrement et le processus d'authentification. Le protocole proposé suppose la présence d'un canal de communication sûr (e.g., SSH) lors du processus d'enregistrement, ce qui n'est pas toujours le cas. De plus, le lieu de stockage des clés de chiffrement n'est pas abordé.

Les auteurs dans [67] [68] [69] [70] adoptent des protocoles utilisant la cryptographie sur les courbes elliptiques. Mo *et al.* [67] proposent un protocole d'authentification mutuelle et de partage de clé dans un environnement client/serveur, en utilisant la cryptographie sur les courbes elliptiques. Le protocole proposé satisfait aux exigences d'authentification mutuelle, d'anonymat, de résistance aux attaques par rejeu, d'initié et d'usurpation d'identité. Cependant, ce protocole

présente des limites, comme le lieu de stockage des informations secrètes au niveau du terminal mobile qui n'est pas pris en compte. De plus, l'implémentation du protocole proposé pour prouver son effectivité n'est pas réalisée. En conséquence, il n'y a pas de précision sur les algorithmes de hachage et de génération de nombre aléatoire utilisés. La supposition de la présence d'un canal sécurisé entre le client et le serveur lors de la phase d'enregistrement n'est pas toujours réaliste.

Mo *et al.* [68] présentent un protocole d'authentification mutuelle sûr et efficace entre le périphérique mobile et le serveur, dans un environnement MCC. Le protocole est composé de trois phases : l'initialisation du système, l'enregistrement de l'utilisateur et l'authentification. Ce protocole présente néanmoins quelques insuffisances. Par exemple, l'intégrité des données transmises entre le client mobile et le serveur pendant la phase d'enregistrement n'est pas prise en compte. De plus, le protocole défini utilise du matériel additionnel, à savoir une carte à puce.

Dans [69], les auteurs conçoivent un protocole d'authentification mutuelle dans un environnement MCC, basé sur l'identité et utilisant la cryptographie sur les courbes elliptiques. Le protocole est divisé en trois phases : l'initialisation, l'enregistrement de l'utilisateur et l'authentification mutuelle avec échange de clé. Au cours de l'initialisation, le serveur définit et publie les paramètres publics du système. Dans la deuxième phase, l'utilisateur transmet au serveur son identifiant. Dans la dernière phase, l'utilisateur est authentifié grâce à son identifiant et son mot de passe. Toutefois, la supposition de la présence d'un canal de communication sécurisé lors de la phase d'enregistrement et l'utilisation de matériel supplémentaire, sont quelques limites que nous avons relevées au protocole proposé.

Xiong *et al.* [70] proposent un protocole d'authentification mutuelle entre un périphérique mobile et un serveur dans un environnement MCC. Le modèle proposé s'appuie sur la cryptographie sur les courbes elliptiques. Il est constitué de quatre phases : la phase d'initialisation, la phase d'enregistrement, la phase d'authentification et la phase de mise à jour de mot de passe. Dans la phase d'initialisation, l'autorité de confiance qui est le *« Smart Card Generator » (SCG),* choisit et publie les paramètres publics du système, parmi lesquels la courbe elliptique et les fonctions de hachage. Au cours de la phase d'enregistrement, l'utilisateur fournit son identifiant et son mot de passe au SCG et le serveur fournit son identifiant au SCG. Lors de la phase d'authentification, le périphérique mobile vérifie l'identité du serveur, qui à son tour, vérifie l'identité du périphérique mobile. La dernière phase consiste à mettre à jour le mot de passe de l'utilisateur. Une des lacunes

au modèle proposé est la supposition de la présence d'un canal sécurisé lors de la phase d'enregistrement. En outre, le modèle accorde une grande importance à une autorité de confiance dans les phases d'initialisation et d'enregistrement. Enfin, la sécurité des paramètres secrets sauvegardés au niveau du périphérique mobile n'est pas prise en compte.

Une autre technique cryptographique est la cryptographie à base de couplages. Les protocoles qui s'appuient sur cette technique sont proposés dans les articles [71] [72] [73] [16] [74]. Jegadeesan *et al.* [71] proposent un protocole d'authentification mutuelle anonyme entre un utilisateur mobile et un fournisseur de service, grâce à des crypto systèmes, sans l'utilisation du protocole SSL. Le protocole proposé doit également être capable d'effectuer l'échange de clé de session. Les trois principales entités du système sont l'utilisateur mobile, le fournisseur de service et l'autorité de confiance TTP. Les clés privées et les paramètres publics de chaque utilisateur mobile et fournisseur de service sont générés par le TTP. L'utilisateur mobile et le fournisseur de service sont identifiés par des identifiants uniques. L'analyse de sécurité montre que le protocole a les caractéristiques suivantes : résistance à l'attaque par rejeu, anonymat et authentification entre autres. Comme limite, le canal de communication entre l'utilisateur mobile et le TTP lors de la phase d'enregistrement est supposé être sécurisé, ce qui n'est pas toujours le cas. De plus, une autorité de confiance est utilisée pour l'enregistrement.

Le protocole présenté dans [72] utilise des opérations de cryptographie à base de couplages dans un environnement MCC multi serveurs. Le protocole est composé de trois phases : l'enregistrement, l'authentification mutuelle et la modification de mot de passe. Lors de la phase d'enregistrement, une autorité d'enregistrement calcule les clés privées de l'utilisateur et du serveur. L'utilisateur stocke ses paramètres secrets dans une carte à puce et il est identifié par un identifiant et un mot de passe. Les manquements relevés sont : l'utilisation d'une autorité d'enregistrement qui représente un point de défaillance du système; le canal de communication entre l'utilisateur et l'autorité d'enregistrement lors de la phase d'enregistrement est supposé confidentiel, ce qui n'est toujours le cas en pratique; et l'utilisation de matériels supplémentaires dont la carte à puce et le scanner biométrique.

Dans [73], les auteurs proposent un protocole d'authentification mutuelle qui respecte la vie privée et adresse les défis de sécurité des services de MCC. Le protocole proposé combine la signature vocale aux opérations cryptographiques. Comme limites, le lieu de stockage des paramètres secrets

au niveau de l'utilisateur n'est pas pris en compte et l'utilisation d'une autorité de confiance représente un point de défaillance du système.

Tsai et Lo [16] présentent un protocole d'authentification anonyme utilisant la cryptographie à base de couplages pour des services de MCC distribués. Le protocole proposé supporte l'authentification mutuelle, l'échange de clé et la non-traçabilité de l'utilisateur. Trois acteurs sont présents dans le protocole proposé : l'utilisateur mobile, le fournisseur de service et l'autorité de confiance SCG. Le SCG est responsable de la génération des paramètres publics du système, ainsi que des clés privées des fournisseurs de service et des utilisateurs. Le protocole proposé est constitué de trois phases : initialisation, enregistrement et authentification. Les insuffisances relevées sont : la supposition de la présence d'un canal de communication sûr lors de la phase d'enregistrement; l'utilisation d'une autorité de confiance; la non-considération de la sécurité des paramètres secrets stockés dans le périphérique mobile; et la mauvaise utilisation de l'empreinte digitale vue que la collecte du paramètre biométrique ne fournit pas toujours la même valeur. De plus, des chercheurs ont démontré que ce protocole est vulnérable à l'attaque d'usurpation d'identité du serveur [75] [76] [77] [78].

Lu et Zhao [74] présentent un protocole d'authentification mutuelle utilisant la biométrie dans un environnement MCC. Ce protocole s'appuie sur la cryptographie à base de couplages. Les auteurs revendiquent que leur protocole possède les propriétés de sécurité que sont l'anonymat, l'authentification mutuelle et la résistance à l'attaque d'usurpation d'identité. Toutefois, les limites qui ressortent de ce protocole sont : la supposition d'un canal sécurisé durant la phase d'enregistrement, la présence d'une autorité de confiance et l'utilisation d'un matériel additionnel.

Deux autres protocoles proposés dans la littérature n'utilisent aucune des techniques cryptographiques vues plus haut, mais uniquement des fonctions de hachage et des opérations de *ou exclusif (XOR)*. Il s'agit des protocoles proposés dans les articles [79] [80]. Gope et Das [79] proposent un protocole d'authentification mutuelle anonyme basé sur la carte à puce pour un environnement MCC. Ce protocole est constitué de trois phases : la phase d'enregistrement et de réhabilitation, la phase d'authentification mutuelle et la phase de renouvellement de mot de passe. Comme limite, le protocole proposé suppose la présence d'un canal de communication sécurisé lors de la phase d'enregistrement, ce qui n'est pas toujours le cas. De plus, il fait appel à un matériel supplémentaire, la carte à puce.

Roy *et al.* [80] proposent un protocole d'authentification mutuelle dans un environnement MCC, basé sur les fonctions de hachage, les opérations XOR et les fonctions *« fuzzy extractor »*. Ce protocole supporte l'échange de clé, l'anonymat et la non-traçabilité. L'utilisateur est identifié par un identifiant, un mot de passe et un paramètre biométrique. La phase d'enregistrement suppose la présence d'un canal de communication sûr, bien que n'étant pas toujours le cas. L'utilisation d'une autorité de confiance représente un point de défaillance. La sécurité des paramètres secrets stockés dans le périphérique mobile de l'utilisateur n'est pas prise en compte. De plus, des détails sur la biométrie utilisée ne sont pas spécifiés.

Le Tableau A.1 de l'annexe A présente une classification des différents protocoles d'authentification mutuelle de la littérature revus précédemment. Les critères de classification retenus sont :

- La technique cryptographique utilisée : le protocole doit avoir une charge computationnelle faible, afin d'être approprié pour des périphériques mobiles ayant des ressources limitées. Il doit également avoir une forte sécurité pour faire face aux menaces de sécurité (e.g., usurpation d'identité, attaque de l'homme du milieu, etc.) des réseaux mobiles sans fil.
- La présence ou non d'une autorité de confiance : l'autorité de confiance représente un point de défaillance qui peut compromettre la sécurité du système tout entier. De plus, la communication avec l'autorité de confiance augmente la quantité de données transmises sur le réseau. En cas de présence d'une autorité de confiance, son intervention dans le protocole doit être réduite au stricte nécessaire.
- La présence ou non de matériel supplémentaire : il s'agit de tout matériel qui n'est pas embarqué dans la constitution du périphérique mobile. Le matériel supplémentaire peut engendrer des coûts et il n'est pas toujours confortable pour l'utilisateur. Bien qu'ils soient considérés comme du matériel supplémentaire, les cartes SIM et USIM peuvent s'avérer être pratiques pour les utilisateurs dans la mesure où ils sont directement intégrés dans le périphérique mobile. En outre, la carte eSIM, qui est l'évolution la plus récente de la carte SIM, est embarqué dans la constitution du périphérique mobile.
- Le lieu de stockage des paramètres secrets : ce critère est important dans la mesure où la sécurité fournie par les opérations cryptographiques est éliminée si la clé est dévoilée. Il est donc nécessaire de stocker les paramètres secrets de façon sécuritaire.

- L’identité du client mobile et l’identité du serveur : il s’agit des informations utilisées pour vérifier l’identité du client mobile et celle du serveur.

CHAPITRE 3 DÉMARCHE DE L'ENSEMBLE DU TRAVAIL DE RECHERCHE

Cette thèse vise à concevoir des modèles robustes pour améliorer la sécurité des communications dans un environnement client mobile/serveur. Pour atteindre cet objectif, nous avons énoncé quatre objectifs spécifiques à la section 1.3. Dans ce chapitre, nous présentons la démarche de l'ensemble du travail de recherche, tout en indiquant la cohérence entre les objectifs de la recherche et les chapitres subséquents. De manière générale, les travaux sont divisés en trois grands volets, faisant chacun l'objet d'un article scientifique. Ces volets sont : la conception du protocole d'authentification mutuelle, la conception du système de détection des attaques de déni de service distribué *« Distributed Denial of Service » (DDoS)* par inondation et la conception du cadre pour l'évaluation de la sécurité des applications de banque mobile.

3.1 Volet 1 : Conception du protocole d'authentification mutuelle

Une communication sécurisée entre un client mobile et un serveur passe par la sécurisation du canal de communication. De plus, chaque entité de la communication doit vérifier l'identité de son interlocuteur, pour s'assurer qu'elle ne communique pas avec un imposteur. Dans cette optique, le premier objectif spécifique de cette thèse est de concevoir un protocole d'authentification mutuelle entre un périphérique mobile et un serveur distant, qui prend en compte les caractéristiques des périphériques mobiles. Cet objectif est atteint dans l'article intitulé *« A New Mutual Authentication and Key Agreement Protocol for Mobile Client – Server Environment »* et présenté au chapitre 4.

3.1.1 Technique cryptographique

La cryptographie sur les courbes elliptiques *« Elliptic Curve Cryptography » (ECC)* est la technique cryptographique adoptée dans notre protocole d'authentification mutuelle. L'*ECC* possède un avantage sur les autres techniques cryptographiques à clé publique (e.g., *« Rivest, Shamir, Adleman » (RSA)*), puisqu'elle permet d'obtenir un niveau de sécurité élevé, avec une faible utilisation de la mémoire et un faible temps de calcul [15]. De ce fait, cette technique est plus adaptée pour les périphériques mobiles limités en ressource. À titre de comparaison, le Tableau 3.1 montre qu'*ECC* atteint un niveau de sécurité de 128 bits avec une clé de taille 256 bits, tandis que *Diffie-Hellman (DH)* et *RSA* atteignent le même niveau de sécurité avec une clé de 3072 bits [12].

Tableau 3.1 Taille de clé (en bits) des algorithmes DH, RSA et ECC en fonction du niveau de sécurité

Niveau de sécurité (en bits)	**DH**	**RSA**	**ECC**
≤ 80	1024	1024	160
112	2048	2048	224
128	3072	3072	256

3.1.2 Évaluation de la sécurité

Nous considérons deux méthodes pour évaluer la sécurité de notre protocole d'authentification mutuelle. Ces deux méthodes sont : l'analyse avec la logique *« Burrows-Abadi-Needham » (BAN)* [23] et la simulation avec l'outil *« Scyther »* [24].

La logique BAN a été mise en place en 1989 par Burrows, Abadi et Needham [23]. Elle permet d'effectuer une analyse de la sécurité des protocoles d'authentification. Le but est de s'assurer que ces protocoles fonctionnent correctement. La logique BAN doit être en mesure de détecter des redondances et des failles de sécurité dans des protocoles d'authentification. Elle aide à répondre aux questions ci-dessous [23] :

- « Ce protocole fonctionne-t-il? Peut-on le faire fonctionner?
- Que fait exactement ce protocole?
- Ce protocole nécessite-t-il plus d'hypothèses qu'un autre protocole?
- Ce protocole fait-il quelque chose d'inutile? »

La logique BAN se focalise sur les croyances des entités impliquées dans le protocole et sur l'évolution de ces croyances au fur et à mesure que des messages sont échangés entre les différentes entités [23].

« Scyther » est un outil développé en 2008 par le chercheur Cremers, dont le but est d'effectuer la vérification, la falsification et l'analyse des protocoles de sécurité [24]. C'est un logiciel gratuit

disponible sur les plateformes Windows, Linux et Mac OS X. Cet outil peut être utilisé pour vérifier si les propriétés de sécurité revendiquées par un protocole sont correctes. Il peut également servir à générer automatiquement des propriétés de sécurité pour un protocole et à les vérifier. *« Scyther »* supporte uniquement le langage *« Security Protocol Description Language » (SPDL)*. Ainsi, pour évaluer notre protocole avec « Scyther », nous devons le traduire en SPDL.

3.1.3 Évaluation des performances

L'évaluation des performances consiste à démontrer l'applicabilité du protocole proposé dans un environnement client mobile/serveur limité par des contraintes de capacité de calcul, d'espace mémoire et de bande passante. Pour évaluer ces performances, nous nous référons aux trois indicateurs suivants : le *coût de calcul*, le *coût de mémoire* et le *coût de communication*.

Le *coût de calcul*, exprimé en *millisecondes*, est le temps nécessaire au client mobile et au serveur pour exécuter les différentes opérations du protocole. Cet indicateur permet d'évaluer la capacité du client mobile à réaliser les tâches qui lui sont assignées dans un temps raisonnable. De plus, la durée totale du protocole d'authentification doit être commode pour les utilisateurs.

Le *coût de mémoire*, exprimé en *octets*, correspond à la quantité d'espace mémoire nécessaire pour sauvegarder les données permanentes du protocole, comme la clé publique du serveur. Il est important de mentionner que l'espace mémoire utilisé au niveau du serveur n'est pas évalué, car l'on suppose qu'il possède une capacité de stockage très grande par rapport à la quantité d'information qu'il devra traiter et sauvegarder.

Le *coût de communication*, exprimé en *octets*, mesure la taille des données transmises sur le canal de communication. L'étude de la taille des données transmises sur le canal de communication a pour but d'évaluer l'utilisation des ressources du réseau par le protocole proposé. Pour un protocole efficace, la taille des données transmises sur le réseau doit être limitée autant que possible.

3.2 Volet 2 : Conception du système de détection des attaques DDoS

Les attaques DDoS par inondation consistent à envoyer un nombre élevé de requêtes à un serveur dans le but d'épuiser ses ressources [81]. En conséquence, le serveur n'est plus en mesure de répondre aux requêtes des clients légitimes. Les effets des attaques DDoS sont néfastes pour une

organisation. Ces effets sont entre autres l'arrêt du service, les pertes économiques et la réduction de la réputation de l'organisation [82].

La prolifération des objets connectés constitue une aubaine pour les cybercriminels. En effet, ces objets connectés sont caractérisés par : une connectivité continue à Internet, une absence de protocole de sécurité élaboré et les mots de passe facilement exploitables [83]. D'ailleurs, l'attaque DDoS *Mirai* s'est servie des objets connectés (e.g., les caméras de vidéosurveillance) [20].

Pour faire face aux attaques DDoS, notre deuxième objectif spécifique est de concevoir un système de détection des attaques DDoS menées contre des serveurs. Cet objectif est atteint dans l'article intitulé *« An Online Entropy-based DDoS Flooding Attack Detection System with Dynamic Threshold »* et présenté au chapitre 5.

3.2.1 Approche de détection

Dans la littérature, il existe deux grandes approches pour détecter les attaques DDoS : l'approche basée sur la signature et l'approche basée sur l'anomalie [84]. L'approche basée sur la signature, consiste à analyser le trafic réseau et à comparer ses caractéristiques avec celles des attaques connues. De ce fait, l'approche basée sur la signature n'est pas en mesure d'identifier les attaques inconnues.

Dans le système de détection proposé, nous adoptons l'approche basée sur l'anomalie. Cette approche consiste à analyser le profil du trafic réseau et à le comparer au profil préétabli du trafic réseau légitime.

Dans l'approche basée sur l'anomalie, nous distinguons deux grandes familles : l'approche basée sur les algorithmes d'intelligence artificielle et l'approche basée sur les statistiques [85]. L'approche basée sur les algorithmes d'intelligence artificielle présente les inconvénients suivants : une consommation élevée des ressources de calcul, un délai de détection élevé et un taux de faux positifs élevé [85] [86]. L'inconvénient majeur de l'approche basée sur les statistiques, est la difficulté à fixer un seuil optimal [85].

Dans notre système de détection, nous optons pour l'approche basée sur les statistiques, en utilisant l'entropie de l'information, plus précisément l'entropie de Shannon [25]. Pour faire face à la difficulté à fixer un seuil optimal, nous proposons un algorithme de seuil dynamique basé sur le

théorème de Chebyshev [87]. En outre, l'approche basée sur les statistiques permet une détection précise et rapide des attaques DDoS [85] [86].

3.2.2 Ensemble de données

Une limite de certains systèmes de détection des attaques DDoS existants est l'évaluation à partir d'un trafic légitime et un trafic d'attaque provenant de deux ensembles de données différents [88] [89]. En effet, la fusion de deux ensembles de données générés à partir des réseaux ayant des caractéristiques différentes peut influencer les résultats [26].

L'ensemble de données que nous utilisons pour la validation de notre système de détection est *CICIDS2017* du *« Canadian Institute for Cybersecurity »* [28]. Cet ensemble de données contient à la fois, le trafic réseau légitime et le trafic d'attaque. La capture du trafic s'étend sur une période de cinq jours, allant du 3 juillet 2017 au 7 juillet 2017. La simulation du comportement de vingt-cinq utilisateurs permet de générer le trafic légitime. Les protocoles du trafic légitime sont : *« HyperText Transfer Protocol » (HTTP), « HyperText Transfer Protocol Secure » (HTTPS), « File Transfer Protocol » (FTP), « Secure Shell » (SSH)* et les protocoles de courriel. Le trafic d'attaque présent dans *CICIDS2017* est généré avec l'outil *« Low Orbit Ion Canon » (LOIC).*

3.2.3 Évaluation des performances

Pour évaluer les performances du système de détection proposé, nous utilisons deux méthodes spécifiques. La première méthode d'évaluation se fait à travers des simulations et la seconde, à travers l'ensemble de données *CICIDS2017* [28].

Pour les simulations, nous implémentons d'abord le système de détection en utilisant le langage de programmation *Python*. Ensuite, nous concevons et configurons un réseau étendu sur le simulateur réseau *« Graphical Network Simulator-3 » (GNS3)* [27]. Le réseau contient à la fois des clients légitimes, des clients malveillants et des serveurs. Le trafic légitime est constitué des requêtes *HTTP* générées avec l'outil *httperf* [90]. En ce qui concerne le trafic d'attaque, nous utilisons l'outil *hping3* [91] pour générer les attaques DDoS.

En ce qui concerne l'évaluation avec l'ensemble de données *CICIDS2017*, nous considérons le trafic de la journée du 7 juillet 2017, qui contient à la fois le trafic légitime et le trafic d'attaque.

Les indicateurs de performance retenus pour l'évaluation sont [92] : le *taux de détection*, le *taux de faux positif*s, la *précision du système*, la *précision globale* et le *délai de détection*.

Le *taux de détection* est le rapport entre le nombre de connexions d'attaque correctement identifiées et le nombre total de connexions d'attaque. Il valide l'efficacité du système de détection.

Le *taux de faux positifs* est le rapport entre le nombre de connexions légitimes identifiées comme connexions d'attaque et le nombre total de connexions légitimes. Il mesure la capacité du système de détection à distinguer le trafic légitime du trafic d'attaque.

La *précision du système* est le rapport entre le nombre de connexions d'attaque correctement identifiées et le nombre total de connexions d'attaque reportées. Elle mesure la qualité du système de détection.

La *précision globale* est la proportion des connexions correctement catégorisées (i.e., connexions légitimes ou connexions d'attaque). Elle permet de comparer les performances de différents systèmes de détection.

Le *délai de détection* est la différence entre le temps de détection de l'attaque et le temps de réception de la première connexion d'attaque. Il mesure la rapidité du système à détecter les attaques.

3.3 Volet 3 : Conception du cadre pour l'évaluation de la sécurité des applications de banque mobile

Ce troisième volet aborde le problème de vulnérabilité dans les applications mobiles. Nous nous focalisons sur les applications de banque mobile, car c'est un moyen de plus en plus utilisé pour accéder aux services bancaires. 65 % des Canadiens ont utilisé une application de banque mobile en 2021, contrairement à 56 % en 2018 [93]. En outre, les applications de banque mobile manipulent les données financières qui sont très sensibles.

Le troisième objectif spécifique de cette thèse est de proposer un cadre pour l'évaluation de la sécurité des applications de banque mobile. Cet objectif est atteint dans l'article intitulé *« A Framework for Security Assessment of Android Mobile Banking Applications »* et présenté au chapitre 6.

3.3.1 Évaluation du cadre

Le cadre proposé comprend un ensemble de vingt-six critères, destinés à l'évaluation de la sécurité des applications Android de banque mobile. Pour évaluer ce cadre, nous définissons au préalable les requis qu'un cadre doit satisfaire. Pour définir les requis, nous nous sommes inspirés des travaux de Wang et Wang [94].

Wang et Wang [94] conçoivent un cadre composé de douze critères pour l'évaluation des protocoles d'authentification à deux facteurs. Cependant, les critères contenus dans notre cadre servent à évaluer la sécurité des applications Android de banque mobile.

Nous distinguons trois requis à savoir, la *non-redondance*, la *non-ambiguïté* et la *complétude*. La *non-redondance* stipule que deux critères ne doivent pas avoir la même interprétation et aucun critère ne devrait être inclus dans un autre. La *non-ambiguïté* désigne le fait que chaque critère doit avoir une interprétation unique dans le contexte de l'évaluation de la sécurité des applications Android de banque mobile. La *complétude* renvoie à la couverture d'un large éventail des vulnérabilités des applications Android de banque mobile. Le modèle de référence utilisé, est la liste des dix risques de sécurité les plus courants liés aux applications mobiles publiée par *« Open Web Application Security Project » (OWASP)* [95].

3.3.2 Étude de cas

L'étude de cas consiste à évaluer des applications Android de banque mobile du marché canadien, sur la base d'un sous-ensemble des critères définis dans notre cadre.

La première étape est la collecte des applications à évaluer. À partir du Google Play Store, nous téléchargeons et installons sept applications de banque mobile sur un téléphone intelligent.

La seconde étape est la configuration de l'environnement d'évaluation et l'évaluation proprement dite. Nous réalisons l'analyse dynamique qui consiste à étudier le comportement de l'application de banque mobile en fonctionnement. Nous nous servons de l'outil *« Nogotofail »* [96] pour générer les attaques de l'homme du milieu contre les applications de banque mobile.

3.4 Synthèse

En somme, les réalisations de cette thèse, présentées dans les trois chapitres suivants, nous permettent d'atteindre les objectifs énoncés à la section 1.3. Le premier objectif spécifique est

atteint à travers le volet 1, le deuxième objectif spécifique à travers le volet 2 et le troisième objectif spécifique à travers le volet 3. Le quatrième objectif spécifique quant à lui, est intégré dans chacun des volets 1, 2 et 3. Ces différents objectifs spécifiques concourent à atteindre l'objectif principal qui est l'amélioration de la sécurité dans un environnement client mobile/serveur.

CHAPITRE 4 ARTICLE 1: A NEW MUTUAL AUTHENTICATION AND KEY AGREEMENT PROTOCOL FOR MOBILE CLIENT – SERVER ENVIRONMENT

Loïc D. Tsobdjou, Samuel Pierre, *Senior Member, IEEE*, and Alejandro Quintero, *Senior Member, IEEE*

Department of Computer and Software Engineering, École Polytechnique Montréal, Montreal, QC H3T 1J4, Canada

E-mail: loic.tsobdjou-dongmo@polymtl.ca; samuel.pierre@polymtl.ca; alejandro.quintero@polymtl.ca.

Revue : Accepté et publié dans le journal *IEEE Transactions on Network and Service Management*, volume 18, numéro 2, pages 1275-1286, juin 2021.

Abstract

Mobile devices are becoming an essential part of many users' lives. Users exchange sometimes very sensitive data with remote servers. This raises a security problem in terms of the confidentiality and integrity of these data, and users' privacy. Mutual authentication protocols allow a user and a server to confirm each other's legitimacy and share a session key to encrypt subsequent communications. Several protocols have been proposed to achieve this goal. However, these have certain weaknesses, such as impersonation, lack of anonymity, the use of additional hardware, and the synchronization problem associated with the use of timestamps. In this paper, we propose a mutual authentication protocol based on elliptic curve cryptography for mobile client – server environments, which addresses the above problems. This protocol is intended to be lightweight as it is designed for resource constrained mobile devices. Moreover, we present a formal and informal analysis of the security of the proposed protocol. This latter has security attributes, such as session key security, perfect forward secrecy, user anonymity, resistance to impersonation, replay and insider attacks. Performance evaluation shows that we outperform similar protocols. Therefore, the proposed protocol is secure, efficient and suitable for mobile environments.

Keywords: Authentication, elliptic curve cryptography, mobile application, security.

4.1 Introduction

The design and deployment of current telecommunications networks place particular emphasis on the concept of mobility. This allows users to access a variety of services regardless of their geographical location. In addition, we notice a strong use of mobile devices (smartphones, tablets…) to perform daily tasks and access the Internet. For example, Statistics Canada reports that in 2018, 88% of Canadians owned a smartphone [1]. This trend will become even more pronounced with the advent of fifth-generation mobile networks, which will offer an ultra-dense network and higher speeds. Service providers are taking advantage of this situation by offering users innovative services for mobile devices, such as mobile payment.

However, these mobile devices are characterized by relatively short battery life, limited computing capacity for resource-intensive applications and low memory space [7], [8]. A new trend is to transfer the execution of some or all the resource-intensive applications to remote servers with larger resources. In this model, the exchange of user data between the mobile device and the server over wireless networks leads to a fundamental challenge for data security, in terms of confidentiality, integrity and user privacy. The mechanisms proposed to deal with this security problem must meet a set of requirements [16]:

- A low computational load because mobile devices are limited in resources.
- High security since messages are transmitted over unsecured wireless networks.
- Protection of user privacy.

To secure communications between a mobile device and a remote server, there are a variety of mutual authentication protocols in the literature. Nevertheless, some of these protocols have security weaknesses, such as impersonation, vulnerability to replay attacks, and failure to respect the anonymity and untraceability of users. Others are resource-intensive, making them inappropriate for resource-limited mobile devices. A research challenge is to propose a model that combines a high level of security with low resource usage of the mobile device. Our goal is to design a mutual authentication protocol between a mobile device and a remote server that is secure, efficient and suitable for mobile environments. This protocol will have to meet security properties,

such as user anonymity and untraceability, session key security, perfect forward secrecy, resistance to man in the middle, replay, impersonation and insider attacks.

The contributions of this paper are summarized as follows:

- We propose a mutual authentication and key agreement protocol for mobile client – server environment, based on elliptic curve cryptography (ECC) [22]. This protocol is resistant to known attacks including replay attack, parallel session attack, de-synchronization attack, offline password guessing attack, impersonation attack, man in the middle attack and insider attack. Moreover, it satisfies different attributes, such as session key security, perfect forward secrecy, user anonymity and untraceability, mutual authentication and key agreement.
- We use the elliptic curve integrated encryption scheme (ECIES) [22] to eliminate the assumption of the presence of a secure channel during the registration phase.
- We use the subscriber identity module (SIM) to cope with the requirement for additional hardware as is the case with smart card-based schemes. Indeed, the smart card is an external hardware that can be costly and not always convenient for the end user (one more card in the user's wallet, risk of loss). However, the SIM is a tamper-resistant material [97] that is directly integrated into the mobile device.
- We formally demonstrate the security of the proposed protocol using the Burrows-Abadi-Needham (BAN) logic [23] and the Scyther tool [24].
- We conduct a performance analysis of the proposed protocol. We also compare it with other similar works using a set of criteria defined by a third party [94]. The results show that our protocol has a better trade-off between security and performance.

The rest of this paper is organized as follows. We will critically review the literature in Section 4.2. In Section 4.3, we will describe the proposed protocol. Then, we will discuss security and performance in Sections 4.4 and 4.5. We will finally conclude in Section 4.6.

4.2 Related Work

Existing mutual authentication protocols can be categorized according to the cryptographic techniques used, i.e., asymmetric cryptography [59], [60], [61], symmetric cryptography [63], [64],

[65], [66], elliptic curve cryptography (ECC) [98], [99], [68], [67], [100], [101], [102], [69], [70], [103], [104] and paring-based cryptography [16], [71], [72], [73].

Fan *et al.* [59] present a protocol based on the universal second factor, to ensure the security of mobile payment systems. This protocol can identify and rejecting counterfeit servers and clients. Purnomo *et al.* [60] propose a public key infrastructure (PKI)-based scheme to enhance the security of mobile payment systems. However, PKI-based systems are resource-intensive and not suitable for mobile devices. Munivel and Kannammal [61] propose a scheme for mobile cloud computing (MCC) environment, that uses a trusted third party as authentication server.

Dey *et al.* [63] propose a mutual authentication protocol between a mobile client and a cloud server. The scheme uses symmetric key cryptography and employs dynamic encryption keys to minimize key predictability. Dey *et al.* [64], [65] and Donald and Arockiam [66] propose schemes to secure communications in MCC environment.

Qiu *et al.* [98] propose a lightweight two-factor authentication and key agreement protocol based on ECC. The researchers claim that their protocol can resist common attacks, including privileged insider attack, replay attack, stolen verifier attack, off-line password guessing attack, key-compromise user impersonation attack and server impersonation attack. Moreover, their protocol has attributes, such as user anonymity and perfect forward secrecy. Nevertheless, anyone can corrupt the secret value stored on the smart card during the password updating part, since no verification is done on the identity and the password provided by the user. Kumari *et al.* [99] present a two-factor mutual authentication scheme for smart cards based on ECC. The protocol consists of the initialization, registration, and authentication phases. During the authentication phase, the user's identifier is not sent to the server, thus the server has no way of identifying the user. Mo *et al.* [68] describe a secure and efficient mutual authentication protocol between the mobile device and the cloud server in an MCC environment. The integrity of the data transmitted between the mobile client and the server during the registration phase is not considered. Mo *et al.* [67] expose an ECC-based authenticated key exchange protocol in client – server environment. The researchers claim that their scheme meets the requirements of mutual authentication, anonymity, resistance to replay attack, insider attack and impersonation. Xie *et al.* [100] suggest an authenticated key exchange protocol. This protocol must support user anonymity and untraceability and resist smart card loss attack. The researchers use ECC to propose a lightweight protocol suitable for mobile

communications. Luo *et al.* [101] present a mutual authentication scheme between a user and a remote server based on ECC. This scheme consists of five phases: initialization, registration, authentication, password change and smart card revocation. Qi and Chen [102] describe an ECC-based authenticated key exchange protocol for mobile environments. The user's password is submitted in plaintext to the server during the registration phase, making the protocol vulnerable to insider attack. Chen *et al.* [69] propose an ECC ID-based mutual authentication scheme for MCC environment. The user is authenticated thanks to his identifier and password. Xiong *et al.* [70] propose a mutual authentication scheme between a mobile device and a cloud server in an MCC environment. However, the model places great importance on a trusted authority in the initialization and registration phases. This constitutes a point of failure of the system. Sun *et al.* [103] present a mutual authentication and key exchange protocol for a mobile client – server environment. This protocol must resist insider attack and guarantee the properties of anonymity and perfect forward secrecy. Nevertheless, this protocol is vulnerable to replay attack. Farash and Attari [104] describe a mutual authentication and key exchange protocol based on ECC for a mobile client – server environment. The user authenticates with his user identifier and private key, whereas the server authenticates with its private key. However, this protocol does not guarantee user anonymity.

Jegadeesan *et al.* [71] propose an anonymous mutual authentication protocol between a mobile user and a service provider. The proposed protocol places great importance on a trusted authority in the registration phase. This constitutes a point of failure of the system. Irshad *et al.* [72] present an authentication scheme using pairing-based cryptography in a multi-server MCC environment. The use of a registration authority represents a point of failure in the system. Olakanmi and Oke [73] propose a mutual authentication protocol that respects privacy and addresses the security challenges of MCC's services. The proposed scheme combines user's voice signature with cryptographic operations. The use of a trusted authority represents a point of failure in the system. Tsai and Lo [16] present an anonymous authentication scheme using pairing-based cryptography for distributed MCC services. The researchers claim that their protocol supports mutual authentication, key exchange, and user untraceability. As a shortcoming, we have the misuse of the fingerprint since the collection of the biometric parameter does not always provide the same value. Furthermore, some researchers have revealed that this protocol is vulnerable to server spoofing attack [75], [76], [77].

Gope and Das [79] and Roy *et al.* [80] propose two different schemes using cryptographic hash functions and exclusive or (XOR) operations to secure communications in MCC environments. Qiu *et al.* [105] present a three-factor authentication and key agreement protocol based on extended chaotic-maps for mobile lightweight devices. The three factors used in this protocol are password, smart card and biometrics. The protocol has better security compared to other similar works. On the other hand, the computational cost and communication cost are higher. Three-factor authentication provides an additional layer of security over two-factor authentication. Nevertheless, it raises new issues. First, the third factor increases the computational cost and the storage cost. This is not relevant for resource constrained mobile devices. Second, the use of biometrics as a third factor introduces security issues specific to biometric authentication. The main issue is the presence of false positives and false negatives. Third, three-factor authentication can be cumbersome from an end-user perspective. In fact, the user must provide the password, the smart card and the biometrics to be authenticated. In addition, the mobile device must be equipped with a biometric sensor and a smart card reader.

Some recurring shortcomings are noticeable in the analyzed protocols.

- Many schemes assume the presence of a secure channel between the mobile device and the server during the registration phase, which is not always the case [16], [61], [64], [65], [66], [98], [99], [67], [100], [101], [102], [69], [70], [103], [104], [71], [72], [79]. We address this limitation by using ECIES [22], which guarantees the confidentiality and integrity of the data.
- Some protocols require additional hardware [16], [59], [98], [99], [68], [100], [101], [102], [69], [72], [79]. This can be costly and not always convenient for the end user.
- In some schemes, secret data are stored in the internal memory of the mobile device [16], [61], [63], [64], [65], [66], [68], [70], [103], [104], [73], [80]. This makes them vulnerable to malware attacks. We tackle this problem by leveraging the SIM, which is a tamper-resistant material [97]. Although it can be considered additional hardware, the SIM is convenient for end-users as it is directly integrated into the mobile device.
- The use of timestamping in some protocols [99], [100], [101], [104] requires synchronization between all entities in the system. In our protocol, we use the random nonce to deal with the replay attack.

In this paper, we propose a mutual authentication and key agreement protocol for mobile client – server environment. The proposed protocol builds on ECC to meet security and efficiency requirements. Existing schemes assume the presence of a secure channel between the mobile device and the server during the registration phase. We address this limitation by using the elliptic curve integrated encryption scheme (ECIES), which guarantees the confidentiality and integrity of the data. Smart card-based schemes require additional hardware. In fact, the smart card is additional hardware that is not always convenient for the end user. Furthermore, the mobile device must be equipped with a smart card reader. In our protocol, we use the SIM, since it is directly integrated into the mobile device. The security analysis of the proposed protocol shows that it can resist known attacks including replay attack, parallel session attack, de-synchronization attack, offline password guessing attack, impersonation attack, man in the middle attack and insider attack. Moreover, it satisfies different attributes, such as session key security, perfect forward secrecy, user anonymity and untraceability, mutual authentication and key agreement. The performance analysis shows that the proposed protocol has a better trade-off between security and performance, compared to other similar works.

4.3 Proposed Protocol

4.3.1 Elliptic Curve Cryptography

ECC is a special case of public key cryptography, using elliptic curves defined over a finite field $\mathbb{F}$ [14], [15]. An elliptic curve E over a field $\mathbb{F}$ is the set of points $(x, y) \in \mathbb{F} * \mathbb{F}$ satisfying (1) along with the point at infinity, denoted O.

$$y^2 = x^3 + ax + b, \quad (1)$$

Where $a, b \in \mathbb{F}$ and $4a^3 + 27b^2 \neq 0$.

From (1), we can derive $y = \pm\sqrt{x^3 + ax + b}$. Instead of representing a point with its coordinates (x, y), we can represent it with x and the sign of y. $(x, sign\ of\ y)$ is the compressed form of a point of the elliptic curve. All the above operations are carried out over the finite field $\mathbb{F}$.

For cryptographic applications, the curves used are elliptic curves defined over a finite field $\mathbb{F}_p$, where $\mathbb{F}_p$ is the set of integers modulo p, and p is a large prime number. The security of ECC is based on the elliptic curve discrete logarithm problem (ECDLP), which is hard [14], [15]. The

definition of this problem is as follows. Let E be an elliptic curve over the finite field $\mathbb{F}_p$ and P, Q two points of E. The ECDLP consists in finding an integer n such that $Q = n.P$.

The elliptic curve Diffie-Hellman problem (ECDHP), which is also hard, consists in finding $n.m.P$, given $n.P$ and $m.P$, where n, m are two integers and P is a point of E.

We choose ECC because the proposed protocol is intended for resource-constrained mobile devices. Indeed, ECC has a better efficiency in terms of security and resource consumption compared to other public key cryptographic techniques. According to the National Institute of Standards and Technology (NIST) [12], ECC achieves a 112-bit security level with a 224-bit key size, while Diffie-Hellman (DH) and Rivest, Shamir, Adleman (RSA) require a 2048-bit key for the same level of security.

4.3.2 Adversary Model

To design an effective security protocol, it is important to define an adversary's capabilities. In this work, we adopt the Dolev-Yao [106] and Canetti-Krawczyk [107] adversary models.

- The messages exchanged pass through an unsecured communication channel. Thus, an adversary can intercept, delete, modify, resend or insert messages over the network.
- An adversary can be an insincere user of the system. Therefore, he can initiate a conversation or be a receiver to any user.
- An adversary can make session-state reveal query, session-key query and party corruption.

4.3.3 Notations

The notations used in the proposed protocol are listed in Table 4.1.

Table 4.1 Notations

Symbol	Description
$\boldsymbol{MC}$	Mobile Client
$\boldsymbol{S}$	Server
$\boldsymbol{p}, \boldsymbol{q}$	Two large prime numbers
$\mathbb{F}_{\boldsymbol{p}}$	Field of integers modulo p
$\boldsymbol{E}$	Elliptic curve over $\mathbb{F}_p$
$\boldsymbol{P}$	A point of E of order q
$\boldsymbol{H}$	Cryptographic hash function
$\mathbb{Z}_{\boldsymbol{q}}^*$	$\{1, 2, \dots, q-1\}$
$\boldsymbol{d}$	Server private key
$\boldsymbol{Q}$	Server public key
$\boldsymbol{c}$	Server secret key for symmetric encryption
$\boldsymbol{ID}, \boldsymbol{PW}$	User identifier and password respectively
$\boldsymbol{PID}$	User pseudo-identifier
$\boldsymbol{h}$	Hash of the user identifier, $h = H(ID)$
$\boldsymbol{K_S}, \boldsymbol{K_{MC}}$	Session keys computed by the server and the mobile client respectively
$\boldsymbol{K}$	Session key ($K = K_S = K_{MC}$)
$\boldsymbol{r}$	Random number generated by the mobile client to protect user password
$\boldsymbol{Z}$	Protected user password
$\boldsymbol{B}$	Protected user password encrypted with server secret key
$\boldsymbol{n}, \boldsymbol{m}$	Ephemeral random numbers generated by the mobile client and the server respectively
$\boldsymbol{N}, \boldsymbol{M}$	Ephemeral public keys computed by the mobile client and the server respectively

Symbol	Description
$\boldsymbol{A_{MC}}, \boldsymbol{A_S}$	Values computed by the mobile client and the server respectively to prove their legitimacy
$\boldsymbol{A_S^*}$	Value computed by the mobile client to verify the legitimacy of the server
$\boldsymbol{A_{MC}^*}$	Value computed by the server to verify the legitimacy of the mobile client
$\boldsymbol{CTR}$	Counter for the number of failed authentication attempts
$\boldsymbol{CTR_{max}}$	Maximum number of failed authentication attempts
$\|\|$	Concatenation operation
$\oplus$	Exclusive OR operation
.	Elliptic curve point multiplication

4.3.4 Protocol Specification

Figure 4.1 illustrates a mobile client – server communication model. Since the communication channel is insecure, the system is exposed to attacks from a malicious entity. The purpose of a mutual authentication protocol is to provide a means of ensuring the confidentiality, integrity, and non-repudiation of data in transit over an insecure communication channel.

The system is mainly composed of two actors: the server and the mobile client.

- The server is the entity that provides services to clients.
- The mobile client is an application that runs on a mobile device and acts as a client; that is, it requests the services of a server. It is worth noting that under our protocol, the mobile device on which the mobile application runs must be equipped with the SIM. This module will serve to store the secret parameters of the mobile client in a secure manner.

We use the SIM for two main reasons. First, the SIM is a tamper-resistant material [97], which provides hardware security to protect the proposed protocol from malware attacks. Second, it eliminates the need for additional hardware that would be costly and inconvenient for the end user. The use of SIM is a constraint of the proposed protocol. Moreover, this protocol can be implemented on a mobile device with a built-in secure element. Nevertheless, the proposed

protocol remains practical. Indeed, several mobile devices can be equipped with a SIM. In addition, the embedded subscriber identity module (eSIM), which is the future evolution of the SIM, will be directly integrated into the mobile device when it is manufactured.

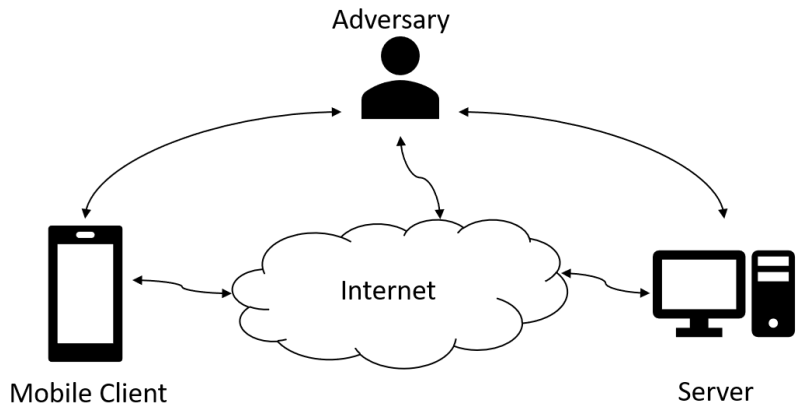

Figure 4.1 Mobile client – server communication model

The proposed protocol consists of three phases: the initialization phase, the user registration phase and the authentication phase.

4.3.4.1 Initialization phase

The initialization phase is carried out by the server to define the public parameters of the system. This phase proceeds as follows.

1. S chooses a large prime number p and an elliptic curve E over the field $\mathbb{F}_p$.
2. S chooses a point $P \in E$ of order q.
3. S chooses a hash function H.
4. S generates a random number $d \in \mathbb{Z}_q^*$ as its private key and computes its public key $Q = d.P$.
5. S publishes the system parameters $\{p, E, P, q, H, Q\}$.

4.3.4.2 User Registration Phase

During this phase, a mobile client requesting the services of the server must be registered by providing its identifier and password. The different steps of this phase are as follows.

1. The user chooses an identifier ID and a password PW.
2. The SIM generates a random number $r \in \mathbb{Z}_q^*$ and then computes $Z = H(PW||r)$, $h = H(ID)$, and $M_0 = E_Q(h||Z)$; where $E_Q(X)$ represents the encryption of X with the public key Q. The encryption algorithm used is ECIES which ensures the confidentiality and integrity of the message.
3. *MC* sends the message M_0 to *S*.
4. *S* computes $h||Z = D_d(M_0)$; where $D_d(X)$ represents the decryption of X with the private key d.
5. *S* checks the existence of h in its database. If h already exists, then *S* notifies the *MC* to return another registration request with a different identifier. We return to step 1. If h does not yet exist, then *S* computes $B = I_c(Z), CTR = 0$, where $I_c(Z)$ is the symmetric encryption of Z with the secret key c. *S* saves (h, B, CTR) to its database and notifies the *MC* of the successful registration operation.
6. Upon receipt of the notification, the SIM saves r in its memory.

4.3.4.3 Authentication Phase

The authentication phase is the phase during which the mutual authentication operation between the mobile client and the remote server takes place. In addition, it allows both entities to exchange a session key. This will be used to encrypt future data exchanged during the communication. The summary of the authentication phase is shown in Figure 4.2. The server is first authenticated by the mobile client, which is then authenticated by the server. Both entities perform the following steps.

1. The user enters his identifier ID and password PW.
2. The SIM generates a random number $n \in \mathbb{Z}_q^*$, then computes $N = n.P$, $R = n.Q$, $h = H(ID)$, $PID = h \oplus H(R)$ and sends the authentication request $M_1 = (PID, N)$ to *S* via the *MC*.
3. *S* generates a random number $m \in \mathbb{Z}_q^*$, then computes $M = m.P$, $V = d.N$, $h = PID \oplus H(V)$, $A_S = H(h||M||N||V)$ and sends the message $M_2 = (M, A_S)$ to the *MC*.

4. The SIM computes $A_S^* = H(h||M||N||R)$. If $A_S^* = A_S$, then S is authenticated. Otherwise, the *MC* terminates the procedure.

5. The SIM computes $Z = H(PW||r)$, $A_{MC} = H(Q||M||N||R||Z)$ and sends the message $M_3 = (A_{MC})$ to S via the *MC*.

6. S uses h to find B and CTR corresponding to the *MC* in its database. If $CTR \geq CTR_{max}$, then S terminates the procedure. Otherwise, S computes $Z = J_c(B)$, where $J_c(B)$ is the symmetric decryption of B with the secret key c. $A_{MC}^* = H(Q||M||N||V||Z)$. If $A_{MC}^* = A_{MC}$, then the *MC* is authenticated and S updates $CTR = 0$. Otherwise, S terminates the procedure and increments the *CTR* counter.

7. S computes $K_S = m.N$ and *MC* computes $K_{MC} = n.M$. The shared session key is $K = K_{MC} = K_S$.

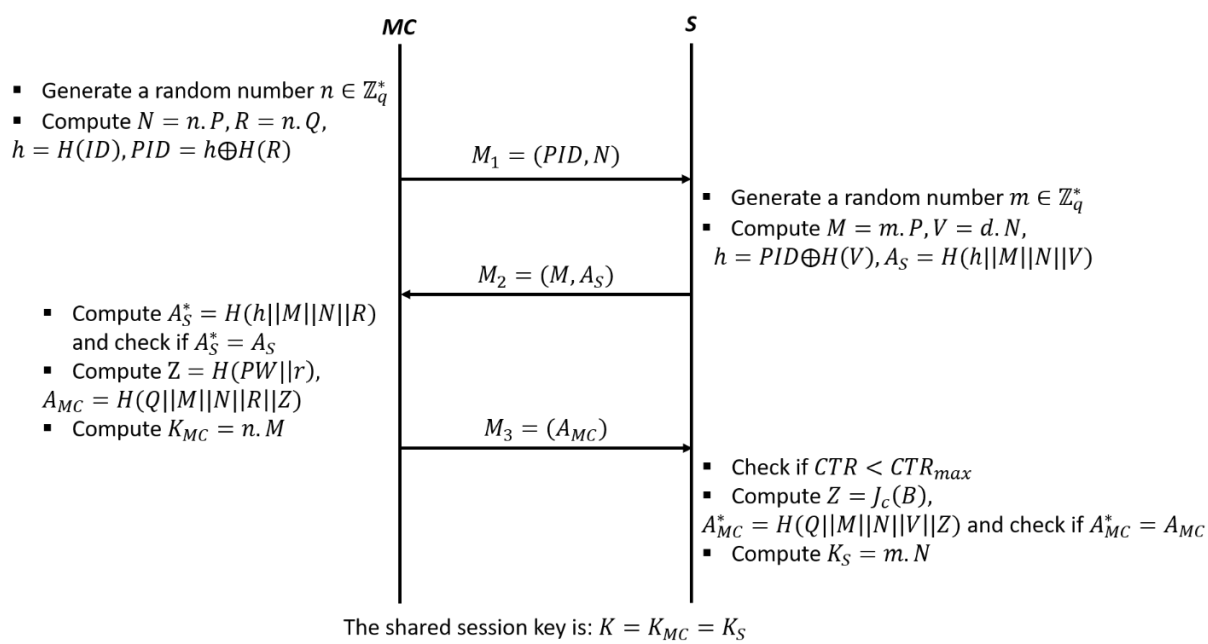

Figure 4.2 Authentication phase

4.4 Security Analysis

4.4.1 Informal Security Analysis

This analysis aims to prove some of the security properties of the proposed protocol.

4.4.1.1 Resistance to Forgery Attack

This attack refers to the possibility for an adversary to retrieve the private key of the server. The public key of the server is $Q = d.P$. An adversary who wants to retrieve the private key of the server must be able to extract d from Q. This amounts to solving the ECDLP. However, this is a hard problem on which the ECC is based.

4.4.1.2 Resistance to Man in the Middle Attack

In this type of attack, an adversary can sniff out the communication channel and modify the messages passing through it. The messages exchanged during the authentication phase are $M_1 = (PID, N)$, $M_2 = (M, A_S)$ and $M_3 = (A_{MC})$. The different possible scenarios are as follows.

- PID or N is changed in message M_1. The $A_S^* = A_S$ check performed by the *MC* will fail and it will terminate the authentication procedure.
- M or A_S is changed in message M_2. The $A_S^* = A_S$ check performed by the *MC* will fail and it will terminate the authentication procedure.
- A_{MC} is changed in message M_3. The $A_{MC}^* = A_{MC}$ check performed by S will fail and it will terminate the authentication procedure.

In all possible scenarios, a possible man in the middle attack is detected.

4.4.1.3 Session Key Security

In this context, an adversary must retrieve the session key from the public data of the system and the messages exchanged on the communication channel. An adversary with this objective should be able to extract the session key $K = m.N = n.M$ from $N = n.P$ and $M = m.P$. This amounts to solving the ECDHP, which is hard.

Furthermore, a new session key is generated for each authentication.

4.4.1.4 Perfect Forward Secrecy

This property ensures that knowledge of the private key of the server d or of the secret parameter Z of the mobile client by an adversary does not reveal previously generated session keys. Without knowledge of the secret random numbers n and m generated in each session by the mobile client

and the server respectively, it is impossible to compute the session key $K = m.N = n.M$. Therefore, the perfect forward secrecy property is ensured by the proposed protocol.

4.4.1.5 Resistance to Impersonation Attack

An adversary may want to impersonate both the mobile client and the server. An adversary masquerading as the legitimate mobile client must be able to come into possession of $Z = H(PW||r)$. However, only the legitimate user knows the password and r is a random number stored in the memory of the SIM.

An adversary masquerading as the legitimate server must be able to get into possession of the legitimate server's private key to compute $V = d.N$. However, according to the property of resistance to forgery attack, this is not possible.

4.4.1.6 User Anonymity and Untraceability

The proposed protocol ensures the anonymity and untraceability of users. Indeed, the real identity of the user is never transmitted over the communication channel. Instead, the information exchanged is $PID = h \oplus H(R)$. To obtain h, an adversary must compute $R = n.Q = d.N$. This is not possible, because n and d are secret parameters of the mobile client and the server, respectively.

Moreover, since n is a random number generated in each session, PID is different in each authentication request $M_1 = (PID, N)$. As a result, an adversary cannot determine whether an authentication request is sent by the same user. Hence the property of untraceability.

4.4.1.7 Mutual Authentication

In the proposed protocol, the mobile client authenticates the server by checking whether $A_S^* = A_S$ and the server authenticates the mobile client by checking whether $A_{MC}^* = A_{MC}$. This is because only the legitimate server that is in possession of d can compute A_S. Similarly, only the legitimate mobile client that is in possession of PW and r can compute A_{MC}.

4.4.1.8 Resistance to Replay Attack

The proposed protocol is resistant to replay attacks, using the concept of random nonce. Indeed, the numbers n and m are randomly generated. This implies that $N = n.P$ and $M = m.P$ are also random values that serve as random nonces. Consider the scenario where messages M_1 and M_3

from a previous protocol run were saved by an adversary. Let us call these messages M_{1_old} and M_{3_old} respectively. The adversary tries a replay attack by sending the message M_{1_old} to the server. The server executes the protocol and sends the message M_2 to the adversary, who in turn sends the message M_{3_old} to the server. The $A_{MC}^{*} = A_{MC}$ check performed by the server will fail, because the value of M contained in message M_{3_old} is different from the value that the server generated in step 3 of the authentication phase. This is due to the randomness of M.

As a result, the proposed protocol is resistant to replay attack.

4.4.1.9 Resistance to Lost SIM Attack

Suppose the SIM is in the possession of an adversary. The latter can then access the secret information r stored in the SIM memory by using the side-channel attack for example. The adversary can guess the user's password PW^* and compute $Z^* = H(PW^*||r)$. To check whether the guessed password is correct, the adversary must successfully complete the authentication phase of the protocol. Nevertheless, after CTR_{max} unsuccessful attempts, the adversary will no longer be able to execute the authentication phase. We consider that the value of CTR_{max} does not allow the adversary to successfully guess the user's password.

In conclusion, even knowing the value of r, an adversary cannot pretend to be the legitimate mobile client.

4.4.1.10 Resistance to Insider Attack

This is an attack led by an adversary who has access to the server's database. Such an adversary will come into possession of the parameters h, B and CTR. However, not knowing the secret key c of the server, it will be impossible for this adversary to compute $Z = J_c(B)$.

4.4.2 Formal Security Analysis

In this section, we conduct a formal security analysis of the proposed protocol using the BAN logic [23] and the Scyther tool [24].

4.4.2.1 Formal Security Analysis with the BAN Logic

Table 4.2 presents some symbols from the BAN logic used in this analysis.

Table 4.2 Symbols of the BAN Logic

Symbol	Description
$\boldsymbol{C} \mid\equiv \boldsymbol{X}$	The entity C believes that the formula X holds
$\boldsymbol{C} \lhd \boldsymbol{X}$	The entity C sees the formula X
$\boldsymbol{C} \Longrightarrow \boldsymbol{X}$	The entity C has complete control over the formula X
$\boldsymbol{C} \mid\sim \boldsymbol{X}$	The entity C once said the formula X
$\#(\boldsymbol{X})$	The formula X is fresh
$\boldsymbol{C} \overset{\boldsymbol{K}}{\leftrightarrow} \boldsymbol{D}$	The entities C and D share a secret key K
$\boldsymbol{C} \overset{\boldsymbol{X}}{\rightleftharpoons} \boldsymbol{D}$	The formula X is a secret shared by the entities C and D
$\overset{\boldsymbol{K}}{\rightarrow} \boldsymbol{C}$	The entity C has a public key K
$\{\boldsymbol{X}\}_{\boldsymbol{K}}$	Encryption of the formula X with key K
$\langle \boldsymbol{X} \rangle_{\boldsymbol{Y}}$	The formula X is combined with the secret Y

The idealized form of the messages exchanged during the authentication phase according to the BAN logic is as follows.

- Message 1: $MC \rightarrow S: \{h\}_{H(n.Q)}, n.P$
- Message 2: $S \rightarrow MC: \langle h, m.P, n.P \rangle_{d.n.P}$
- Message 3: $MC \rightarrow S: \langle Q, m.P, n.P, n.Q \rangle_{Z}$

To demonstrate the security of the proposed protocol with the BAN logic, we need to achieve the following goals.

- Goal 1: $MC \mid\equiv MC \overset{K}{\leftrightarrow} S$
- Goal 2: $S \mid\equiv MC \overset{K}{\leftrightarrow} S$
- Goal 3: $MC \mid\equiv S \mid\equiv MC \overset{K}{\leftrightarrow} S$

- Goal 4: $S \mid\equiv MC \mid\equiv MC \overset{K}{\leftrightarrow} S$

Goals 1 and 2 refer to the fact that each of the entities believes in the session key. Goals 3 and 4 reflect the fact that each entity believes that the other entity believes in the session key.

To analyze the protocol, we make the following assumptions.

- A1: $MC \mid\equiv MC \overset{Z}{\rightleftharpoons} S$
- A2: $MC \mid\equiv \overset{Q}{\rightarrow} S$
- A3: $S \mid\equiv MC \overset{Z}{\rightleftharpoons} S$
- A4: $S \mid\equiv \overset{Q}{\rightarrow} S$
- A5: $MC \mid\equiv S \Longrightarrow m.P$
- A6: $S \mid\equiv MC \Longrightarrow n.P$

The analysis of the protocol is carried out as follows.

MC generates a random number n allows us to deduce:

- D1: $MC \mid\equiv n$
- D2: $MC \mid\equiv \#(n)$
- D3: $MC \mid\equiv \#(d.n.P)$ (deduced from D2)

Message 1 leads to:

- D4: $S \lhd \{h\}_{H(n.Q)}, n.P$

S generates a random number m allows us to deduce:

- D5: $S \mid\equiv m$
- D6: $S \mid\equiv \#(m)$

Message 2 leads to:

- D7: $MC \lhd \langle h, m.P, n.P\rangle_{d.n.P}$
- D8: $MC \mid\equiv MC \overset{d.n.P}{\longleftrightarrow} S$ (deduced from D3 and A2)

- D9: $MC \mid\equiv S \mid\sim h, m.P, n.P$ (deduced from D7 and D8)
- D10: $MC \mid\equiv \#(h, m.P, n.P)$ (deduced from D2)
- D11: $MC \mid\equiv S \mid\equiv h, m.P, n.P$ (deduced from D9 and D10)
- D12: $MC \mid\equiv S \mid\equiv m.P$ (deduced from D11)
- D13: $MC \mid\equiv S \mid\equiv m$ (deduced from D12)
- D14: $MC \mid\equiv m.P$ (deduced from D12 and A5)
- D15: $MC \mid\equiv S \mid\equiv n.P$ (deduced from D11)

Message 3 leads to:

- D16: $S \lhd \langle Q, m.P, n.P, n.Q \rangle_Z$
- D17: $S \mid\equiv MC \mid\sim Q, m.P, n.P, n.Q$ (deduced from D16 and A3)
- D18: $S \mid\equiv \#(Q, m.P, n.P, n.Q)$ (deduced from D6)
- D19: $S \mid\equiv MC \mid\equiv Q, m.P, n.P, n.Q$ (deduced from D17 and D18)
- D20: $S \mid\equiv MC \mid\equiv n.P$ (deduced from D19)
- D21: $S \mid\equiv MC \mid\equiv n$ (deduced from D20)
- D22: $S \mid\equiv n.P$ (deduced from D20 and A6)
- D23: $S \mid\equiv MC \mid\equiv m.P$ (deduced from D19)
- D24: $MC \mid\equiv \#(K)$ (deduced from D2)
- D25: $MC \mid\equiv MC \overset{K}{\leftrightarrow} S$ (deduced from D13 and D24)
- D26: $MC \mid\equiv S \mid\equiv MC \overset{K}{\leftrightarrow} S$ (deduced from D13 and D15)
- D27: $S \mid\equiv \#(K)$ (deduced from D6)
- D28: $S \mid\equiv MC \overset{K}{\leftrightarrow} S$ (deduced from D21 and D27)
- D29: $S \mid\equiv MC \mid\equiv MC \overset{K}{\leftrightarrow} S$ (deduced from D21 and D23)

Goals 1, 2, 3 and 4 are achieved through deductions D25, D28, D26 and D29, respectively. This leads to the conclusion that the two entities have authenticated each other and have a shared session key.

4.4.2.2 Formal Security Analysis with the Scyther Tool

We analyze the security of the proposed protocol formally with the Scyther tool. For this purpose, the protocol was modeled in security protocol description language (SPDL) as follows.

```
//Protocol Description
hashfunction H;
const P;
//Point Multiplication is simply a hash function
hashfunction mult;
protocol	MutualAuth(MC, S) {
	//Z is defined as a secret shared between MC and S
	macro Z = k(MC, S);
	role MC {
		fresh n: Nonce;
		var M: Ticket;
		send_1(MC, S, {H(MC)}pk(S), mult(n, P));
		recv_2(S, MC, M, {H(MC), M, mult(n, P)}sk(S));
		send_3(MC, S, {pk(S), M, mult(n, P), mult(n, pk(S))}Z);
		claim(MC, SKR, mult(n, M));
		claim(MC, Secret, n);
		claim(MC, Alive);
		claim(MC, Weakagree);
		claim(MC, Niagree);
		claim(MC, Nisynch);
	}
	role S {
		fresh m: Nonce;
		var N, R: Ticket;
		recv_1(MC, S, {H(MC)}pk(S), N);
```

```
        send_2(S, MC, mult(m, P), {H(MC), mult(m, P), N}sk(S));
        recv_3(MC, S, {pk(S), mult(m, P), N, R}Z);
        claim(S, SKR, mult(m, N));
        claim(S, Secret, m);
        claim(S, Alive);
        claim(S, Weakagree);
        claim(S, Niagree);
        claim(S, Nisynch);
    }
}
```

The result is shown in Figure 4.3. We can conclude that the Scyther tool did not find any attack on the proposed protocol.

Scyther results : verify

Claim				Status	Comments
MutualAuth	MC	MutualAuth,MC1	SKR mult(n,M)	Ok	No attacks within bound
		MutualAuth,MC2	Secret n	Ok	No attacks within bound
		MutualAuth,MC3	Alive	Ok	No attacks within bound
		MutualAuth,MC4	Weakagree	Ok	No attacks within bound
		MutualAuth,MC5	Niagree	Ok	No attacks within bound
		MutualAuth,MC6	Nisynch	Ok	No attacks within bound
	S	MutualAuth,S1	SKR mult(m,N)	Ok	No attacks within bound
		MutualAuth,S2	Secret m	Ok	No attacks within bound
		MutualAuth,S3	Alive	Ok	No attacks within bound
		MutualAuth,S4	Weakagree	Ok	No attacks within bound
		MutualAuth,S5	Niagree	Ok	No attacks within bound
		MutualAuth,S6	Nisynch	Ok	No attacks within bound

Done.

Figure 4.3 Result of the analysis with the Scyther tool

4.5 Performance Analysis

We analyze the performance of the proposed protocol by evaluating three indicators. These are computation cost, memory cost and communication cost. The initialization and user registration phases are one-time phases and are therefore not included in the performance analysis.

As recommended by the NIST [12], we will consider ECC with a 32-byte key size (elliptic curve secp256r1) and SHA-256 as a hash function.

4.5.1 Computation Cost

The computation cost is the time required by the mobile client and the server to perform the various operations of the protocol. We will consider the notations in Table 4.3.

Table 4.3 Notation for Computation Cost

Symbol	Description
$\boldsymbol{T_{RNG}}$	The running time of random number generation
$\boldsymbol{T_{PM}}$	The running time of elliptic curve point multiplication
$\boldsymbol{T_{PA}}$	The running time of elliptic curve point addition
$\boldsymbol{T_{H}}$	The running time of hash function
$\boldsymbol{T_{XOR}}$	The running time of XOR operation
$\boldsymbol{T_{\|\|}}$	The running time of concatenation operation
$\boldsymbol{T_{INV}}$	The running time of modular inversion operation
$\boldsymbol{T_{Mult}}$	The running time of modular multiplication operation
$\boldsymbol{T_{E/D}}$	The running time of symmetric encryption/decryption

The computation cost of the proposed protocol is $8\boldsymbol{T_{||}} + \boldsymbol{T_{XOR}} + \boldsymbol{5T_{H}} + \boldsymbol{T_{RNG}} + \boldsymbol{3T_{PM}}$ on the client side and $7\boldsymbol{T_{||}} + \boldsymbol{T_{XOR}} + \boldsymbol{3T_{H}} + \boldsymbol{T_{RNG}} + \boldsymbol{3T_{PM}} + \boldsymbol{T_{E/D}}$ on the server side.

4.5.2 Memory Cost

This indicator measures the amount of memory required to store the permanent protocol data on the mobile device. It is important to mention that the memory used at the server side is not evaluated, since we assume that it has a large capacity related to the processing and the storage of the big amount of data.

The protocol data permanently stored on the mobile device (including the SIM) are r and Q. The sizes of these data are 32 bytes and 33 bytes, respectively. We consider the representation of the points of the elliptic curve in their compressed form, which gives us a size of 33 bytes for the public key of the server.

Thus, the memory cost of the proposed protocol is **65 bytes**.

4.5.3 Communication Cost

The communication cost represents the amount of information transmitted over the communication channel. During the authentication phase, the mobile device and the server exchange three messages with each other: $M_1 = (PID, N)$, $M_2 = (M, A_S)$ and $M_3 = (A_{MC})$. The sizes of the different parameters are given below:

- Size of the hash function result: 32 bytes.
- Size of the points of the elliptic curve in compressed form: 33 bytes.
- Size of message 1: 32 + 33 = 65 bytes
- Size of message 2: 33 + 32 = 65 bytes
- Size of message 3: 32 bytes

The communication cost is therefore **162 bytes**.

4.5.4 Comparison

To prove the effectiveness of the proposed protocol, we compare its performance with similar protocols from the literature [99], [68], [67], [100], [101], [102].

We consider as symmetric encryption algorithm advanced encryption standard (AES) with a 16-byte key, the size of the identifiers is 32 bytes, the size of the timestamp is 6 bytes as in [99] and the running time of a key derivation function is equal to the running time of a hash function.

Table 4.4 gives us the average running times of the different operations at the server and mobile device side. To obtain these values, we repeated each of the operations 100 times and used the arithmetic average of the different execution times obtained. As the server, we used a computer with an Intel Core i7-6700 3.40 GHz processor, 16 GB RAM and Windows 10 Enterprise 64-bit operating system. As a mobile device, we used a smartphone from Asus, model Z00D, with an Intel Atom Z2560 1.60 GHz processor, 2 GB RAM and Android 5.0.1 operating system. The various operations were programmed using the Java programming language.

Table 4.4 Average Running Time of Various Operations

Running time (in ms)	$\mathbf{T_{E/D}}$	$\mathbf{T_{\|\|}}$	$\mathbf{T_{H}}$	$\mathbf{T_{INV}}$	$\mathbf{T_{Mult}}$	$\mathbf{T_{PA}}$	$\mathbf{T_{PM}}$	$\mathbf{T_{RNG}}$	$\mathbf{T_{XOR}}$
Client side	0.114	0.004	0.053	0.213	0.057	1.039	350.493	7.419	0.089
Server side	0.147	0.006	0.173	0.456	0.038	0.731	32.478	79.560	0.020

Table 4.5 and Figure 4.4 give us a comparison of computation cost, whereas Table 4.6 and Figure 4.5 compare the memory and communication costs.

The comparisons carried out allow us to conclude that the proposed protocol has a better computation cost compared to [99], [68], [67], [100], [101], [102]. With respect to the memory cost, we equal protocol [102] and outperform the others. As for the communication cost, only protocol [67] has a better performance than ours. Nevertheless, we outperform it in terms of computation cost and memory cost.

Finally, we compare our scheme and other related schemes considering the set of evaluation criteria defined in [94]. Table 4.7 presents the results. We fail to meet the “No password verifier-table”, “Password friendly” and “Sound repairability” criteria. However, the proposed scheme is resistant to the insider attack as explained in section 4.4.1.10. According to [94], a scheme is password

friendly if "the password is memorable and can be chosen freely and changed locally by the user". In the proposed protocol, the password is memorable and can be chosen freely by the user. However, the password update needs an interaction with the server. Therefore, we fail to meet the "password friendly" criterion. To satisfy this criterion, we need to store the user password or some derived values of the user password in the SIM. This has two drawbacks. First, memory and computation costs will increase. Second, the scheme will be vulnerable to the smart card loss attack. Indeed, an attacker in possession of the smart card could execute the offline password guessing attack to retrieve the user password or some derived values of the user password stored in the smart card. Moreover, the related works that satisfy the "password friendly" criterion, fail to meet the "No smart card loss attack" criterion [99] [68] [101]. In conclusion, although the change of the user password requires interaction with the server, this is not a security issue. In addition, we save in terms of memory and computation costs, as well as we eliminate the vulnerability to the smart card loss attack. The scheme in [68] has two more security properties than ours. The scheme in [100] has one more security property than ours. However, our scheme is superior in term of performance (computation cost, memory cost and communication cost). In a mobile environment, we should make a trade-off between security and performance.

Table 4.5 Computation Cost Comparison

	[99]	**[68]**	**[67]**	**[100]**	**[101]**	**[102]**	**Ours**
Computation cost on client side	$31T_{\|\|} + 7T_{XOR} + 12T_H + T_{RNG} + 3T_{PM} + 2T_{E/D}$	$19T_{\|\|} + 3T_{XOR} + 7T_H + T_{RNG} + 3T_{PM} + T_{PA}$	$7T_{\|\|} + T_{XOR} + 3T_H + T_{RNG} + 3T_{PM}$	$7T_{\|\|} + 2T_{XOR} + 6T_H + T_{RNG} + 3T_{PM} + T_{PA}$	$21T_{\|\|} + T_{XOR} + 5T_H + T_{RNG} + 6T_{PM} + 2T_{PA} + T_{Mult}$	$7T_{\|\|} + 2T_{XOR} + 5T_H + T_{RNG} + 3T_{PM}$	$8T_{\|\|} + T_{XOR} + 5T_H + T_{RNG} + 3T_{PM}$
Computation cost on server side	$31T_{\|\|} + 4T_{XOR} + 9T_H + T_{RNG} + 3T_{PM} + 2T_{E/D}$	$19T_{\|\|} + T_{XOR} + 6T_H + T_{RNG} + 3T_{PM} + T_{PA}$	$9T_{\|\|} + T_{XOR} + 4T_H + T_{RNG} + 3T_{PM} + T_{INV}$	$8T_{\|\|} + T_{XOR} + 5T_H + 2T_{RNG} + 3T_{PM} + T_{PA} + 2T_{E/D}$	$22T_{\|\|} + T_{XOR} + 5T_H + T_{RNG} + 5T_{PM} + T_{PA} + T_{Mult}$	$6T_{\|\|} + T_{XOR} + 5T_H + T_{RNG} + 3T_{PM}$	$7T_{\|\|} + T_{XOR} + 3T_H + T_{RNG} + 3T_{PM} + T_{E/D}$
Total (in ms)	1239.63	1239.55	1237.39	1318.97	2356.69	1237.29	1237.01

Table 4.6 Memory and Communication Costs Comparison

	[99]	**[68]**	**[67]**	**[100]**	**[101]**	**[102]**	**Ours**
Memory cost (in bytes)	128	131	130	97	163	65	65
Communication cost (in bytes)	390	226	142	270	175	194	162

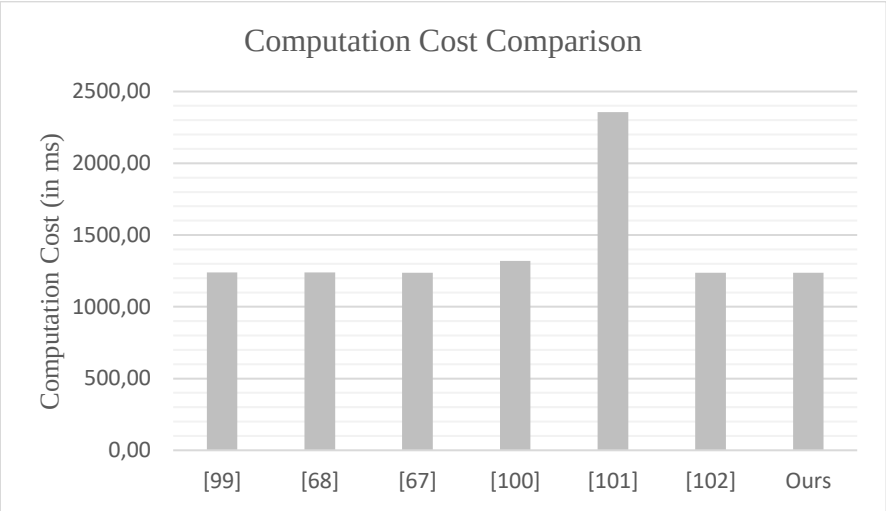

Figure 4.4 Computation cost comparison

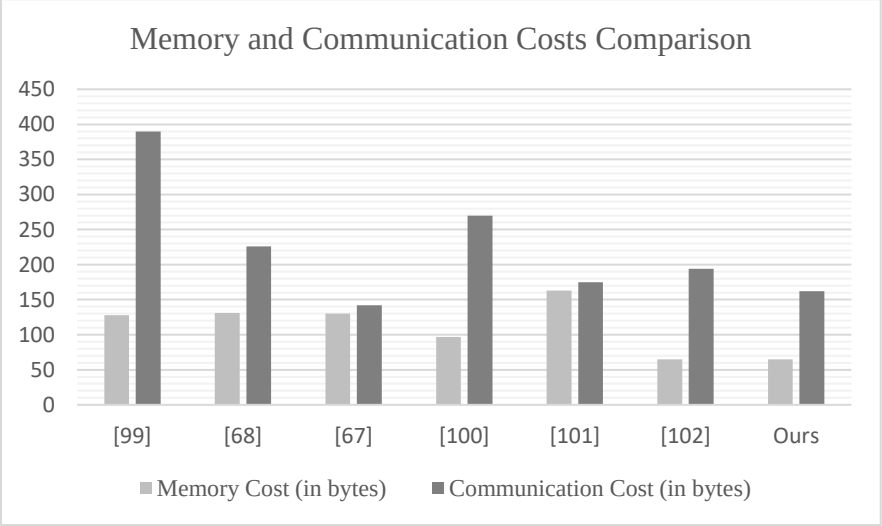

Figure 4.5 Memory and communication costs comparison

Table 4.7 Comparison of Security Properties

Evaluation criteria	**[99]**	**[68]**	**[67]**	**[100]**	**[101]**	**[102]**	**Ours**
No password verifier-table	Y	Y	NA	Y	Y	Y	N
Password friendly	Y	Y	NA	N	Y	N	N
No password exposure	Y	Y	NA	Y	Y	Y	Y
No smart card loss attack	N	N	NA	Y	N	Y	Y
Resistance to known attacks	N	Y	N	Y	N	N	Y
Sound repairability	Y	Y	NA	Y	Y	N	N
Provision of key agreement	Y	Y	Y	Y	Y	Y	Y
No clock synchronization	N	Y	N	N	N	Y	Y
Timely typo detection	Y	Y	NA	Y	Y	Y	Y
Mutual authentication	Y	Y	Y	Y	Y	Y	Y
User anonymity	N	Y	Y	Y	Y	N	Y
Forward secrecy	Y	Y	Y	Y	Y	Y	Y

Y: Attribute is satisfied by the scheme. N: Attribute is not satisfied by the scheme. NA: Attribute is not related to scheme.

4.6 Conclusion

In this paper, we proposed a mutual authentication protocol between a mobile client and a remote server. The objective was to design a secure and efficient protocol, suitable for mobile environments. Existing protocols in the literature have security flaws, such as impersonation, vulnerability to replay attacks, lack of user anonymity and untraceability, and synchronization problems due to the use of timestamps. The proposed protocol builds on ECC to meet security and efficiency requirements. Informal security analysis shows that our protocol is resistant to man in the middle attack, impersonation, replay attack, and guarantees user anonymity and untraceability.

In addition, we have formally demonstrated the security of the protocol with the BAN logic and the Scyther tool. Moreover, we have conducted a performance analysis of our protocol. The results show that it performs better than similar protocols in the literature. We can conclude that the proposed protocol is secure, efficient and suitable for mobile environments. As a future direction, it would be interesting to define a server key management policy and consider the use of user-side biometrics instead of passwords.

Acknowledgment

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

CHAPITRE 5 ARTICLE 2: AN ONLINE ENTROPY-BASED DDOS FLOODING ATTACK DETECTION SYSTEM WITH DYNAMIC THRESHOLD

Loïc D. Tsobdjou, Samuel Pierre, *Senior Member, IEEE*, and Alejandro Quintero, *Senior Member, IEEE*

Department of Computer and Software Engineering, École Polytechnique Montréal, Montreal, QC H3T 1J4, Canada

E-mail: loic.tsobdjou-dongmo@polymtl.ca; samuel.pierre@polymtl.ca; alejandro.quintero@polymtl.ca.

Revue : Accepté et publié dans le journal *IEEE Transactions on Network and Service Management*, volume 19, numéro 2, pages 1679-1689, juin 2022.

Abstract

Distributed denial of service attacks are cyberattacks that target the availability of servers. As a result, legitimate users no longer have access to the service. This can have a negative impact on an organization, such as lack of reputation and economic losses. Therefore, it is important to design defense mechanisms against these attacks. There are systems for detecting distributed denial of service attacks in the literature, which still have various shortcomings. Some of these systems detect the presence of attack traffic without identifying the attack packets or flows. Others use static thresholds and therefore cannot adapt to changes in legitimate traffic. In this paper, we propose an online system that aims to detect flooding attacks in a short timeframe and a client – server environment. The proposed detection system consists of five modules, namely features extraction and connections construction, suspicious activity detection, attack connections detection, alert generation and threshold update. The suspicious activity detection module calculates the normalized Shannon entropy by considering the source Internet Protocol address as a random variable. Suspicious activity is detected when the computed entropy is below a threshold. The threshold calculation is based on Chebyshev's theorem. We propose a dynamic threshold algorithm to track changes in legitimate traffic. We evaluate the proposed system through simulations and

using a publicly available dataset. Compared to other similar works, the proposed detection system has a better performance in terms of detection rate, false positive rate, precision and overall accuracy.

Keywords: DDoS detection, Information entropy, Network security.

5.1 Introduction

In a client – server environment, it is essential to ensure the availability of the server to provide a good quality of service (QoS) to customers. A distributed denial of service (DDoS) attack targets the availability of a server by flooding it with undesirable traffic from distributed devices [81]. Some of its effects are service downtime, reduction of QoS, economic losses, reputation and brand image losses, and others [82]. According to Kaspersky [108], we observe an 80% growth in DDoS attacks in Quarter I 2020 compared to Quarter I 2019.

There are several types of DDoS attacks. Salim *et al.* [83] differentiates between two classes of DDoS attacks, namely bandwidth depletion attacks and resource depletion attacks. Bandwidth depletion attacks aim to consume all the network's bandwidth including user datagram protocol (UDP) flood attack and internet control message protocol (ICMP) flood attack. However, resource depletion attacks aim to consume memory, central processing unit (CPU) and sockets; and they include ping of death and hypertext transfer protocol (HTTP) flood attacks. Vishwakarma and Jain [81] categorize DDoS attacks into application layer attacks and infrastructure layer attacks. HTTP flood and domain name system (DNS) amplification attacks are examples of application layer attacks. Infrastructure layer attacks exploit vulnerabilities in the transport and network layers of the open systems interconnection (OSI) model. Examples of such attacks are UDP flood and SYN flood. Agrawal and Tapaswi [84] organize DDoS attacks according to the rate at which malicious requests are sent. Thus, we have high-rate DDoS attacks and low-rate DDoS attacks.

The tools to conduct DDoS attacks are available online. Salim *et al.* [83] describe some of them, namely Golden eye, Low Orbit Ion Cannon (LOIC), Slowloris, Pyloris, DDoSim and Tor's Hammer.

There are two main approaches to detecting DDoS attacks: signature-based and anomaly-based [84]. In the signature-based approach, the characteristics of known attacks are stored in a database. The characteristics of the incoming traffic are compared with those of known attacks. The anomaly-

based approach aims to identify behaviors that are different from normal behavior. The limitation of the signature-based approach is that it cannot identify unknown attacks, while the anomaly-based approach has a high computational cost for training [85].

We can classify the anomaly-based approaches according to the algorithms used for attack detection. These are the approaches based on artificial intelligence algorithms and the statistical-based approaches [85]. Some disadvantages of approaches based on artificial intelligence algorithms are high consumption of computing resources, high detection delay due to long processing time and high false positive rate [85] [86]. A limitation of statistical-based approaches is the "difficulty of setting an optimal threshold" [85]. However, these approaches require few network packet header features to build the pattern of legitimate traffic [109]. In addition, they allow accurate and fast detection of DDoS attacks [85] [86]. In the proposed system, we intend to have low detection delay, high detection rate and low false positive rate. Therefore, we use information entropy which is a statistical-based approach.

Designing a defense mechanism against DDoS attacks is a research challenge as there are a multitude of types of DDoS attacks and tools to generate these attacks. Another research challenge is to propose an online system that detects DDoS attack in a short timeframe to minimize its impact on the QoS. We define an online system as a system that is integrated into a functional network and can analyze network traffic as it occurs. Unlike an online system, in an offline system, network traffic is first captured and exported to a file. This file is then used as input to the offline system. Our main objective is to design a real time DDoS flooding attack detection system in a client – server environment, to ensure the availability of the service to legitimate users.

The contributions of this paper are summarized as follows:

- We design an online entropy-based system that detects the presence of DDoS flooding attacks and identifies attack connections in a client – server environment.
- We propose a dynamic threshold algorithm that reflects the evolution of legitimate traffic.
- We evaluate the proposed system by simulation using graphical network simulator-3 (GNS3) [27].
- We also evaluate the proposed system with CICIDS2017 [28] dataset, which is a publicly available dataset.

The rest of this paper is organized as follows. In Section 5.2, we review the related works. In Section 5.3, we describe the proposed system. We present the results in Sections 5.4. Finally, we conclude in Section 5.5.

5.2 Related Work

In this section, we review recent works using information entropy to detect DDoS attacks.

Basicevic *et al.* [110] evaluate the performance of five entropy-based DDoS attack detectors. The entropies involved are Shannon, Renyi, Tsallis, Bhatia-Singh and Ubriaco. The detection process uses the source port distribution on the one hand and the flow size distribution on the other. The evaluation is performed on a dataset generated from the DETERlab (Cyber-Defense Technology Experimental Research Laboratory) emulation testbed. The authors conclude that the flow size distribution performs better than the source port distribution in terms of detection rate. The opposite is observed in terms of detection delay. Detectors based on generalized entropy perform better than those based on Shannon entropy.

Basicevic *et al.* [111] investigate the combination of principal component analysis with entropy-based techniques to detect DDoS attacks. The entropies studied are Shannon, Tsallis and Bhatia-Singh. It is found that, compared to entropy-based techniques, the combination of principal component analysis and generalized entropy (Tsallis and Bhatia-Singh) produces better performance in terms of true and false positive rate, F1 score, precision and recall.

Liu *et al.* [112] propose a method for detecting DDoS attacks. This method is based on fast Fourier transform and information entropy. The calculated fast Fourier transform coefficients and entropy are used as inputs to a neural network that detects DDoS attacks. The detection accuracy is more than 99.99%.

Gaurav *et al.* [113] present an approach to detecting DDoS attacks. This approach is implemented at the Fog layer of the Internet of Things. The Fog node calculates the entropy of incoming traffic using the source Internet Protocol (IP) address as a random variable. An attack is detected when the entropy is higher than a static threshold.

Ali *et al.* [114] describe a DDoS attack detection method that combines entropy and sequential probability ratio test. The entropy calculation is done using the destination IP address as a random variable. This entropy is used as input to the sequential probability ratio test. The result is compared

to a lower and an upper threshold to detect DDoS attacks. However, these thresholds are static and do not track variations in network traffic.

David and Thomas [115] propose an approach to detect DDoS attacks and flash events (FE). This approach is based on Tsallis entropy and a dynamic threshold algorithm. The network packet features used are the packet size and the destination IP address. The proposed approach detects the presence of an attack without identifying the attacking packets.

Gaurav *et al.* [116] suggest a DDoS attack detection system based on entropy and packet score. This system maintains a blacklist of IP addresses to discard malicious packets at an early stage. In addition, static thresholds for entropy and packet score are used to detect attacks. Zhou *et al.* [117] also use entropy and packet score to detect DDoS attacks and FE. Their approach consists of three modules, namely the score calculation module, the threshold calculation module and the entropy calculation module. The detection of attacks is done by comparing the packet score and entropy to thresholds. The accuracy obtained is about 93%.

Singh *et al.* [118] propose a method for identifying DDoS attacks and FE using the concepts of entropy and threshold. The detection metric used is the normalized Shannon entropy. The detection mechanism runs on the edge and gateway routers of autonomous systems. The detection method is divided into four modules: flow construction, suspicious activity detection, suspicious flow detection and DDoS attack confirmation. The authors simulate their method using OMNeT++ and INET. As limitations, this model requires synchronization between routers and introduces an additional communication cost between these routers.

Li *et al.* [119] present a scheme to accelerate the detection of volumetric DDoS attacks and design a real-time alarm system. There are three parts in the detection system, namely a traffic processor, an entropy calculator and a detection module. The detection metric used is the joint entropy. This scheme is evaluated with one author-generated dataset and three public datasets (1999 DARPA [120], 2009 DARPA [121] and UNB CIC DDoS 2019 [122]). However, this scheme uses a static threshold and cannot adapt to changes in legitimate traffic.

David and Thomas [88] describe an information-theoretic approach to detect DDoS attacks. The detection metrics used are Shannon entropy, Tsallis entropy and Renyi entropy. The authors use the stochastic gradient algorithm to calculate a dynamic threshold. The evaluation of the system is done with the MIT Lincoln Laboratory (LLDoS1.0) [123] and CAIDA [124] datasets. As a

drawback, the system detects the presence of an attack without identifying the attacking packets or flows.

Bojovic *et al.* [125] expose a method for detecting DDoS attacks that combines the advantages of feature-based and volume-based approaches. This detection method uses Shannon entropy and packet rate. The authors test their method in a real computer network environment, namely an academic computer network with controlled DDoS attacks. However, this method uses a static threshold and detects the presence of an attack without identifying the attacking packets or flows.

Zhou *et al.* [89] propose an approach to detect DDoS attacks using entropy rate measurement. This approach considers changes in entropy value as well as changes in the number of flows. The authors focus on low-rate DDoS and spoofing attack detection. As drawbacks, this work uses a static threshold and detects the presence of an attack without identifying the attacking packets or flows.

Daneshgadeh *et al.* [126] propose a DDoS attack and FE detection scheme that combines Shannon entropy with Kernel-based Online Anomaly Detection (KOAD) and Support Vector Machine (SVM) algorithms. The packet features used are the source IP address, destination IP address, and received bytes. In [127], the authors present another approach for detecting DDoS attacks and FE that combines entropy with KOAD and Mahalanobis distance. Entropy is computed using the source IP address as a random variable. In both approaches, to evaluate their model, the authors use DDoS traffic from simulations, FE traffic from the FIFA World Cup 98 dataset and normal traffic from a private company.

Behal *et al.* [109] present a generalized detection system for DDoS attacks and FE based on information theory metrics. The information theory metrics used are: Renyi entropy, Shannon entropy, Tsallis entropy, Kullback-Leibler divergence, generalized information distance, Hellinger distance and Jensen-Shannon distance.

Idhammad *et al.* [128] describe a system for detecting HTTP DDoS attacks in a cloud environment based on information entropy and Random Forest learning algorithm. This system consists of three steps: incoming network traffic entropy estimation, network traffic preprocessing and network traffic classification. The detection metric used is the Shannon entropy. Experiments are performed using the CIDDS-001 dataset [129]. The research works [109] and [128] share the following limitations: the use of a static threshold and the detection of the presence of an attack without identifying the attacking packets or flows.

Behal *et al.* [130] expose a distributed, flexible, automated and collaborative generalized entropy-based DDoS defense mechanism for early detection of DDoS attacks and FE. The detection metrics used are generalized entropy and information distance. Some shortcomings of this approach are an additional communication cost between routers, a need for synchronization between routers, the use of a static threshold and the detection of the presence of an attack without identifying the attacking packets or flows.

Kumar *et al.* [131] propose and implement SAFETY, a new solution that effectively detects and mitigates SYN flood attacks in software-defined networking (SDN). The detection metric used is the Shannon entropy. SAFETY has two major components: the detection unit and the mitigation unit. During the detection, the entropy is compared to a threshold. Experiments are performed using the Mininet simulator. SAFETY can detect SYN flood attacks only. Moreover, it introduces an additional communication cost between the switches and the controller.

Behal and Kumar [132] present a DDoS attacks and FE detection system using new information metrics. These metrics are generalized phi-entropy and generalized phi-divergence. Entropy and divergence are calculated with the IP source address as a random variable. As drawbacks, this detection system uses a static threshold and detects the presence of an attack without identifying the attacking packets or flows.

Bhuyan *et al.* [133] describe a victim-end defense mechanism that can detect DDoS flooding attacks. In addition, this mechanism needs to perform IP traceback to identify the attacking machines. The authors use extended entropy as detection metric and static thresholds. As a result, their detection mechanism cannot adapt to changes in legitimate traffic.

Sachdeva *et al.* [134] propose an entropy-based system that detects DDoS attacks against web services. Moreover, this system discriminates DDoS attacks from FE. The detection metric used is the traffic cluster entropy, where the traffic cluster is “the traffic generated from the same networks or administrative domains”. The authors evaluate their approach through simulations on network simulator 2 (NS-2).

Basicevic *et al.* [135] suggest the use of Tsallis entropy to detect SYN flood denial of service attacks. They compare the performance of Tsallis entropy-based detectors and Shannon entropy-based detectors. The simulations are done through NS-2. The authors conclude that Tsallis entropy-

based detector can perform better than Shannon entropy-based detector in terms of false positive rate. However, both detectors have approximately the same detection delay.

David and Thomas [136] present a DDoS attack detection system using an adaptive threshold algorithm and a fast entropy. First, a flow aggregation is performed to have a flow-based detection. A flow is defined by the tuple (source IP address, destination IP address, source port, destination port, protocol number). Then, the fast entropy is calculated to detect DDoS attacks. This system is not evaluated in terms of detection rate and false positive rate.

Rahmani *et al.* [137] describe a joint entropy-based DDoS attack detection system. This system detects DDoS attacks "by measuring the degree of coherence between the total number of packets received by the network and the number of connections". As drawbacks, this work uses a static threshold and detects the presence of an attack without identifying the attacking packets or flows.

Xiang *et al.* [138] expose a system that detects low-rate DDoS attacks using information metrics. Moreover, they propose an IP traceback scheme that identifies the local area network (LAN) of the attacks. The detection metrics used are the generalized entropy and the information distance. As limitations, this scheme requires synchronization between routers and introduces an additional communication cost between these routers. Furthermore, this system uses a static threshold.

Some recurring shortcomings are noticeable in the analyzed schemes.

- Many schemes detect the presence of an attack without identifying the attacking packets or flows [115] [88] [125] [89] [126] [109] [128] [130] [132] [134] [137]. Therefore, the network administrator cannot take further actions such as discarding attack flows. The system we propose in this paper can detect the presence of an attack and identify the attack connections.
- Some schemes use static thresholds to detect DDoS attacks [113] [114] [116] [119] [125] [89] [109] [128] [130] [132] [133] [134] [135] [137] [138]. These detection systems cannot adapt to the changes in legitimate traffic. In this paper, we propose a dynamic threshold algorithm that enables the detection system to be aware of changes in legitimate traffic.
- Some researchers validate their detection systems using legitimate and attack traffic from two different datasets [88] [89] [126] [127] [109] [130] [132] [133] [138]. According to Çakmakçı *et al.* [26], merging two datasets with different network characteristics and

topologies can influence the results. We address this limitation using the CICIDS2017 [28] dataset that contains both legitimate and attack traffics.

5.3 Proposed Detection System

5.3.1 Information entropy

Shannon [25] introduced the notion of entropy as a quantity that measures information. Entropy is also "a measure of the uncertainty of a random variable" [139].

Let X be a discrete random variable with alphabet A and probability mass function $p(x), x \in A$. The Shannon entropy $H(X)$ of a discrete random variable X is defined by (1) [139] [140].

$$H(X) = -\sum_{x \in A} p(x).\log_2 p(x) \qquad (1)$$

The entropy $H(X)$ varies in the interval $[0, \log_2|A|]$, where $|A|$ is the number of elements of the alphabet A. To have a quantity that varies in an interval independent of $|A|$, we define the normalized entropy E as follows:

$$E = -\frac{1}{\log_2|A|}\sum_{x \in A} p(x).\log_2 p(x) \quad (2)$$

The normalized entropy so defined varies in the interval $[0, 1]$.

5.3.2 Network architecture

Figure 5.1 illustrates a network architecture in a client – server environment. The network is divided into two parts: the external network and the internal network. The external network contains both legitimate and malicious clients. The internal network contains the DDoS attack detection system and the servers we need to protect. The detection system passively analyzes the traffic passing through the switch to detect the presence of an attack.

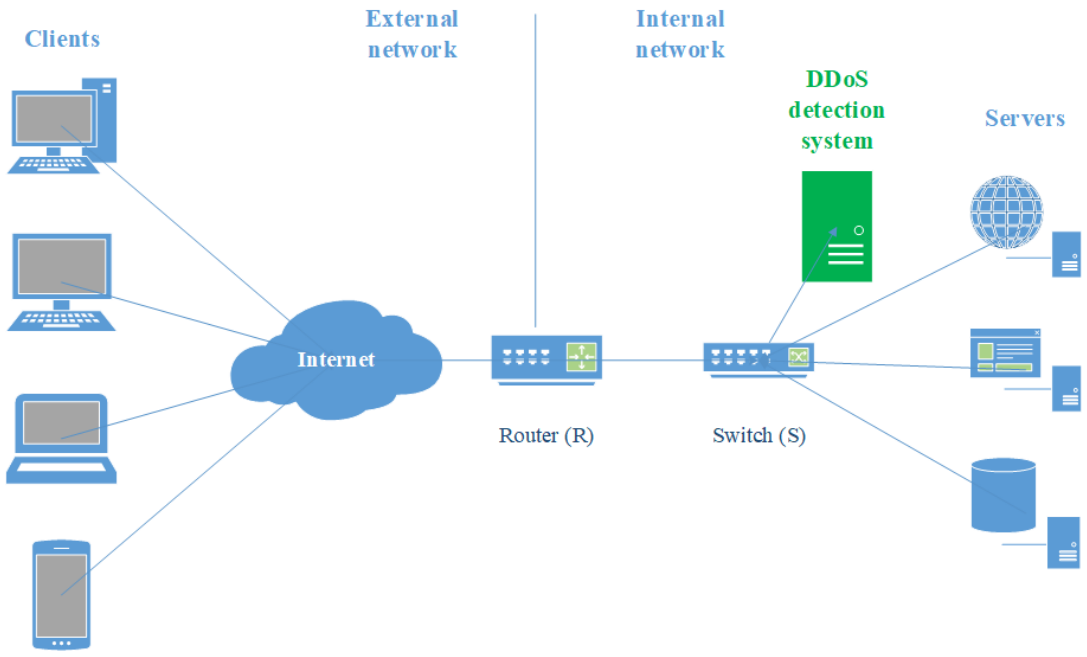

Figure 5.1 Network architecture

5.3.3 Description of the proposed system

In designing the detection system, we make the following two assumptions.

- A1: All packets with a spoofed source IP address are discarded by the router. Thus, these packets are not analyzed by the detection system. This assumption is reasonable since there are works in the literature to detect IP address spoofing [141] [142] [143].
- A2: When it starts, the detection system has enough time to build the profile of legitimate traffic before the attack traffic starts.

The notations used in the proposed system are listed in Table 5.1.

The proposed detection system consists of five modules inspired by [118]: features extraction and connections construction, suspicious activity detection, attack connections detection, alert generation and threshold update. Figure 5.2 shows the overall architecture of the proposed detection system.

Table 5.1 Notations

Symbol	Description
$\Delta\boldsymbol{T}$	Time window, network traffic sampling period
$\boldsymbol{src}, \boldsymbol{dst}$	Source IP address and destination IP address respectively
$\boldsymbol{AC}$	List of attack connections
$\boldsymbol{C}$	List of connections
$\boldsymbol{B}, \boldsymbol{F}$	Number of packets in a connection in the backward and forward directions, respectively
$\boldsymbol{\delta}$	Threshold for normalized entropy
$\boldsymbol{S}$	Threshold for the number of packets in a connection
$\boldsymbol{E}$	Normalized entropy
$\boldsymbol{EV}$	Vector containing the values of the normalized entropy of legitimate traffic
$\boldsymbol{N}$	Size of the EV vector
$\boldsymbol{k}, \boldsymbol{j}$	Two non-zero natural numbers
$\boldsymbol{DR}$	Detection rate
$\boldsymbol{FPR}$	False positive rate
$\boldsymbol{Pr}$	System precision
$\boldsymbol{Ac}$	Overall accuracy
$\boldsymbol{DD}$	Detection delay

5.3.3.1 Module 1: features extraction and connections construction

This module takes network traffic and time window ΔT as inputs and produces bidirectional connections as output. A connection is defined by the couple (source IP address, destination IP address). These features are extracted from the headers of each network packet. Three attributes are associated with each connection: the number of packets in the forward direction (F), the number

of packets in the backward direction (B) and the timestamp of the first packet of the connection. The direction is defined by the first packet of the connection. The network traffic is analyzed and the connections are constructed every time window ΔT.

Algorithm 1 describes the different steps of this module.

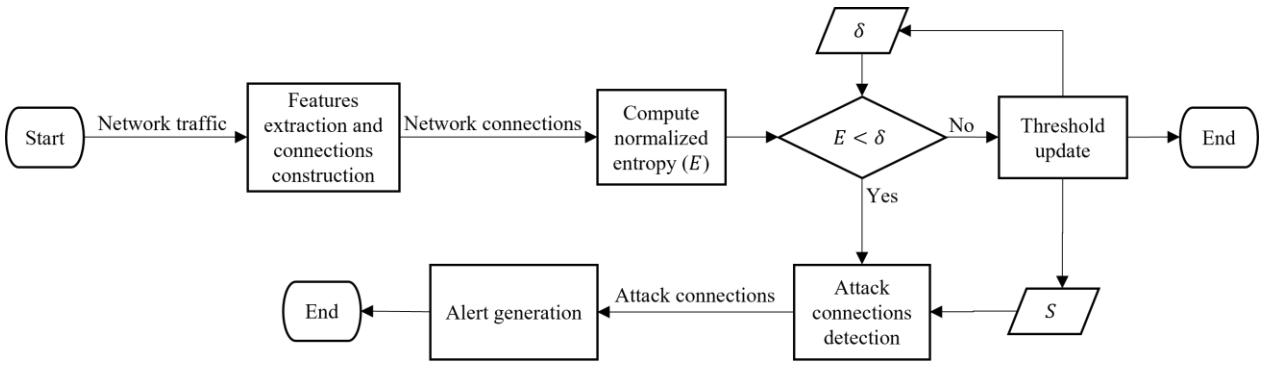

Figure 5.2 DDoS detection system architecture

Algorithm 1. feature extraction and connections construction

Input: Network traffic, time window (ΔT)

Output: List of connections (C)

while True

  $C = [\,]$

  $st = start\ time$

  while $(current\ time - st) < \Delta T$

    read one network packet

    extract packet features $(src, dst, timestamp)$

    if connection (src, dst) in C, **then** $F = F + 1$

    else if connection (dst, src) in C, **then** $B = B + 1$

    else add connection (src, dst) in C; $F = 1; B = 0$

  end

  return C

End

5.3.3.2 Module 2: suspicious activity detection

This module takes as inputs the different connections constructed in module 1 and the threshold value (δ) and produces as output a Boolean value. This value is set to "true" when suspicious activity is detected and "false" otherwise. A suspicious activity is detected when the normalized entropy is below the threshold δ. The normalized entropy is calculated by considering the source IP address as a random variable. In this case, A in (2) is the set of the source IP addresses extracted from network traffic at module 1. During the time window ΔT, the probability (p) of a source IP address taking the value src_i is defined by (3).

$$p = \frac{Number\ of\ packets\ with\ source\ IP\ address\ src_i}{Total\ number\ of\ packets} \qquad (3)$$

Algorithm 2 presents the operation of this module.

Algorithm 2. suspicious activity detection

Input: List of connections (C), threshold (δ)
Output: True or False
Compute the probability mass function (p) using (3)
Compute the normalized entropy (E) using (2)
If $E < \delta$, **then return** True
Else return False

5.3.3.3 Module 3: attack connections detection

This module is triggered when module 2 has detected a suspicious activity. It takes as inputs the different connections built in module 1 and the threshold value (S) and produces as output the attack connections.

A connection is identified as an attack connection when the number of packets of this connection during the time window ΔT is higher than the threshold S.

The operations of this module are described in algorithm 3.

Algorithm 3. attack connection detection

Input: List of connections (C), threshold (S)
Output: List of attack connections (AC)
For each connection in C,
    compute the number of packets ($P_i = F + B$)
    If ($P_i > S$) **then**
        add the corresponding connection in AC
    End
End
Return AC

5.3.3.4 Module 4: alert generation

This module intervenes when an attack connection is detected. It takes as input the attack connections identified in module 3 and produces as output alerts. An alert consists of the attack connection identifier, the connection timestamp and the detection time. The network administrator can then use these alerts to take further action, such as blocking packets from these connections. Algorithm 4 summarizes this module.

Algorithm 4. alert generation

Input: List of attack connections (AC)
Output: Alerts
For each attack connection in AC,
    return Alert (connection ID, connection timestamp, detection time)
End

5.3.3.5 Module 5: threshold update

If module 2 does not detect any suspicious activity, then we call the threshold update module. This module takes as inputs the current value of the normalized entropy of legitimate traffic, the current value of the average number of packets of a legitimate connection, a vector containing the last N

values of the normalized entropy of legitimate traffic and a vector containing the last N values of the average number of packets of a legitimate connection. It produces as outputs the updated thresholds δ and S. Algorithm 5 presents the procedure for updating the threshold.

Algorithm 5. threshold update

Input: Current value of the normalized entropy of legitimate traffic (EC)

Current value of the average number of packets of a legitimate connection (PC)

Vector containing the last N values of the normalized entropy of legitimate traffic (EV)

Vector containing the last N values of the average number of packets of a legitimate connection (PV)

Output: Thresholds (δ, S)

Delete the oldest element of the EV vector

Delete the oldest element of the PV vector

Add EC at the end of the EV vector

Add PC at the end of the PV vector

Calculate the mean M and the standard deviation σ of the new vector EV

Calculate the mean R and the standard deviation d of the new vector PV

Compute $\delta = M - k.\sigma$ and $S = R + j.d$

Return δ and S

Threshold initialization

The proposed system uses a threshold δ at module 2 to detect suspicious activity and another threshold S at module 3 to detect attack connections. To initialize these thresholds, we sample the network traffic every time window ΔT and consider the first N samples. According to assumption A2, these N samples contain only legitimate traffic. For each sample, using (2), we compute the normalized entropy $E_i, i = 1, \dots, N$. We also compute the average number of packets of a connection $P_i, i = 1, \dots, N$. We then construct the vectors $EV = [E_1, \dots, E_N]$ and $PV = [P_1, \dots, P_N]$. The initial thresholds δ and S are computed using (4) and (5) respectively.

$$\delta = M - k.\sigma, \qquad (4)$$

where k is a non-zero natural number, and M and σ are the mean and the standard deviation of the vector EV respectively.

$$S = R + j.d, \qquad (5)$$

where j is a non-zero natural number, and R and d are the mean and the standard deviation of the vector PV respectively.

The theoretical basis for the calculation of thresholds is Chebyshev's theorem. This theorem is stated as follows: for any positive integer k, the probability that a random variable takes a value in the interval $[\mu - k.\sigma, \mu + k.\sigma]$ is at least $1 - \frac{1}{k^2}$ [87]. Where μ and σ are the mean and the standard deviation of the random variable respectively.

During a DDoS flooding attack, network traffic contains many more source IP addresses from malicious clients. This leads to a decrease in the degree of uncertainty about the source IP address, and thus a drop in the normalized entropy. For this reason, we considered the lower bound of the interval given in the Chebyshev's theorem ($\mu - k.\sigma$) as the threshold for the normalized entropy. However, during the attack, the number of packets in an attack connection is greater than the average number of packets in a legitimate connection. Therefore, we considered the upper bound of the interval given in the Chebyshev's theorem ($\mu + k.\sigma$) to identify attack connections.

5.4 Results

5.4.1 Evaluation metrics

We define true positive, true negative, false positive and false negative as follows:

- True positive (TP): number of attack connections identified as attack connections.
- True negative (TN): number of legitimate connections identified as legitimate connections.
- False positive (FP): number of legitimate connections identified as attack connections.
- False negative (FN): number of attack connections identified as legitimate connections.

We use five metrics to evaluate the proposed detection system. These are the detection rate (DR), the false positive rate (FPR), the system precision (Pr), the overall accuracy (Ac) and the detection delay (DD). The detection rate, the false positive rate, the system precision and the overall accuracy

are defined in (6), (7), (8) and (9) respectively. The detection delay is the difference between the attack detection time and the reception time of the first attack connection.

$$DR = \frac{TP}{TP+FN} \quad (6)$$

$$FPR = \frac{FP}{FP+TN} \quad (7)$$

$$Pr = \frac{TP}{TP+FP} \quad (8)$$

$$Ac = \frac{TP+TN}{TP+FP+TN+FN} \quad (9)$$

The DR validates the effectiveness of the detection system while the FPR measures the ability of the system to separate legitimate traffic from attack traffic [92]. The system precision evaluates the quality of the system while the overall accuracy allows comparing several detection systems [92].

5.4.2 Simulation environment

Figure 5.3 represents the network architecture used for the simulation. It consists of four LANs. LANs 1, 2 and 3 each contains three clients, including two legitimate clients and one malicious client. LAN 4 contains two web servers and the DDoS attack detection system. This detection system analyzes all traffics that pass through the LAN 4 switch.

We configure the network architecture of Figure 5.3 on the GNS3 [27] simulator. This simulator is installed on a personal computer with an Intel Core i5-7200U 2.50 GHz processor, 8 GB RAM and Windows 10 Family 64-bit operating system. We develop the detection system using the Python programming language and deploy it on a VMware virtual machine with the Ubuntu 20 operating system.

We generate legitimate traffic with the httperf tool [90]. This tool generates HTTP GET requests and the time interval between two successive requests follows an exponential distribution with a mean interarrival time of one second. The attack traffic is generated with the hping3 tool [91]. We simulate ICMP flood and SYN flood attacks. The web server used is Apache [144].

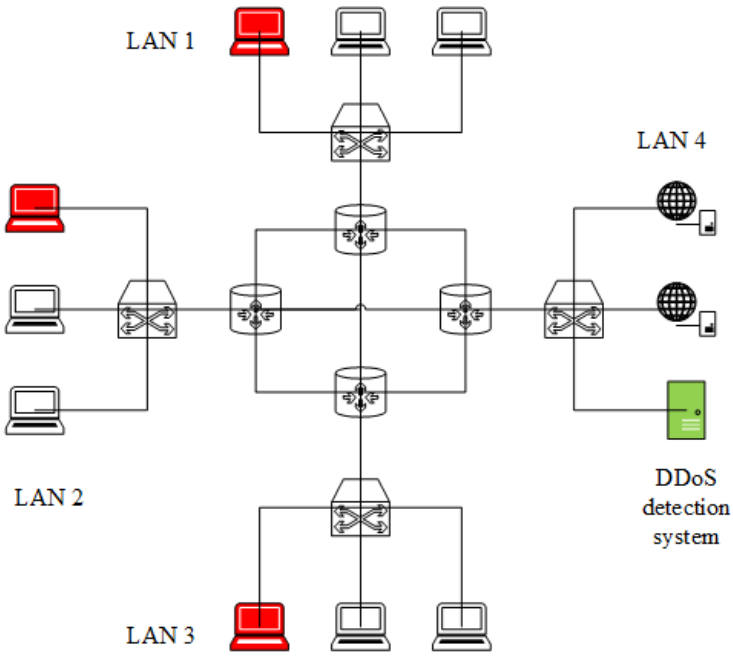

Figure 5.3 Test bed

5.4.3 Simulation results

Table 5.2 presents the simulation parameters. Each simulation lasts six minutes. During these six minutes, legitimate traffic flows through the network. The attack traffic starts at the fourth minute and ends at the fifth minute. The value of N is 20, ΔT varies from 1 to 4 seconds, k varies from 1 to 16 and j varies from 1 to 13.

Each experiment is repeated five times. The value retained for each evaluation metric is the arithmetic mean of the values obtained in the five experiments.

First, we simulate the ICMP flood attack with the values $N = 20, \Delta T = 4\ seconds, j = 5$ and k varying from 1 to 16. Figure 5.4 shows the results obtained. It is the variation of the detection rate and the false positive rate with respect to k. As k increases, the value of the threshold δ decreases. For values of k greater than 12, the detection system is no longer able to identify all attack connections. This explains the drop in the detection rate. We obtain better performance ($DR = 100\%\ and\ FPR = 0\%$) for $k = 12$.

Table 5.2 Simulation parameters

Parameter	**Description**
Attack	ICMP flood and SYN flood
Legitimate traffic	Web traffic
Simulation duration	6 minutes
Duration of legitimate traffic	6 minutes
Duration of attack traffic	1 minute (from the 4th to the 5th minute)
$\boldsymbol{N}$	20
$\boldsymbol{\Delta T}$	1, 2, 3, 4 seconds
$\boldsymbol{k}$	1, 2, 4, 6, …, 14, 16
$\boldsymbol{j}$	1, 3, 5, 7, 9, 11, 13

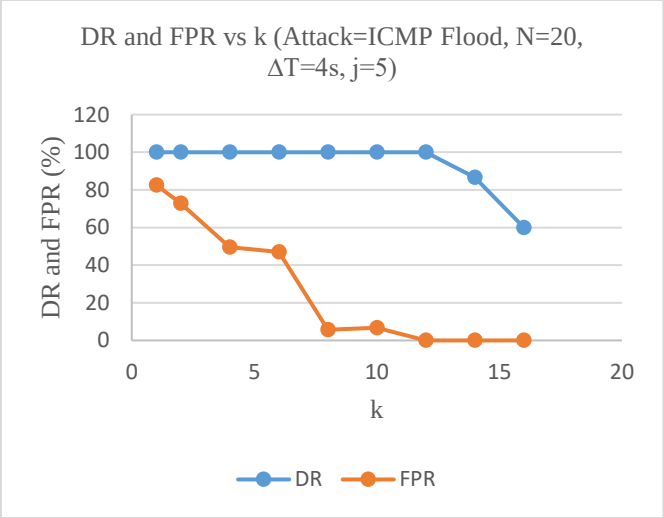

Figure 5.4 Detection rate and false positive rate with respect to k

Then, we set $N = 20, \Delta T = 4\ seconds, k = 12$ and j varying from 1 to 13. Figure 5.5 shows the variation of the detection rate and the false positive rate with respect to j. As j increases, the value of the threshold S also increases. This reduces the ability of the proposed system to detect attacks.

For values of j greater than 9, the system is no longer able to detect all attack connections. We obtain better performance ($DR = 100\%\ and\ FPR = 0\%$) for values of j between 5 and 9.

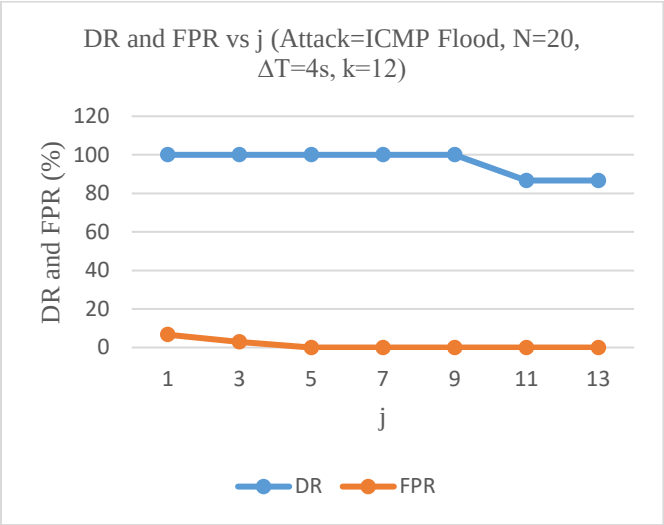

Figure 5.5 Detection rate and false positive rate with respect to j

Next, we set $N = 20, k = 12, j = 5$ and vary ΔT from 1 to 4 seconds. The variation of the detection rate and the false positive rate with respect to ΔT is plotted in Figure 5.6. The value $\Delta T = 4\ seconds$ gives better performance ($DR = 100\%\ and\ FPR = 0\%$). This can be interpreted by the fact that for time windows ΔT less than 4 seconds, the detection system does not have enough network traffic samples to build the legitimate traffic profile.

Finally, we set $N = 20, k = 12, j = 5, \Delta T = 4\ seconds$ and we simulate the SYN flood attack. The overall results are reported in Table 5.3.

The proposed detection system achieves 100% detection rate, 0% false positive rate, 100% precision, 100% overall accuracy and 7.89 seconds detection delay.

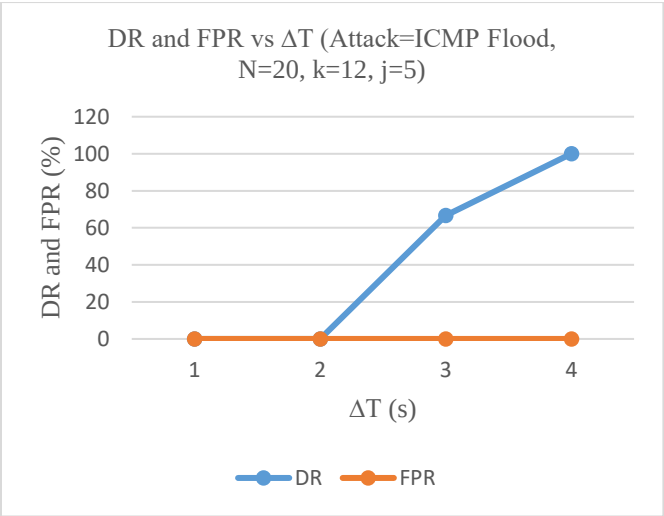

Figure 5.6 Detection rate and false positive rate with respect to ΔT

Table 5.3 Simulation results with $N = 20, \Delta T = 4s, k = 12, j = 5$

Attack	**DR (%)**	**FPR (%)**	**Pr (%)**	**Ac (%)**	**DD (s)**
ICMP flood	100	0	100	100	7.89
SYN flood	100	0	100	100	5.31

5.4.4 Evaluation with CICIDS2017 dataset

We evaluate the proposed detection system using the CICIDS2017 dataset [28]. This dataset is from the Canadian institute for cybersecurity. Traffic is captured from Monday, July 3rd, 2017, through Friday, July 7th, 2017.

To evaluate our work, we consider traffic on Friday, July 7th, 2017, from 3:30 to 5:02 pm. This traffic contains both legitimate and attack traffics. DDoS attacks are generated with the Low Orbit Ion Canon (LOIC) tool.

The network environment used to generate the CICIDS2017 dataset is different from the one we used for our simulations. Therefore, we need to refine again the ΔT, k and j parameters to get good performances.

In this dataset, the timestamp granularity is the minute. Thus, the smallest value we can assign to the time window ΔT is 1 minute, or 60 seconds.

First, we set $N = 20, \Delta T = 60\ seconds, j = 5$ and k varying from 1 to 8. Figure 5.7 shows the variation of the detection rate and the false positive rate with respect to k. We get the best performance ($DR = 100\%\ and\ FPR = 1.37\%$) when $k = 4$.

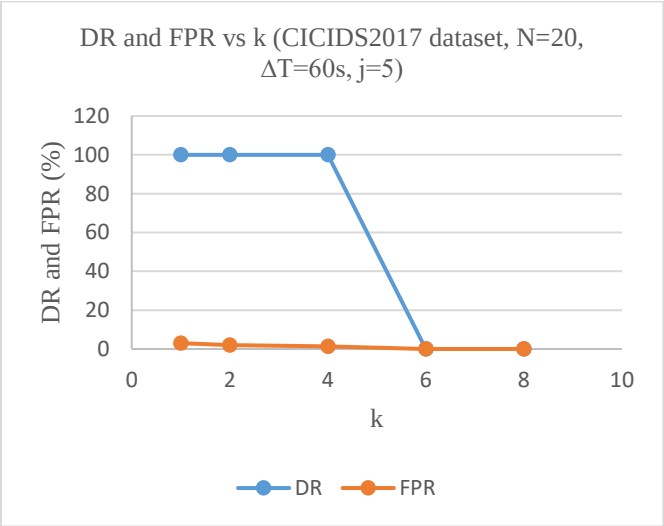

Figure 5.7 Detection rate and false positive rate with respect to k (CICIDS2017 dataset)

Secondly, we set $N = 20, \Delta T = 60\ seconds, k = 4$ and j varying from 1 to 101. The variation of the detection rate and the false positive rate with respect to j is plotted in Figure 5.8. The value $j = 101$ gives better performance ($DR = 100\%\ and\ FPR = 0\%$).

Table 5.4 shows the results of the evaluation with the CICIDS2017 dataset. The detection rate is 100%, the false positive rate is 0%, the system precision is 100% and the overall accuracy is 100%.

5.4.5 Comparison with similar works

In this section, we compare the performance of the proposed detection system with some similar works from the literature [118] [119] [88] [125] [130]. Table 5.5 presents the results.

The work [119] has a better detection delay than ours. However, we outperform it in terms of detection rate and false positive rate.

Compared to other similar works, we can conclude that the proposed detection system has a better performance in terms of detection rate, false positive rate, system precision and overall accuracy.

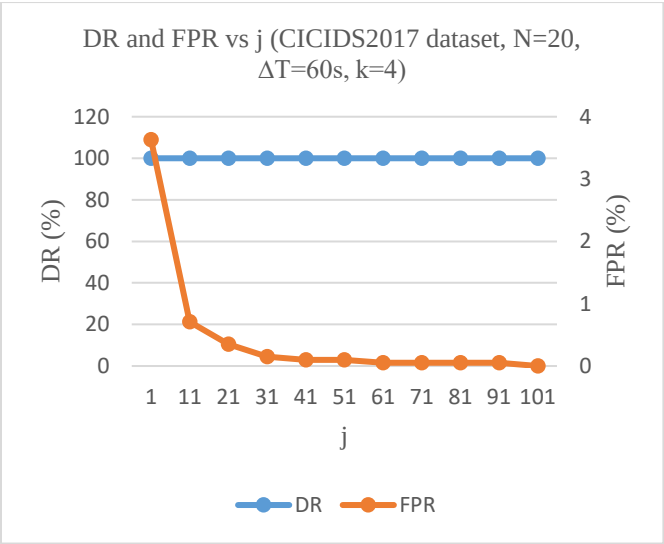

Figure 5.8 Detection rate and false positive rate with respect to *j* (CICIDS2017 dataset)

Table 5.4 Evaluation results with CICIDS2017 dataset

DR (%)	FPR (%)	Pr (%)	Ac (%)
100	0	100	100

Table 5.5 Comparison with similar works

Detection system	DR (%)	FPR (%)	Pr (%)	Ac (%)	DD (s)
[118]	96	4	NA	NA	NA
[119]	90	3	NA	NA	0.01
[88]	97	0	NA	99.6	NA
[125]	100	0	90	NA	60
[130]	93	10	97	NA	NA
Ours	100	0	100	100	7.89

NA: evaluation metric is not available

5.5 Conclusion

In this paper, we proposed a system to detect DDoS flooding attack in a client – server environment. The objective was to design a system that detects DDoS flooding attacks against a server in a short timeframe. The purpose of such a system is to ensure the availability of service to legitimate users. The proposed detection system relies on information entropy to identify the presence of attack traffic in the network. It consists of five modules namely feature extraction and connections construction, suspicious activity detection, attack connections detection, alert generation and threshold update. We evaluated our work through simulation on GNS3. We simulated ICMP flood and SYN flood attacks. We obtained a detection rate of 100%, a false positive rate of 0%, a precision of 100%, an overall accuracy of 100% and a detection delay of 7.89 seconds. Moreover, we evaluated the proposed system using the CICIDS2017 dataset. We achieved a detection rate of 100%, a false positive rate of 0%, a precision of 100% and an overall accuracy of 100%. These two evaluations show that the proposed system can perform well in different environments. The ΔT, k and j parameters need to be fine-tuned to get the best performance. As extension of this work, it would be interesting to consider the detection of low-rate DDoS attacks.

Acknowledgment

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

CHAPITRE 6 ARTICLE 3: A FRAMEWORK FOR SECURITY ASSESSMENT OF ANDROID MOBILE BANKING APPLICATIONS

Loïc D. Tsobdjou, Samuel Pierre, *Senior Member, IEEE*, and Alejandro Quintero, *Senior Member, IEEE*

Department of Computer and Software Engineering, École Polytechnique Montréal, Montreal, QC H3T 1J4, Canada

E-mail: loic.tsobdjou-dongmo@polymtl.ca; samuel.pierre@polymtl.ca; alejandro.quintero@polymtl.ca.

Revue : Soumis pour publication dans le journal *IEEE Transactions on Dependable and Secure Computing*, en mai 2022.

Abstract

Mobile banking applications make users' daily lives easier by allowing them to access banking services, such as balance inquiries and bill payments, anytime and anywhere. Since these applications manage very sensitive financial data, special attention must be paid to data security. Several works in the literature assess the security of mobile banking applications. However, we note the lack of a widely adopted framework among researchers, for assessing the security of mobile banking applications. In this paper, we propose a framework consisting of twenty-six criteria for assessing the security of Android mobile banking applications. These criteria are divided into five categories: *mobile device security*, *data in transit*, *data storage*, *cryptographic misuse* and *others*. Subsequently, we evaluate the proposed framework based on predefined requirements. These requirements are *no redundancy*, *no ambiguity* and *comprehensiveness*. As a case study, we assess the security of the Android mobile banking applications of seven major Canadian banks. The results show that data in transit is adequately protected by these applications.

Keywords: Android, Mobile banking, Mobile code security, Security assessment.

6.1 Introduction

Mobile banking is the act of making financial transactions, such as balance inquiries, bill payments and money transfers on mobile devices (e.g., smartphones, tablets, etc.) [145]. One of the benefits of mobile banking is access to banking services anytime, anywhere. However, its main disadvantage is the data security issue, as financial data is very sensitive. According to the Canadian Bankers Association, 86% of Canadians trust their bank to offer secure digital banking services [93]. The main factors that influence the use of mobile banking are the services offered and the security related to it [146]. Arisya *et al.* [147] reveal that mobile banking applications users are aware of information security. Boutin and Chouinard [148] point out that cyber-attacks in the Canadian financial sector have tripled between the years 2014 to 2019. For example, in May 2018, the leak of confidential personal data affected 40,000 and 50,000 customers of Canadian banks CIBC and BMO, respectively [148].

Services providers must then ensure data confidentiality, integrity, availability and privacy. Data confidentiality refers to the fact that only authorized persons can access the data. Integrity ensures that the data is not altered and is in a usable format. Availability ensures that legitimate users can always access data. Privacy refers to protecting the users' identities.

Mobile devices face several security threats including theft or loss, denial of service, malware, application vulnerabilities and operating system vulnerabilities. A malicious user in possession of a stolen or lost mobile device, for example, can access the data stored on that device using forensic analysis techniques. This jeopardizes data confidentiality if data is not well protected. Denial of service compromises the data availability by preventing users from accessing their data and services. Malware acts without the user's knowledge to steal or alter data, thus affecting the data confidentiality, integrity and privacy. Application and operating system vulnerabilities are security holes in software. Moreover, cybercriminals can exploit these vulnerabilities to compromise data security.

In this paper, we focus on vulnerabilities in Android mobile banking applications. We choose the Android operating system for two reasons. The first reason is that Android is the leader in mobile operating systems worldwide, with about 70% of the market share in January 2022 [149]. The second reason is that the open nature of Android exposes it to more security threats.

Our main objective is to propose a framework for assessing the security of mobile banking applications, in order to enable banks to integrate security considerations into applications' design and development.

The contributions of this paper are summarized as follows:

- We propose a set of concrete and comprehensive criteria for assessing the security of Android mobile banking applications.
- We define the requirements that the proposed set of criteria must satisfy, and we evaluate the proposed set of criteria based on these requirements.
- We assess the security of the Android mobile banking applications of seven major Canadian banks.

The rest of this paper is organized as follows. In Section 6.2, we review the related works. We introduce the proposed framework and the requirements that the proposed framework must satisfy in Section 6.3. In Section 6.4, we conduct a case study to assess the security of seven Android mobile banking applications from major banks in the Canadian market. Finally, we conclude our work in Section 6.5.

6.2 Literature Review

This section consists of three sub-sections. The first sub-section presents the *Open Web Application Security Project (OWASP) Mobile Top 10* project. The second one reviews the Android application security best practices defined by Google. The third sub-section covers research articles about the security assessment of mobile banking applications.

6.2.1 OWASP Mobile Top 10

The Open Web Application Security Project (OWASP) is a foundation launched in 2001, where its objective is to improve software security [150]. In 2016, OWASP published a list of the ten most common security risks related to a mobile application, under the project *OWASP Mobile Top 10* as follows [95]:

- M1: Improper platform usage;
- M2: Insecure Data Storage;

- M3: Insecure Communication;
- M4: Insecure Authentication;
- M5: Insufficient Cryptography;
- M6: Insecure Authorization;
- M7: Client Code Quality;
- M8: Code Tampering;
- M9: Reverse Engineering;
- M10: Extraneous Functionality.

The M1 risk refers to the misuse of mobile device operating system features. For example, using unnecessary Android permission can lead to the leakage of sensitive user data. In addition, the use of app local storage instead of a keychain to store sensitive data in iOS applications.

The M2 risk occurs when sensitive data is stored without proper protection. As an illustration, the user password should not be stored in plaintext. We should protect the user password using a password-based key derivation function.

The Risk M3 reflects the insecure transfer of sensitive data via a communication channel. It is important to mention that the communication channel between the mobile device and the server is not secure. A proper implementation of standard security protocols like Transport Layer Security (TLS) can be a solution to this risk.

In the M4 risk, malicious users can bypass the authentication process. As an example, a short password is prone to brute force attacks.

The M5 risk refers to the misuse of cryptographic techniques. Vulnerabilities in this category include the use of encryption algorithms known to be insecure, hardcoding of encryption keys in the source code of the mobile application, etc.

The M6 risk occurs when user accesses features reserved for a higher privilege user. Unauthorized access to a database can lead to data leakage and identity theft.

The risk M7 “captures bad code that executes on the mobile device itself” [95]. Examples include buffer overflow and untrusted inputs.

In the M8 risk, the source code of the mobile application is modified and redistributed to users. The repackaging attack is an example of this category. To prevent this risk, the mobile application can check its integrity at runtime.

The M9 risk refers to the generation and analysis of the source code of an application. It can be prevented by using obfuscation.

Finally, the risk M10 consists of "leaving functionality enabled in the app that was not intended to be released" [95].

6.2.2 Google App Security Best Practices

In this section, we review some Android application security best practices. Google specifies these best practices [151] as the developer of the Android operating system. We distinguish four categories:

- enforce secure communication;
- provide the right permissions;
- store data safely;
- keep services and dependencies up-to-date.

The first category *enforce secure communication* concerns the security of communication between applications on the same device, or between the application and a remote device like a server. Best practices in this category include:

- use implicit intents and non-exported content providers;
- ask for credentials before showing sensitive information;
- apply network security measures;
- use WebView objects carefully.

The second category *provide the right permissions* refers to the proper use of permissions. On Android, permission is a concept used to restrict the application's access to some data and resources of the mobile device. Google recommends using only those permissions that are necessary for the application to function properly. Best practices in this category are: 1) use intents to defer permissions; and 2) share data securely across apps.

The third category *store data safely* is to protect sensitive data at rest. Google recommends storing confidential data within internal storage, as other applications cannot access it, and the device delete these data when the application is uninstalled. Moreover, sensitive data can be encrypted before being saved. Other best practices in this category include: 1) store data in external storage based on use case; 2) store only non-sensitive data in cache files; and 3) use *SharedPreferences* in private mode.

Finally, the fourth category *keep services and dependencies up-to-date* consists of updating all dependencies such as libraries, before the application is released.

6.2.3 Related Work

In this section, we review recent works assessing the security of mobile banking applications.

Chen *et al.* [18] [152] assess the data-related weaknesses in Android banking applications. The study is made on 693 banking apps across eighty-three countries. The authors propose an automated security risk assessment system (AUSERA) that identifies security weaknesses in banking apps. They claim that their system outperforms four state-of-the-art industrial and open-source tools in data-related weaknesses detection, namely AndroBugs, Mobile Security Framework (MobSF), Quick Android Review Kit (QARK) and Qihoo 360. Security weaknesses are classified into four categories: input harvest, data storage, data transmission and communication infrastructure. The authors found out that the most popular weaknesses of banking apps are screenshot, insecure hash function and inter component communication (ICC) Leakage.

Bojjagani and Sastry [153] propose a threat model that aims to detect and mitigate vulnerabilities in Android and iOS mobile applications. Moreover, they build a security testing framework to identify man-in-the-middle attacks in mobile applications. Finally, the compliance of mobile banking applications to the OWASP listed mobile security risks is analyzed. Mobile security threats are divided into three categories: application-level threats, communication level threats and device level threats. Application-level threats are insecure data storage, lack of binary protection, unintended data leakage, malware in apps, weak cryptography and privilege escalation. The communication level threat is an insufficient transport layer protection. Devices level threats include mobile forensics and malware in mobile. For the security testing framework, the authors adopt both static and dynamic analysis. Static analysis examines the program structure while

dynamic analysis examines the program in operation. As case studies, the authors analyze five Android and three iOS mobile banking applications in India.

Zheng *et al.* [154] analyze security vulnerabilities in Android mobile applications. They focus on repackaging attacks. In these attacks, an attacker disassembles a legitimate application, injects malicious code, and re-builds it into a new application. The authors use three mobile banking applications in Australia as case studies. They implement repackaging attacks to steal sensitive information, such as Short Message Service (SMS) and contact list. Some techniques to mitigate repackaging attacks are obfuscation, runtime detection and improving user awareness.

Chanajitt *et al.* [155] analyze seven Android mobile banking apps in Thailand. Their aim is to assess the security of these apps. More specifically, they analyze the security of sensitive data after user registration, banking transaction and transition of data over the network. Moreover, they evaluate the resistance of the app against repackaging attack. The devices used for test are two Android devices: Samsung Galaxy S4 GT-I9500 (rooted) and Galaxy S4 mini-GT-I9190 (unrooted). First, some activities are performed on the mobile banking apps. Then, the authors acquire an image of the flash memory for further forensic analysis. In addition, the application code is statically analyzed to evaluate the security features.

Kaur *et al.* [156] assess the security of the Android Pay application, along with the Android e-wallet apps of three Canadian banks. They focus on Host Card Emulation – Near-Field Communication (HCE-NFC) enabled apps. The authors propose a set of security rules obtained from the Canadian Bankers Association and OWASP Top 10 Mobile Risks. The set of rules consists of device security (e.g., no rooted device and no emulator), application security (e.g., self-verification integrity, protected app, etc.), communication security and dynamic data (e.g., https enforced, proper certificate pinned, etc.), device storage (e.g., no plain text logs and no sensitive information in backups) and memory (e.g., no sensitive information in memory). The authors conduct both static and dynamic analysis. The results show that e-wallet banking Android apps in the Canadian market are not well secured against some attack vectors, including rooted devices, emulator and self-verification integrity.

Bojjagani and Sastry [157] analyze Android mobile banking apps at the application level, network level and device level. They identify some threat scenarios, namely untrusted party learning of bank data, tampering with bank data and a customer choosing the wrong bank app. The mobile

vulnerabilities are masquerade, man-in-the-middle, replay, traffic analysis / Wi-Fi sniffing, browser exploits, SQL injection, lack of binary protection, broken cryptography, data leakage and insecure data storage. The testing strategy consists of four parts: static analysis, dynamic analysis, Web app server security and device forensic.

Jung *et al.* [158] study repackaging attacks against Android banking apps in the South Korean market. Seven Android banking apps are analyzed. They implement a repackaging attack in which money transfer is sent not to the intended recipient, but to the attacker. The attack succeeds for all seven banking apps. As countermeasures to repackaging attacks, the authors suggest self-signing restriction, code obfuscation and code attestation.

Reaves *et al.* [159] investigate the security of branchless banking applications (also known as mobile money applications) in developing countries. First, the authors analyze forty-eight Android mobile money apps from twenty-eight countries using the automated tool Mallodroid. This is a static analysis tool for detecting TLS vulnerabilities. Then, the entire application communication flow analysis is performed on seven apps from Brazil, India, Indonesia, Thailand, and the Philippines. This analysis consists of two phases. Phase 1 is the inspection to identify the basic functionality of the application, whereas phase 2 is the reverse engineering method. The authors are interested in security errors in the following procedures: registration and login, user authentication after login and money transfers.

Bassolé *et al.* [160] discuss the security of mobile banking and mobile payment applications. Specifically, they analyze vulnerabilities in Android mobile banking and mobile payment applications in African countries. The case study is based on fifty-three mobile applications from African banks or banks with subsidiaries in Africa. The analysis procedure consists of two steps, namely the static analysis and the dynamic analysis. For static analysis, the authors use Apktool for reverse engineering, whereas for dynamic analysis, they use virustotal platform for malware detection.

Castle *et al.* [161] address the question of security vulnerabilities in digital financial services (DFS) in developing World. They define a threat model. Then, they analyze the security of 397 DFS deployments, including 197 Android apps. Finally, they interview seven developers and product managers. In the threat model, the security goals are confidentiality, integrity, and availability. Potential adversaries are customers, agents (i.e., intermediaries between users and their accounts)

and organization employees. Potential attacks include external apps, man-in-the-middle, authentication attack, denial of service, etc. For security analysis, the authors consider information leakage, Android permissions, Uniform Resource Locator (URL) and third-party libraries.

Chothia *et al.* [162] analyze TLS related vulnerabilities in mobile banking apps in the United Kingdom (UK). As case studies, they examine Android and iOS apps from fifteen banks in the UK. The most common TLS flaws are self-signed certificates, no hostname verification, protocol downgrades or weak cipher suites and Secure Sockets Layer (SSL) stripping. The first stage of the investigation is to look at well-known TLS vulnerabilities using BurpSuite, Mallodroid and SSLStrip tools. Furthermore, the authors analyze the vulnerability of using certificate pinning without hostname verification.

Darvish and Husain [163] discuss the security of Android mobile banking and mobile payment applications. They perform both static analysis and dynamic analysis on twenty-six applications. The authors use AndroBugs, APKtool and Moblizer tools for static analysis, Charles Proxy and BurpSuite tools for dynamic analysis. The security analysis highlights some vulnerabilities, such as insecure data storage and SSL certificate verification.

Osho *et al.* [164] conduct a forensic analysis of twelve Android mobile banking applications in Nigeria. This analysis is done based on the five requirements defined by OWASP's Mobile AppSec Verification Standard - level 2 (MASVS-L2). As a result, none of the analyzed applications save sensitive data in the generated back-up and none of them remove sensitive data when the application switches to the background.

Uduimoh *et al.* [165] conduct a forensic analysis of five Android mobile banking applications in Nigeria. The results reveal that all analyzed mobile banking applications save sensitive data in plaintext.

The observation made at the end of this literature review is the lack of a widely adopted framework among researchers, for assessing the security of mobile banking applications. We propose a solution to this problem by defining a concrete and comprehensive set of criteria for the security assessment of mobile banking applications.

6.3 Proposed Framework

6.3.1 Requirements

In this section, we define the requirements of the proposed framework. These requirements are inspired from the work presented in [94]. Wang and Wang [94] propose a framework for evaluating two-factor authentication schemes, which includes a set of twelve criteria. In our framework, we suggest a set of twenty-six criteria for the security assessment of mobile banking applications. The requirements of our framework are as follows:

- No redundancy. We should have a unique meaning for each criterion within a proposed set of criteria. In addition, a given criterion should not be included in another. The requirement of non-redundancy avoids having an unnecessarily long list of criteria.
- No ambiguity. This requirement refers to the fact that each criterion must have a unique interpretation in the context of assessing the security of Android mobile banking applications. Moreover, each criterion must allow to assess a mobile banking application individually, without referring to another mobile banking application.
- Comprehensiveness. The proposed set of criteria must capture a wide range of security vulnerabilities faced by Android mobile banking applications. Vulnerabilities can arise from insecure storage of sensitive data, insecure exchange of sensitive data with a remote server, misuse of cryptography, etc.

6.3.2 Criteria Set

In this section, we define a set of criteria to help evaluate the security of Android mobile banking applications. For this purpose, we integrate security vulnerabilities from previous research [18] [153] [155] [156] [157] [159] [161] [162] [166] [167] and industrial reports [95] [151] [168] [169]. The result is a set of twenty-six criteria that mobile banking applications should satisfy to ensure good security. These criteria are summarized in Table 6.1.

We distinguish five categories: 1) mobile device security; 2) data in transit; 3) data storage; 4) cryptographic misuse; and 5) others.

Table 6.1 Criteria Set

Category	**Criterion**
Mobile device security	No rooted device
	No emulator
Data in transit	Reject self-signed certificate
	Certificate hostname verification
	Reject expired or revoked certificate
	Transmit sensitive data in encrypted form
Data storage	No sensitive data in plaintext in internal storage
	No sensitive data in plaintext in external storage
	No sensitive data in plaintext in log files
	No sensitive data in plaintext in cache files
	Use *SharedPreferences* in private mode
Cryptographic misuse	No hard-coded decryption key or other credentials
	No ECB mode for encryption
	Use random IV for CBC encryption
	Use random salt for PBE
	Use more than one thousand iterations for PBE
	Use secure random value
	Use secure encryption algorithm or hash function
Others	UI screenshots disable
	Obfuscation
	No debuggable
	No third-party ads libraries

Category	Criterion
	Limit on login attempts
	Integrity check at runtime
	Disable access to app's content providers
	Control of method inputs

The first category *mobile device security* refers to the detection of insecure devices. The two criteria in this category are: 1) no rooted device; and 2) no emulator. It is necessary for sensitive applications like mobile banking applications to verify the integrity of the mobile device at runtime. Indeed, rooted devices and emulators are sources of security vulnerabilities. A rooted device is an Android device whose user has superuser rights. Thus, any application on the device can perform dangerous actions like deleting system files. In addition, access to data and resources on the device is no longer restricted. An emulator is a virtual machine that simulate a real Android device. The behavior of an application running on an emulator can be analyzed by a hacker [156].

The second category *data in transit* contains criteria related to the protection of sensitive data when they are transmitted over the communication channel. The objective is to check the vulnerability of the applications to network attacks, such as man-in-the-middle attacks and server spoofing. Data in transit faces security risks associated with the communication network between the application and the server. We focus on the TLS protocol since it is the standard used to secure communications on the Internet. The criteria in this category assess the proper implementation of the TLS protocol within the application as follows:

- Reject self-signed certificate. An application should not accept communication with a server that presents a self-signed certificate. By doing so, it accepts communication with any server, including a server controlled by a hacker. The application can then transmit sensitive data to this malicious server. The application should only communicate with a server that has a certificate signed by a trusted authority.
- Certificate hostname verification. The application must verify that the requested hostname matches the one of the received certificate. Without hostname verification, the application

may accept communication with a server that has a valid certificate signed by a trusted authority. However, it might not be the server the application is supposed to communicate with. In this case, the application may transmit sensitive data to an unknown entity.

- Reject expired or revoked certificate. This criterion verifies that the application accepts only certificates that have not passed their expiry date and have not been revoked. The failure to do this check may result in communication with a server that has been revoked and is now under the control of a hacker.
- Transmit sensitive data in encrypted form. Sensitive data must be encrypted before being transmitted over the communication channel. This can be done using the TLS protocol. Moreover, data can be encrypted before being encapsulated by the TLS protocol to provide an additional layer of security. Furthermore, we use the protocol HyperText Transfer Protocol Secure (HTTPS) instead of HyperText Transfer Protocol (HTTP) to communicate securely.

The third category of criteria is *data storage*. The aim is to verify that the sensitive data on the mobile device is properly protected. The sources of attacks can be malware or a hacker in possession of a stolen or lost mobile device. The criteria in this category are listed below:

- No sensitive data in plaintext in internal storage. In the Android operating system, the data stored in the internal memory is sandboxed by application [151]. Thus, a given application cannot access the data of another application. However, this data can be recovered by a hacker in possession of the device, using forensic tools. Therefore, it is recommended to always encrypt sensitive data before storing it in the internal storage. Hence, this data is protected from any entity that does not have the decryption key.
- No sensitive data in plaintext in external storage. As much as possible, a mobile banking application should avoid saving sensitive data in the external storage, since it is accessible by other applications including malware. When external storage is used, the data must be encrypted beforehand. Moreover, the availability of external storage should be checked since it is removable. In addition, the validity of the data must be checked, as the data can be corrupted or modified by other applications [151].

- No sensitive data in plaintext in log files. During application development, developers usually use logs for testing purposes. Sometimes, these logs remain when the application is deployed. The logs generated by an application can constitute a security vulnerability if they contain sensitive data in plaintext.
- No sensitive data in plaintext in cache files. The cache memory is often used to allow the application to access data quickly. This memory should only be used for non-sensitive data [151].
- Use *SharedPreferences* in private mode. According to Google's recommendation [151], *SharedPreferences* objects should be used in private mode. Therefore, other applications will not have access to the data stored in these objects.

The fourth category *cryptographic misuse* evaluates the implementation of cryptographic techniques within the mobile application. Cryptography can be used to protect sensitive data. Moreover, it can be used as an additional security layer for the transmitted data over a communication channel. In this case, the data is encrypted before being encapsulated by the TLS protocol. However, some rules must be respected so that a hacker in possession of the encrypted data cannot retrieve the plaintext data. These rules are presented as follows:

- No hard-coded decryption key or other credentials. If the decryption key and other credentials, such as the credentials to access a database, are hard coded in the application, they can be recovered with reverse engineering tools. The hacker can then decrypt the encrypted data or access the database. It is important not to hardcode the decryption keys and other credentials.
- No ECB mode for encryption [166]. Electronic Code Book (ECB) is a mode of operation used by block symmetric encryption. In this mode, given an encryption key, a block of plaintext will always produce the same block of ciphertext [170]. This mode of operation is not recommended for sensitive applications, such as mobile banking applications.
- Use random IV for CBC encryption [166]. Cipher Block Chaining (CBC) is another mode of operation used by block symmetric encryption. This mode of operation requires an initialization vector (IV) to generate the ciphertext from the plaintext. This initialization vector must be unpredictable to avoid attacks on the encryption algorithm [170].

- Use random salt for PBE [166]. In Password-Based Encryption (PBE), we use the notion of salt to counter dictionary and rainbow table attacks. The salt must be random and different for each password to avoid two identical passwords having the same hash. Having a constant salt is the same as having no salt, because in both cases two identical passwords will always have the same hash.
- Use more than one thousand iterations for PBE [166]. The notion of iteration is used in the PBE to counter brute force attacks. As per the National Institute of Standards and Technology (NIST) recommendation, “the iteration count shall be selected as large as possible, as long as the time required to generate the key using the entered password is acceptable for the users” [171]. The minimum recommended value is one thousand.
- Use secure random value. Random values used for cryptographic purposes must have high entropy. Android provides the *SecrureRandom* class to generate secure random numbers.
- Use secure encryption algorithm or hash function. Some encryption algorithms and hash functions have known security flaws. They should no longer be used to protect sensitive data. Examples of insecure encryption algorithms and hash functions are RC2, MD4, MD5, SHA1 [172].

The last category *others* contains various criteria that do not fit into any of the above presented categories. These criteria are presented as follows:

- UI screenshots disable. Malware can take screenshots to collect sensitive data like passwords when entered by the user. Mobile banking applications must prevent user interface (UI) screenshots.
- Obfuscation. Obfuscation is a technique that makes it difficult to reverse engineer the application. Obfuscation is, therefore, useful to protect the source code of the application.
- No debuggable. If the application is debuggable, then it can be connected to a debugger to analyze the application's behavior step by step [156]. The *android:debuggable* attribute present in the Android manifest allows to determine if an application is debuggable (true value) or not (false value).

- No third-party ads libraries. Since a mobile banking application is a sensitive application, it should not use third-party ads libraries. This is because third-party ads libraries can be sources of security vulnerabilities [156].
- Limit on login attempts. To prevent brute force attacks, the application must limit the number of incorrect login attempts for a single user.
- Integrity check at runtime. At runtime, the application must check that it has not been modified. This prevents the repackaging attack where a hacker injects malicious code into the original application.
- Disable access to app's content providers. According to Google's recommendation [151], access to the *ContentProvider* object must be disabled so that other applications cannot access it.
- Control of method inputs. All method inputs should be considered untrusted. Therefore, it is important to check the inputs to prevent attacks like code injection.

In this set of criteria, we did not consider the use of unnecessary permissions because this criterion depends on the services offered by each application. Indeed, applications that have the remote cheque deposit feature require access to the smartphone camera to scan a cheque, while other applications would not. Furthermore, we do not provide weights to the different criteria, since a bank may decide to give preference to certain criteria over others, depending on its specific goals.

6.3.3 Evaluation

In this Section, we show that the proposed framework satisfies the defined requirements: *no redundancy*, *no ambiguity* and *comprehensiveness*.

- No redundancy. Chen *et al.* [18] proposed the most recent set of security weaknesses of the mobile banking applications. The weakness *stored on SD card* can be confused with the weakness *written in text files*. Indeed, when sensitive data are written in text files, these text files are stored either in the internal memory of a mobile device or in the external memory (e.g., SD card). If the file is stored on an SD card, then the weaknesses *stored on SD card* and *written in text files* are identical. Furthermore, the weaknesses *uses invalid certificates* and *uses invalid certificate authentication*, have the same meaning. Hence, the redundancy

is in this set of security weaknesses. To avoid redundancy in our set of criteria, we have divided the criteria into five categories. Anyone can check that no one criterion has the same meaning as another, nor is included in another criterion.

- No ambiguity. To eliminate ambiguity in the proposed framework, we have clearly described each criterion in Section 6.3.2. As a result, each criterion has a unique interpretation. Furthermore, each criterion allows to assess a mobile banking application individually, without referring to another mobile banking application.
- Comprehensiveness. The proposed set of criteria covers all the risks listed in the *OWASP Mobile Top 10* project. The criterion *no hard-coded decryption key or other credentials* refers to the M1 risk. In this case, the application developer hardcodes the decryption keys instead of using the Android Keystore System, which offers better protection of cryptographic keys. The criteria of the category *data storage* cover the M2 risk, while the M3 risk is covered by the criteria of the category *data in transit*. The criteria corresponding to the M4 and M6 risks are *reject self-signed certificate*, *certificate hostname verification*, *reject expired or revoked certificate*, *no hard-coded decryption key or other credentials* and *limit on login attempts*. The criteria of the category *cryptographic misuse* cover the M5 risk. The criterion *control of method inputs* refers to the M7 risk. The criterion *integrity check at runtime* deals with the M8 risk, while the criterion *obfuscation* deals with the M9 risk. Finally, the M10 risk is addressed by the criteria *no sensitive data in plaintext in log files* and *no debuggable*. Since the proposed set of criteria addresses all the risks listed in the *OWASP Mobile Top 10* project, we can consider this set as comprehensive.

6.4 Case study

In this section, we present the methodology used to assess the security of Android mobile banking applications. First, we select seven mobile banking applications from the Canadian market. Second, we describe the environment set up for the evaluation. Finally, we discuss the obtained results.

6.4.1 Dataset

Our dataset consists of seven Android mobile banking applications from the Canadian market. We collected the applications on the Google Play Store during the month of March 2022. Table 6.2

presents a summary of the applications selected for our case study. We anonymized the banks whose applications were assessed.

Table 6.2 Dataset

Mobile banking application	Version	Number of downloads
B1	4.21	1M+
B2	8.35.0	5M+
B3	8.35.1	1M+
B4	20.36.1	1M+
B5	5.32.1	1M+
B6	4.13.0	500k+
B7	17.1.24	10k+

6.4.2 Assessment Environment

In this case study, we focus only on the criteria of the category *data in transit*. We believe this is the most important category because the criteria in this category affect both the mobile application and the server. Moreover, this category addresses the third risk of the *OWASP Mobile Top 10*, which is *insecure communication*. Finally, a hacker can easily intercept traffic on an insecure wireless communication channel.

To assess mobile banking applications against the criteria of the category data in transit, we adopt the dynamic analysis. Our methodology is inspired from the work of Bojjagani and Sastry [153].

Figure 6.1 shows the network topology for dynamic analysis. The mobile banking application is installed and running on the mobile device (i.e., Samsung Galaxy S10e with Android version 12). All traffic between the mobile device and the bank's server passes through the fake access point. This fake access point helps to simulate the man-in-the-middle attack. For this purpose, we use a personal computer with an Intel Core i5-7200U 2.50 GHz processor, 8 GB RAM and Ubuntu 20.04.4 LTS 64-bit operating system. This personal computer is equipped with Nogotofail [96].

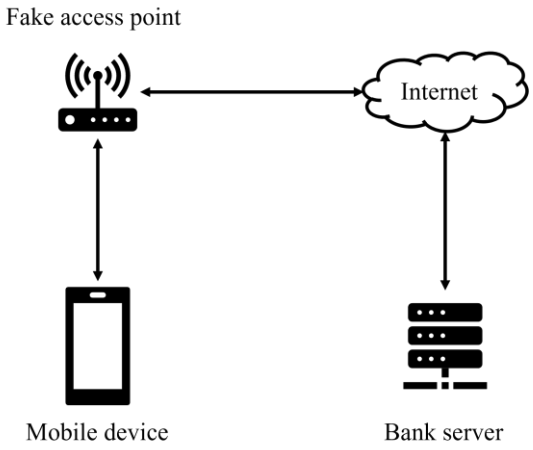

Figure 6.1 Network topology for dynamic analysis

Nogotofail [96] is an open-source software developed by Google to evaluate network security issues. Through this software, we can perform man-in-the-middle attacks on connections between the mobile banking application and the bank's server.

Login to the selected mobile banking applications requires bank accounts with the respective banks. Since we do not have this facility, we just made login attempts to the various applications with invalid credentials. Then, the generated traffic is attacked with the Nogotofail software.

6.4.3 Findings

Thanks to the Nogotofail tool, we were able to conduct a man-in-the-middle attack on the TLS connections initiated by the mobile banking applications. Table 6.3 presents the security assessment results.

- Reject self-signed certificate. Instead of the real server certificate, we send to the mobile application a self-signed certificate by Nogotofail. All mobile banking applications rejected the self-signed certificate and aborted the connection.
- Certificate hostname verification. In this case, the attack consists of presenting the mobile banking application with a trusted certificate with a domain name other than the bank's server. For this purpose, we create a certificate authority (CA) that we add to our mobile device as a trusted CA. We then generate a trusted certificate with a different domain name than the bank's server. All mobile banking applications detected the invalid hostname and terminated the connection.

- Reject expired or revoked certificate. In this case, we present the mobile banking application with an expired certificate, but signed by a trusted CA. All mobile banking applications aborted the connection.
- Transmit sensitive data in encrypted form. Here, we inspect the traffic between the mobile application and the server for sensitive plaintext data. None of the mobile banking applications transmit sensitive data in plaintext.

All the assessed applications meet the security criteria of the category *data in transit*. We can conclude that mobile banking applications in the Canadian market pay attention to the security of data in transit. However, the assessment conducted is partial, since we are unable to log into the applications and attack the traffic generated by other features, such as balance inquiry and money transfer.

Table 6.3 Security Assessment Results

Mobile banking application	**Reject self-signed certificate**	**Certificate hostname verification**	**Reject expired or revoked certificate**	**Transmit sensitive data in encrypted form**
B1	Yes	Yes	Yes	Yes
B2	Yes	Yes	Yes	Yes
B3	Yes	Yes	Yes	Yes
B4	Yes	Yes	Yes	Yes
B5	Yes	Yes	Yes	Yes
B6	Yes	Yes	Yes	Yes
B7	Yes	Yes	Yes	Yes

6.5 Conclusion

In this paper, we introduced a framework for assessing the security of Android mobile banking applications. This framework consists of a set of twenty-six criteria that mobile banking applications must meet to maintain an important level of security. We evaluated the proposed

framework based on three requirements: *no redundancy*, *no ambiguity* and *comprehensiveness*. Furthermore, we assessed the security of the mobile banking applications of seven major Canadian banks. This work can benefit both security researchers and Canadian banks. Security researchers can use the set of criteria for future security assessment of other mobile banking applications. In addition, they can design an automatic security assessment tool taking into consideration the wide range of criteria proposed in this article. Canadian banks, in turn, can convince more customers to trust them, given the important level of security of their mobile banking applications.

Acknowledgment

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

CHAPITRE 7 DISCUSSION GÉNÉRALE

Ce chapitre est consacré à la discussion de l'ensemble des travaux réalisés dans cette thèse. Dans un premier temps, nous présentons les aspects méthodologiques qui nous ont orientés vers l'énoncé des objectifs de recherche et l'atteinte de ceux-ci. Dans un deuxième temps, nous faisons l'analyse des résultats obtenus.

7.1 Aspects méthodologiques

La thématique principale abordée dans cette thèse est la sécurité des communications dans un environnement client mobile/serveur. Pour traiter cette thématique, le travail effectué au préalable dans le but d'identifier les problèmes de sécurité dans un environnement client mobile/serveur, est celui de la revue de la littérature. De cette revue de littérature, il ressort que, les problèmes de sécurité dans un environnement client mobile/serveur peuvent être classés en fonctions de trois zones d'action de ces problèmes : le client mobile, l'infrastructure réseau et le serveur.

Au vu de ces trois zones d'action, cette thèse est divisée en trois volets. Le premier volet s'attaque aux problèmes de sécurité liés à l'infrastructure réseau. Le deuxième volet aborde le problème de déni de service au niveau du serveur. Le troisième volet quant à lui, traite le problème de vulnérabilité des applications au niveau du client mobile.

La méthodologie adoptée dans chacun de ces volets se présente comme suit :

- Une revue critique de la littérature dans le but d'identifier les lacunes des solutions existantes;
- La conception d'une solution qui prend en compte les lacunes des solutions existantes;
- L'implémentation et l'évaluation de la solution proposée.

Dans le premier volet, nous proposons un protocole d'authentification mutuelle et d'échange de clé entre le client mobile et le serveur. Le but de ce protocole est de créer un canal de communication sécurisé entre le client mobile et le serveur.

Dans le deuxième volet, nous concevons un système de détection des attaques de déni de service distribué *« Distributed Denial of Service » (DDoS)* par inondation. Le but de ce système de

détection est de maintenir la disponibilité du serveur pour qu'il puisse répondre aux requêtes des clients mobiles légitimes.

Dans le troisième volet, nous proposons un cadre pour l'évaluation de la sécurité des applications de banque mobile. Le but est d'apporter une solution au problème de vulnérabilité dans les applications mobiles, en particulier dans celles de banque mobile.

7.2 Analyse des résultats

Les résultats obtenus au terme des travaux de cette thèse contribuent à améliorer la sécurité dans un environnement client mobile/serveur.

Pour sécuriser le canal de communication entre le client mobile et le serveur, nous proposons un protocole d'authentification mutuelle et d'échange de clé. L'utilisation de la cryptographie sur les courbes elliptiques dans ce protocole nous permet d'avoir un protocole sécuritaire qui consomme peu de ressources. L'introduction de la carte *« Subscriber Identity Module » (SIM)* dans ce protocole présente deux principaux avantages. D'une part, la carte SIM élimine l'inconvénient d'avoir un matériel supplémentaire que nous observons dans les protocoles utilisant les cartes à puce. D'autre part, la carte SIM est un élément sécurisé pour le stockage des données secrètes des utilisateurs. L'évaluation de la sécurité et des performances du protocole proposé montre un meilleur compromis entre la sécurité et les performances, par rapport aux protocoles existants. Le protocole proposé est approprié pour une variété de clients mobiles, tels que les téléphones intelligents, les tablettes, les montres connectées et les capteurs corporels.

Pour assurer la disponibilité du serveur, nous proposons un système de détection des attaques DDoS par inondation. L'adoption de l'entropie de l'information nous permet de distinguer le trafic légitime du trafic d'attaque, en se basant uniquement sur l'*adresse IP source* contenue dans les paquets. En conséquence, le temps de traitement des paquets et le délai de détection des attaques sont réduits. Contrairement aux systèmes de détection existants qui détectent la présence d'une attaque sans identifier les paquets d'attaque, notre système détecte la présence d'une attaque et identifie les paquets d'attaque. L'évaluation de performances nous donne un taux de détection de 100 % et un délai de détection de 7,89 secondes.

Pour faire face à la vulnérabilité des applications mobiles, nous proposons un cadre pour évaluer la sécurité des applications de banque mobile. La revue de littérature effectuée dans ce domaine

nous permet de conclure qu'il manque un cadre largement adopté par les chercheurs pour évaluer la sécurité des applications de banque mobile. Le cadre proposé vise à combler ce manque. L'évaluation de ce cadre montre qu'il satisfait aux requis de *non-redondance*, de *non-ambiguïté* et de *complétude*. Notre étude de cas porte sur l'évaluation de sept applications de banque mobile du marché canadien. Les résultats montrent que les données sensibles échangées entre ces sept applications et leurs serveurs respectifs sont adéquatement protégées. Par conséquent, les banques canadiennes accordent une attention particulière à la sécurité de leurs applications de banque mobile.

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 donné lieu à un article scientifique présenté dans les chapitres 4, 5 et 6. Le chapitre suivant présente les limites de cette thèse et les potentielles avenues de recherche.

CHAPITRE 8 CONCLUSION

Ce chapitre présente un récapitulatif des différents travaux réalisés dans le cadre de cette thèse. Premièrement, nous faisons un rappel des contributions de cette thèse. Ensuite, nous identifions et exposons les limites des travaux réalisés. Enfin, nous faisons des recommandations qui peuvent faire l'objet des travaux futurs.

8.1 Contributions de la thèse

L'objectif principal de cette thèse est de concevoir des modèles robustes pour améliorer la sécurité des communications dans un environnement client mobile/serveur. Dans un contexte marqué par la prolifération des téléphones intelligents et des objets connectés, il est important d'assurer la sécurité des données des utilisateurs et de garantir le respect de leur vie privée. Dans cette optique, les travaux de cette thèse contribuent à l'amélioration de la sécurité dans un environnement client mobile/serveur, répartis sur trois volets. Plus précisément, dans le premier volet de cette thèse, nous proposons un protocole d'authentification mutuelle et d'échange de clé, pour sécuriser le canal de communication entre un client mobile et un serveur. Dans le deuxième volet, nous concevons un système de détection des attaques de déni de service distribué *« Distributed Denial of Service » (DDoS)* par inondation pour assurer la disponibilité du serveur. Dans le dernier volet, nous proposons un cadre *« Framework »* pour l'évaluation de la sécurité des applications de banque mobile en vue de limiter les vulnérabilités qui y sont présentes.

Les principales contributions de cette thèse se résument comme suit :

1. Conception d'un nouveau protocole d'authentification mutuelle et d'échange de clé entre un client mobile et un serveur distant, dans un environnement client mobile/serveur. Le protocole proposé permet de sécuriser le canal de communication entre le client mobile et le serveur, tout en respectant les contraintes de ressources du client mobile. Ce protocole proposé améliore les protocoles existants, en combinant les notions théoriques, telles que la cryptographie sur les courbes elliptiques et le nonce aléatoire. La simulation sur le logiciel *« Scyther »* [24] démontre la sécurité dudit protocole.

2. Conception d'un système de détection des attaques DDoS par inondation. Le système de détection proposé utilise l'approche basée sur l'anomalie et la notion d'entropie de l'information pour détecter les attaques dans un délai réduit. L'entropie est calculée en

utilisant comme variable aléatoire l'*adresse IP source* des paquets du trafic réseau. L'avantage du système de détection proposé par rapport aux systèmes de détection existants, est qu'il est capable de détecter la présence d'une attaque et identifier les paquets qui sont responsables de cette attaque. L'évaluation des performances montre un taux de détection de 100 %.

3. Conception d'un algorithme de seuil dynamique. L'algorithme de seuil dynamique proposé est utilisé dans le système de détection des attaques DDoS par inondation. Dans ce système, nous utilisons l'entropie de l'information, qui est une approche basée sur les statistiques. La difficulté à fixer un seuil optimal est un problème rencontré dans les approches basées sur les statistiques [85]. Nous nous référons au théorème de Chebyshev [87], pour calculer des seuils qui varient en fonction des changements observés dans le trafic légitime.

4. Conception d'un cadre pour l'évaluation de la sécurité des applications de banque mobile. Le cadre proposé a pour but de limiter les vulnérabilités présentes dans les applications de banque mobile. Ce cadre est constitué de vingt-six critères répartis dans cinq catégories : *sécurité du périphérique mobile*, *données en transit*, *stockage des données*, *mauvaise utilisation cryptographique* et *autres*. Ce cadre satisfait aux requis de *non-redondance*, de *non-ambiguïté* et de *complétude*.

5. Évaluation de la sécurité des applications Android de banque mobile de sept banques canadiennes. Dans cette contribution, nous évaluations la sécurité des applications Android de banque mobile déployées sur le marché canadien. Les critères utilisés pour l'évaluation sont ceux appartenant à la catégorie *données en transit* du cadre proposé à la quatrième contribution. L'évaluation montre que les données en transit sont adéquatement protégées par les banques canadiennes. Les utilisateurs peuvent ainsi faire confiance à ces applications de banque mobile du marché canadien.

8.2 Limitations des travaux réalisés

L'identification des limites de tout travail est nécessaire à l'avancement des connaissances. Les limites des travaux réalisés dans le cadre de cette thèse sont présentées comme suit :

- L'utilisation de la carte *« Subscriber Identity Module » (SIM)* dans le protocole d'authentification mutuelle présenté au chapitre 4, rend ce protocole inapproprié pour des

périphériques mobiles qui ne supportent pas la carte SIM. En effet, il existe des périphériques mobiles, tels que certaines tablettes, qui ne prennent pas en charge la carte SIM.

- Le calcul du temps d'exécution des opérations cryptographiques (e.g., la génération des nombres aléatoires, les fonctions de hachage, le chiffrement symétrique, etc.) est effectué sur un téléphone intelligent. Cependant, vu que le protocole d'authentification mutuelle proposé fait appel à la carte SIM pour l'exécution des opérations cryptographiques, il serait utile d'évaluer le temps d'exécution de ces différentes opérations sur une carte SIM.
- Le système de détection des attaques DDoS par inondation, présenté au chapitre 5, ne prend en compte que deux types de trafic : le trafic légitime et le trafic d'attaque DDoS par inondation. Toutefois, il existe deux autres types de trafic qui pourraient survenir dans un réseau informatique : le trafic d'attaque DDoS à faible débit *« low-rate DDoS »* et le trafic des évènements éclairs *« flash events » (FE)*. Les attaques *« low-rate DDoS »* [173] génèrent du trafic à faible débit, qui peut facilement se confondre au trafic légitime. Les FE sont des augmentations brusques du trafic réseau causées par des utilisateurs légitimes [174]. Ce type de trafic peut facilement se confondre à une attaque DDoS par inondation.
- Pour évaluer notre système de détection des attaques DDoS par inondation, nous avons considéré l'ensemble de données *CICIDS2017* [28] et des simulations à travers le simulateur réseau *GNS3* [27]. Néanmoins, le réseau configuré dans *GNS3* est un réseau de petite taille, constitué de six clients légitimes et de trois clients malveillants. Il serait intéressant d'étudier le comportement de notre système de détection dans un réseau réel de grande taille.
- Le cadre pour l'évaluation de la sécurité des applications de banque mobile, présenté au chapitre 6, est dépendant du système d'exploitation Android. Il ne prend pas en compte les vulnérabilités spécifiques à d'autres systèmes d'exploitation mobiles, tels que iOS.
- L'évaluation de la sécurité des applications Android de banque mobile effectuée est partielle. En effet, nous avons mené des attaques uniquement sur le trafic généré par l'opération de connexion aux applications avec des identifiants invalides. La connexion aux différentes applications nécessite un compte bancaire auprès des banques respectives. N'ayant pas cette facilité, il nous était impossible de nous connecter aux applications pour

mener des attaques sur le trafic généré par d'autres opérations, telles que le transfert d'argent et le paiement de facture.

8.3 Travaux futurs

Pour conclure cette thèse, nous faisons des recommandations sur des directions de recherches futures, dont certaines émanent des limites relevées à la section précédente. Ces recommandations sont les suivantes :

- L'évaluation du temps d'exécution des opérations cryptographiques sur une carte SIM. Les cartes SIM utilisent le système d'exploitation *« Java Card »* [175], qui supporte les opérations cryptographiques en général, et la cryptographie sur les courbes elliptiques en particulier.
- La conception d'un système de détection des attaques DDoS qui prend en compte les différents types de trafic suivants : le trafic légitime, le trafic d'attaque DDoS par inondation, le trafic *« low-rate DDoS »* et le trafic FE.
- La détection des attaques DDoS à la source. Le système de détection des attaques DDoS proposé est déployé au niveau de la cible, à savoir le serveur. Le trafic d'attaques DDoS par inondation provient généralement de plusieurs périphériques sous le contrôle d'un cybercriminel. Une piste de recherche serait de détecter les attaques DDoS dès la source, en identifiant les périphériques sous le contrôle d'un cybercriminel.
- L'extension du cadre pour l'évaluation de la sécurité des applications de banque mobile, afin de prendre en compte les vulnérabilités spécifiques au système d'exploitation iOS. Android et iOS étant les leaders des systèmes d'exploitation mobiles, avec environ 95 % des parts de marché en janvier 2022 [149].
- La conception d'un outil automatisé pour l'évaluation de la sécurité des applications de banque mobile. Cet outil peut se baser sur les analyses statique et dynamique, tout en exploitant le large éventail de critères définis dans notre cadre.

RÉFÉRENCES

[1] Statistics Canada, «Comment les Canadiens restent-ils connectés?,» 29 octobre 2019. [En ligne]. Available: https://www150.statcan.gc.ca/n1/pub/11-627-m/11-627-m2019063-eng.htm. [Accès le 12 novembre 2019].

[2] Cisco, «Cisco Annual Internet Report (2018–2023) White Paper,» 9 mars 2020. [En ligne]. Available: https://www.cisco.com/c/en/us/solutions/collateral/executive-perspectives/annual-internet-report/white-paper-c11-741490.html. [Accès le 27 avril 2022].

[3] ARcare, «Notice of data privacy incident,» [En ligne]. Available: https://www.arcare.net/wp-content/themes/altitude-pro/security_notice.html. [Accès le 29 avril 2022].

[4] E. A. Lile, «Client/Server Architecture: A Brief Overview,» *Journal of Systems Management*, vol. 44, n° 12, pp. 26-29, 1993.

[5] A. Berson, Client/Server Architecture, New York: McGraw-Hill, 1996.

[6] S. Pierre, Réseaux et systèmes informatiques mobiles - Fondements, architectures et applications, Montréal: Presses internationales Polytechnique, 2003.

[7] R. S. Somula et R. Sasikala, «A Survey on Mobile Cloud Computing: Mobile Computing + Cloud Computing (MCC = MC + CC),» *Scalable Computing: Practice and Experience*, vol. 19, n° 4, pp. 309-337, 2018.

[8] S. Kumar, M. Tyagi, A. Khanna et V. Fore, «A Survey of Mobile Computation Offloading: Applications, Approaches and Challenges,» chez *2018 International Conference on Advances in Computing and Communication Engineering (ICACCE)*, Paris, France, 2018.

[9] M. M. Keith, Everyday cryptography : fundamental principles and applications, New York: Oxford University Press, 2017.

[10] R. Kumar et M. P. S. Bhatia, «A Systematic Review of the Security in Cloud Computing: Data Integrity, Confidentiality and Availability,» chez *2020 IEEE International Conference on Computing, Power and Communication Technologies (GUCON)*, Greater Noida, India, 2020.

[11] E. Gillier, «Sécurisez vos données avec la cryptographie,» OpenClassrooms, 13 août 2019. [En ligne]. Available: https://openclassrooms.com/fr/courses/1757741-securisez-vos-donnees-avec-la-cryptographie. [Accès le 19 septembre 2019].

[12] E. Barker, «Recommendation for Key Management, Part 1: General,» National Institute of Standards and Technology, NIST Special Publication 800-57 Part 1, rev.4, Gaithersburg, Maryland, 2016.

[13] W. Diffie et M. Hellman, «New directions in cryptography,» *IEEE Transactions on Information Theory*, vol. 22, n° 6, pp. 644-654, 1976.

[14] J. Hoffstein, J. Pipher et J. H. Silverman, An Introduction to Mathematical Cryptography, New York, NY: Springer, 2014, pp. 299-371.

[15] S. Vanstone et A. J. Menezes, Guide to Elliptic Curve Cryptography, New York, NY: Springer, 2004, pp. 153-204.

[16] J.-L. Tsai et N.-W. Lo, «A Privacy-Aware Authentication Scheme for Distributed Mobile Cloud Computing Services,» *IEEE Systems Journal*, vol. 9, n° 3, pp. 805-815, 2015.

[17] P. Subramaniam et K. M. Jeet, «Review of Security in Mobile Edge Computing with Deep Learning,» chez *2019 Advances in Science and Engineering Technology International Conferences (ASET)*, Dubai, United Arab Emirates, 2019.

[18] S. Chen, L. Fan, G. Meng, T. Su, M. Xue, Y. Xue, Y. Liu et L. Xu, «An Empirical Assessment of Security Risks of Global Android Banking Apps,» chez *2020 IEEE/ACM 42nd International Conference on Software Engineering (ICSE)*, Seoul, Korea (South), 2020.

[19] S. A. S. Muhseen et A. S. Elameer, «A Review in Security Issues and Challenges on Mobile Cloud Computing (MCC),» chez *2018 1st Annual International Conference on Information and Sciences (AiCIS)*, Fallujah, Iraq, 2018.

[20] E. Cox, «Mirai IoT Botnet: 5 Fast Facts You Need to Know,» 23 octobre 2016. [En ligne]. Available: https://heavy.com/tech/2016/10/mirai-iot-botnet-internet-of-things-ddos-attacks-internet-outage-blackout-why-is-internet-down/. [Accès le 27 avril 2022].

[21] A. A. Ahmed et K. Wendy, «Mutual authentication for mobile cloud computing: Review and suggestion,» chez *2017 IEEE Conference on Application, Information and Network Security (AINS)*, Miri, Malaysia, 2017.

[22] D. Brown, «SEC 1: Elliptic Curve Cryptography,» Standard for Efficient Cryptography Group, 21 mai 2009. [En ligne]. Available: https://www.secg.org/sec1-v2.pdf.

[23] M. Burrows, M. Abadi et R. Needham, «A Logic of Authentication,» *ACM Transactions on Computer Systems*, vol. 8, n° 1, pp. 18-36, 1990.

[24] C. Cremers, «The Scyther Tool: Verification, Falsification, and Analysis of Security Protocols,» chez *International Conference on Computer Aided Verification (CAV)*, 2008.

[25] C. E. Shannon, «A mathematical theory of communication,» *The Bell System Technical Journal*, vol. 27, n° 3, pp. 379-423, 1948.

[26] S. D. Çakmakçı, T. Kemmerich, T. Ahmed et N. Baykal, «Online DDoS attack detection using Mahalanobis distance and Kernel-based learning algorithm,» *Journal of Network and Computer Applications*, vol. 168, 2020.

[27] GNS3, «GNS3 (version 2.2.17),» [En ligne]. Available: https://www.gns3.com/.

[28] I. Sharafaldin, A. H. Lashkari et A. A. Ghorbani, «Toward Generating a New Intrusion Detection Dataset and Intrusion Traffic Characterization,» chez *4th International Conference on Information Systems Security and Privacy (ICISSP)*, 2018.

[29] A. N. Khan, M. M. Kiah, S. U. Khan et S. A. Madani, «Towards secure mobile cloud computing: A survey,» *Future Generation Computer Systems*, vol. 29, n° 5, pp. 1278-1299, 2013.

[30] R. Roman, J. Lopez et M. Mambo, «Mobile edge computing, Fog et al.: A survey and analysis of security threats and challenges,» *Future Generation Computer Systems*, vol. 78, pp. 680-698, 2018.

[31] Vishal, B. Kaur et S. Jangra, «Assessment of Different Security Issues, Threats with Their Detection and Prevention Security Models in Mobile Cloud Computing (MCC),» chez *International Conference on Advanced Informatics for Computing Research*, 2018.

[32] M. Ficco et F. Palmieri, «Introducing Fraudulent Energy Consumption in Cloud Infrastructures: A New Generation of Denial-of-Service Attacks,» *IEEE Systems Journal*, vol. 11, n° 2, pp. 460-470, 2015.

[33] A. Khanna, A. Kero, D. Kumar et A. Agarwal, «Adaptive mobile computation offloading for data stream applications,» chez *2017 3rd International Conference on Advances in Computing,Communication & Automation (ICACCA) (Fall)*, Dehradun, India, 2017.

[34] Vishal, B. Kaur et S. Jangra, «Comparative Analysis of Security Threats in Mobile Cloud Computing Environment,» chez *2018 7th International Conference on Reliability, Infocom Technologies and Optimization (Trends and Future Directions) (ICRITO)*, Noida, India, 2018.

[35] D. X. J. Juárez et P. Cedillo, «Security of mobile cloud computing: A systematic mapping study,» chez *2017 IEEE Second Ecuador Technical Chapters Meeting (ETCM)*, Salinas, Ecuador, 2017.

[36] J. Zhang, B. Chen, Y. Zhao, X. Cheng et F. Hu, «Data Security and Privacy-Preserving in Edge Computing Paradigm: Survey and Open Issues,» *IEEE Access*, vol. 6, pp. 18209-18237, 2018.

[37] M. B. Mollah, M. A. K. Azad et A. Vasilakos, «Security and privacy challenges in mobile cloud computing: Survey and way ahead,» *Journal of Network and Computer Applications*, vol. 84, pp. 38-54, 2017.

[38] O. S. J. Nisha et S. M. Bhanu, «A Survey on Code Injection Attacks in Mobile Cloud Computing Environment,» chez *2018 8th International Conference on Cloud Computing, Data Science & Engineering (Confluence)*, Noida, India, 2018.

[39] T. Ylonen et C. Lonvick, «The Secure Shell (SSH) Protocol Architecture,» RFC 4251, IETF, 2006.

[40] T. Ylonen et C. Lonvick, «The Secure Shell (SSH) Transport Layer Protocol,» RFC 4253, IETF, 2006.

[41] T. Ylonen et C. Lonvick, «The Secure Shell (SSH) Authentication Protocol,» RFC 4252, IETF, 2006.

[42] T. Ylonen et C. Lonvick, «The Secure Shell (SSH) Connection Protocol,» RFC 4254, IETF, 2006.

[43] S. Kent et K. Seo, «Security Architecture for the Internet Protocol,» RFC 4301, IETF, 2005.

[44] S. Kent, «IP Authentication Header,» RFC 4302, IETF, 2005.

[45] S. Kent, «IP Encapsulating Security Payload (ESP),» RFC 4303, IETF, 2005.

[46] E. Rescorla, «The Transport Layer Security (TLS) Protocol Version 1.3,» RFC 8446, IETF, 2018.

[47] C. Kaufman, P. Hoffman, Y. Nir, P. Eronen et T. Kivinen, «Internet Key Exchange Protocol Version 2 (IKEv2),» RFC 7296, IETF, 2014.

[48] D. M. Ajay et E. Umamaheswari, «Packet Encryption for Securing Real-Time Mobile Cloud Applications,» *Mobile Networks and Applications*, vol. 24, n° 4, pp. 1249-1254, 2019.

[49] A. Fournier, F. El Khoury et S. Pierre, «A Client/Server Malware Detection Model Based on Machine Learning for Android Devices,» *IoT*, vol. 2, n° 3, pp. 355-374, 2021.

[50] 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, n° 23, 2021.

[51] A. Inani, C. Verma et S. Jain, «A Machine Learning Algorithm TsF K-NN Based on Automated Data Classification for Securing Mobile Cloud Computing Model,» chez *2019 IEEE 4th International Conference on Computer and Communication Systems (ICCCS)*, Singapore, 2019.

[52] M. Ali et L. T. Jung, «Confidentiality Based File Attributes and Data Classification Using TsF-KNN,» chez *2015 5th International Conference on IT Convergence and Security (ICITCS)*, Kuala Lumpur, Malaysia, 2015.

[53] Y. Xie, H. Wen, B. Wu, Y. Jiang et J. Meng, «A Modified Hierarchical Attribute-Based Encryption Access Control Method for Mobile Cloud Computing,» *IEEE Transactions on Cloud Computing*, vol. 7, n° 2, pp. 383-391, 2015.

[54] V. Goyal, O. Pandey, A. Sahai et B. Waters, «Attribute-based encryption for fine-grained access control of encrypted data,» chez *CCS '06 Proceedings of the 13th ACM conference on Computer and communications security*, Alexandria, Virginia, USA, 2006.

[55] R. Sandip, A. K. Das, S. Chatterjee, N. Kumar, S. Chattopadhyay et J. J. P. C. Rodrigues, «Provably Secure Fine-Grained Data Access Control Over Multiple Cloud Servers in Mobile Cloud Computing Based Healthcare Applications,» *IEEE Transactions on Industrial Informatics*, vol. 15, n° 1, pp. 457-468, 2018.

[56] S. Keykhaie et S. Pierre, «A Generic Model for Privacy-Preserving Authentication on Smartphones,» chez *2021 IEEE International Systems Conference (SysCon)*, Vancouver, BC, Canada, 2021.

[57] S. Keykhaie et S. Pierre, «Mobile Match on Card Active Authentication Using Touchscreen Biometric,» *IEEE Transactions on Consumer Electronics*, vol. 66, n° 4, pp. 376-385, 2020.

[58] S. Keykhaie et S. Pierre, «Lightweight and Secure Face-based Active Authentication for Mobile Users,» *IEEE Transactions on Mobile Computing*, 2021.

[59] K. Fan, H. Li, W. Jiang, C. Xiao et Y. Yang, «Secure Authentication Protocol for Mobile Payment,» *Tsinghua Science and Technology*, vol. 23, n° 5, pp. 610-620, 2018.

[60] A. T. Purnomo, Y. S. Gondokaryono et C. Kim, «Mutual Authentication in Securing Mobile Payment System using Encrypted QR Code based on Public Key Infrastructure,» chez *2016 IEEE 6th International Conference on System Engineering and Technology*, Bandung, 2016.

[61] E. Munivel et A. Kannammal, «New Authentication Scheme to Secure against the Phishing Attack in the Mobile Cloud Computing,» *Security and Communication Networks*, vol. 2019, n° 5141395, 2019.

[62] A. Abuarqoub, «A Lightweight Two-Factor Authentication Scheme for Mobile Cloud Computing,» chez *ICFNDS '19 Proceedings of the 3rd International Conference on Future Networks and Distributed Systems*, Paris, France, 2019.

[63] S. Dey, Q. Ye et S. Sampalli, «AMLT: A Mutual Authentication Scheme for Mobile Cloud Computing,» chez *2018 IEEE International Conference on Internet of Things (iThings) and*

IEEE Green Computing and Communications (GreenCom) and IEEE Cyber, Physical and Social Computing (CPSCom) and IEEE Smart Data (SmartData), Halifax, NS, Canada,, 2018.

[64] S. Dey, S. Sampalli et Q. Ye, «MDA: Message Digest-based Authentication for Mobile Cloud Computing,» *Journal of Cloud Computing: Advances, Systems and Applications,* vol. 5, n° 18, 2016.

[65] S. Dey, S. Sampalli et Q. Ye, «A Light-weight Authentication Scheme Based on Message Digest and Location for Mobile Cloud Computing,» chez *2014 IEEE 33rd International Performance Computing and Communications Conference (IPCCC)*, Austin, TX, USA, 2014.

[66] A. C. Donald et D. L. Arockiam, «Key Based Mutual Authentication (KBMA) Mechanism for Secured Access in MobiCloud Environment,» chez *2015 International Conference on Mechanical Engineering and Electrical Systems (ICMES 2015)*, 2016.

[67] J. Mo, Z. Hu et Y. Lin, «Remote User Authentication and Key Agreement for Mobile Client–Server Environments on Elliptic Curve Cryptography,» *The Journal of Supercomputing,* vol. 74, n° 11, pp. 5927-5943, 2018.

[68] J. Mo, Z. Hu, H. Chen et W. Shen, «An Efficient and Provably Secure Anonymous User Authentication and Key Agreement for Mobile Cloud Computing,» *Wireless Communications and Mobile Computing,* vol. 2019, n° 4520685, 2019.

[69] T.-H. Chen, H. Yeh et W. Shih, «An Advanced ECC Dynamic ID-Based Remote Mutual Authentication Scheme for Cloud Computing,» chez *2011 Fifth FTRA International Conference on Multimedia and Ubiquitous Engineering*, Loutraki, Greece, 2011.

[70] L. Xiong, T. Peng, D. Peng, H. Liang et Z. Liu, «An Efficient Privacy-aware Authentication Scheme for Distributed Mobile Cloud Computing Services without Bilinear Pairings,» *Journal of Information Science and Engineering,* 2018.

[71] S. Jegadeesan, M. Azees, P. M. Kumar, G. Manogaran, N. Chilamkurti, R. Varatharajan et C. Hsu, «An Efficient Anonymous Mutual Authentication Technique for Providing Secure Communication in Mobile Cloud Computing for Smart City Applications,» *Sustainable Cities and Society,* vol. 49, n° 101522, 2019.

[72] A. Irshad, S. A. Chaudhry, M. Shafiq, M. Usman, M. Asif et A. Ghani, «A Provable and Secure Mobile User Authentication Scheme for Mobile Cloud Computing Services,» *Int J Commun Syst.,* 2019.

[73] O. O. Olakanmi et S. O. Oke, «MASHED: Security and Privacy-Aware Mutual Authentication Scheme for Heterogeneous and Distributed Mobile Cloud Computing Services,» *Information Security Journal: A Global Perspective,* vol. 27, n° 5-6, pp. 276-291, 2019.

[74] Y. Lu et D. Zhao, «Providing impersonation resistance for biometric-based authentication scheme in mobile cloud computing service,» *Computer Communications,* vol. 182, pp. 22-30, 2022.

[75] A. Irshad, M. Sher, H. F. Ahmad, B. A. Alzahrani, S. A. Chaudhry et R. Kumar, «An Improved Multi-Server Authentication Scheme for Distributed Mobile Cloud Computing Services,» *KSII Transactions on Internet and Information Systems,* vol. 10, n° 12, pp. 5529-5552, 2016.

[76] H. Jannati et B. Bahrak, «An Improved Authentication Protocol for Distributed Mobile Cloud Computing Services,» *International Journal of Critical Infrastructure Protection,* vol. 19, pp. 59-67, 2017.

[77] V. Odelu, A. K. Das, S. Kumari, X. Huang et M. Wazid, «Provably Secure Authenticated Key Agreement Scheme for Distributed Mobile Cloud Computing Services,» *Future Generation Computer Systems,* vol. 68, pp. 74-88, 2017.

[78] Q. Jiang, J. Ma et F. Wei, «On the Security of a Privacy-Aware Authentication Scheme for Distributed Mobile Cloud Computing Services,» *IEEE Systems Journal,* vol. 12, n° 2, pp. 2039-2042, 2018.

[79] P. Gope et A. K. Das, «Robust Anonymous Mutual Authentication Scheme for n-Times Ubiquitous Mobile Cloud Computing Services,» *IEEE Internet of Things Journal,* vol. 4, n° 5, pp. 1764-1772, 2017.

[80] S. Roy, S. Chatterjee, A. K. Das, S. Chattopadhyay, N. Kumar et A. V. Vasilakos, «On the Design of Provably Secure Lightweight Remote User Authentication Scheme for Mobile Cloud Computing Services,» *IEEE Access,* vol. 5, pp. 25808-25825, 2017.

[81] R. Vishwakarma et A. K. Jain, «A survey of DDoS attacking techniques and defence mechanisms in the IoT network,» *Telecommunication Systems,* vol. 73, pp. 3-25, 2020.

[82] S. K. Allam et G. S. Prasad, «A Survey on Distributed Denial of Service Attacks and Counter Measures,» *Test Engineering & Management,* vol. 82, pp. 16553-16560, 2020.

[83] M. M. Salim, S. Rathore et J. H. Park, «Distributed denial of service attacks and its defenses in IoT: a survey,» *The Journal of Supercomputing,* vol. 76, pp. 5320-5363, 2020.

[84] N. Agrawal et S. Tapaswi, «Defense Mechanisms Against DDoS Attacks in a Cloud Computing Environment: State-of-the-Art and Research Challenges,» *IEEE Communications Surveys & Tutorials,* vol. 21, n° 4, pp. 3769-3795, 2019.

[85] M. A. Alarqan, Z. F. Zaaba et A. Almomani, «Detection Mechanisms of DDoS Attack in Cloud Computing Environment: A Survey,» chez *M. Anbar et al. (Eds.): ACeS 2019, CCIS*, 2020.

[86] M. Nooribakhsh et M. Mollamotalebi, «A review on statistical approaches for anomaly detection in DDoS attacks,» *Information Security Journal: A Global Perspective,* vol. 29, n° 3, pp. 118-133, 2020.

[87] A. B. Badiru et O. A. Omitaomu, «Basic Mathematical Calculations,» chez *Handbook of Industrial Engineering Equations, Formulas, and Calculations*, Boca Raton, CRC Press, 2010.

[88] J. David et C. Thomas, «Detection of distributed denial of service attacks based on information theoretic approach in time series models,» *Journal of Information Security and Applications*, vol. 55, n° 102621, 2020.

[89] L. Zhou, K. Sood et Y. Xiang, «ERM: An Accurate Approach to Detect DDoS Attacks Using Entropy Rate Measurement,» *IEEE Communications Letters*, vol. 23, n° 10, pp. 1700-1703, 2019.

[90] «httperf - HTTP performance measurement tool,» [En ligne]. Available: https://manpages.ubuntu.com/manpages/focal/en/man1/httperf.1.html. [Accès le 4 juin 2021].

[91] «hping3,» [En ligne]. Available: https://manpages.ubuntu.com/manpages/focal/en/man8/hping3.8.html. [Accès le 4 juin 2021].

[92] B. A. Khalaf, S. A. Mostafa, A. Mustapha, M. A. Mohammed et W. M. Abduallah, «Comprehensive Review of Artificial Intelligence and Statistical Approaches in Distributed Denial of Service Attack and Defense Methods,» *IEEE Access*, vol. 7, pp. 51691-51713, 2019.

[93] Canadian Bankers Association, «Focus: How Canadians Bank,» 31 mars 2022. [En ligne]. Available: https://cba.ca/technology-and-banking. [Accès le 11 avril 2022].

[94] D. Wang et P. Wang, «Two Birds with One Stone: Two-Factor Authentication with Security Beyond Conventional Bound,» *IEEE Transactions on Dependable and Secure Computing*, vol. 15, n° 4, pp. 708-722, 2018.

[95] «OWASP Mobile Top 10,» OWASP, 2021. [En ligne]. Available: https://owasp.org/www-project-mobile-top-10/. [Accès le 22 septembre 2021].

[96] Google, «nogotofail (version 1.2.0),» [En ligne]. Available: https://github.com/google/nogotofail.

[97] H. Rongyu, Z. Guolei, C. Chaowen, X. Hui, Q. Xi et Q. Zheng, «A PK-SIM Card Based end-to-end Security Framework for SMS,» *Computer Standards & Interfaces,* vol. 31, n° 4, pp. 629-641, 2009.

[98] S. Qiu, G. Xu, H. Ahmad, G. Xu, X. Qiu et H. Xu, «An Improved Lightweight Two-Factor Authentication and Key Agreement Protocol with Dynamic Identity Based on Elliptic Curve Cryptography,» *KSII Transactions on Internet and Information Systems,* vol. 13, n° 2, pp. 978-1002, 2019.

[99] A. Kumari, S. Jangirala, M. Y. Abbasi, V. Kumar et M. Alam, «ESEAP: ECC Based Secure and Efficient Mutual Authentication Protocol using Smart Card,» *Journal of Information Security and Applications,* vol. 51, n° 102443, pp. 1-12, 2020.

[100] Q. Xie, D. S. Wong, G. Wang, X. Tan, K. Chen et L. Fang, «Provably Secure Dynamic ID-Based Anonymous Two-Factor Authenticated Key Exchange Protocol With Extended Security Model,» *IEEE Transactions on Information Forensics and Security,* vol. 12, n° 6, pp. 1382-1392, 2017.

[101] M. Luo, Y. Zhang, M. K. Khan et D. He, «A Secure and Efficient Identity-based Mutual Authentication Scheme with Smart Card using Elliptic Curve Cryptography,» *International Journal of Communication Systems,* vol. 30, n° 16, 2017.

[102] M. Qi et J. Chen, «An Efficient Two-Party Authentication Key Exchange Protocol for Mobile Environment,» *International Journal of Communication Systems,* vol. 30, n° 16, 2017.

[103] H. Sun, Q. Wen, H. Zhang et Z. Jin, «A Novel Remote User Authentication and Key Agreement Scheme for Mobile Client-Server Environment,» *Applied Mathematics & Information Sciences,* vol. 7, n° 4, pp. 1365-1374, 2013.

[104] M. S. Farash et M. A. Attari, «A Secure and Efficient Identity-based Authenticated Key Exchange Protocol for Mobile Client–Server Networks,» *The Journal of Supercomputing,* vol. 69, pp. 395-411, 2014.

[105] S. Qiu, D. Wang, G. Xu et S. Kumari, «Practical and Provably Secure Three-Factor Authentication Protocol Based on Extended Chaotic-Maps for Mobile Lightweight Devices,» *IEEE Transactions on Dependable and Secure Computing,* vol. 19, n° 2, pp. 1338-1351, 2022.

[106] D. Dolev et A. Yao, «On the security of public key protocols,» *IEEE Transactions on Information Theory,* vol. 29, n° 2, pp. 198-208, 1983.

[107] R. Canetti et H. Krawczyk, «Analysis of Key-Exchange Protocols and Their Use for Building Secure Channels,» *Pfitzmann B. (eds) Advances in Cryptology — EUROCRYPT 2001,* vol. 2045, 2001.

[108] O. Kupreev, E. Badovskaya et A. Gutnikov, «DDoS attacks in Q1 2020,» Kaspersky, 6 mai 2020. [En ligne]. Available: https://securelist.com/ddos-attacks-in-q1-2020/96837/. [Accès le 6 septembre 2020].

[109] S. Behal, K. Kumar et M. Sachdeva, «A generalized detection system to detect distributed denial of service attacks and flash events for information theory metrics,» *Turk J Elec Eng & Comp Sci,* vol. 26, pp. 1759-1770, 2018.

[110] I. Basicevic, N. Blazic et S. Ocovaj, «On the use of generalized entropy formulas in detection of denial-of-service attacks,» *Security and Privacy,* vol. 4, n° 1, 2021.

[111] I. Basicevic, N. Blazic et S. Ocovaj, «On the use of principal component analysis in the entropy based detection of denial-of-service attacks,» *Security and Privacy,* 2021.

[112] Z. Liu, C. Hu et C. Shan, «Riemannian manifold on stream data: Fourier transform and entropy-based DDoS attacks detection method,» *Computers & Security,* vol. 109, 2021.

[113] A. Gaurav, B. B. Gupta, C.-H. Hsu, S. Yamaguchi et K. T. Chui, «Fog Layer-based DDoS attack Detection Approach for Internet-of-Things (IoTs) devices,» chez *2021 IEEE International Conference on Consumer Electronics (ICCE),* Las Vegas, NV, USA, 2021.

[114] B. H. Ali, N. Sulaiman, S. A. R. Al-Haddad, R. Atan, S. L. M. Hassan et M. Alghrairi, «Identification of Distributed Denial of Services Anomalies by Using Combination of Entropy and Sequential Probabilities Ratio Test Methods,» *Sensors,* vol. 21, n° 19, 2021.

[115] J. David et C. Thomas, «Discriminating flash crowds from DDoS attacks using efficient thresholding algorithm,» *Journal of Parallel and Distributed Computing,* vol. 152, pp. 79-87, 2021.

[116] A. Gaurav, B. B. Gupta, C.-H. Hsu, D. Perakovic et F. J. G. Penalvo, «Filtering of Distributed Denial of Services (DDoS) Attacks in Cloud Computing Environment,» chez *2021 IEEE International Conference on Communications Workshops (ICC Workshops),* Montreal, QC, Canada, 2021.

[117] Z. Zhou, A. Gaurav, B. B. Gupta, H. Hamdi et N. Nedjah, «A statistical approach to secure health care services from DDoS attacks during COVID-19 pandemic,» *Neural Computing and Applications,* 2021.

[118] K. Singh, K. S. Dhindsa et D. Nehra, «T-CAD: A threshold based collaborative DDoS attack detection in multiple autonomous systems,» *Journal of Information Security and Applications,* vol. 51, n° 102457, 2020.

[119] J. Li, M. Liu, Z. Xue, X. Fan et X. He, «RTVD: A Real-Time Volumetric Detection Scheme for DDoS in the Internet of Things,» *IEEE Access,* vol. 8, pp. 36191-36201, 2020.

[120] R. Lippmann, «Proposed 1999 DARPA Off-line Intrusion Detection Evaluation Plans,» MIT Lincoln Laboratory, 28 janvier 1999. [En ligne]. Available: https://archive.ll.mit.edu/ideval/files/1999_NewPlans.PDF.

[121] M. Gharaibeh et C. Papadopoulos, «DARPA-2009 Intrusion Detection Dataset Report,» [En ligne]. Available: http://www.darpa2009.netsec.colostate.edu/DARPA_Set_Report.pdf.

[122] I. Sharafaldin, A. H. Lashkari, S. Hakak et A. A. Ghorbani, «Developing Realistic Distributed Denial of Service (DDoS) Attack Dataset and Taxonomy,» chez *2019 International Carnahan Conference on Security Technology (ICCST)*, Chennai, India, 2019.

[123] «DARPA Intrusion Detection Evaluation,» MIT Lincoln Laboratory, [En ligne]. Available: https://archive.ll.mit.edu/ideval/data/2000/LLS_DDOS_1.0.html.

[124] «The CAIDA Anonymized Internet Traces Dataset (April 2008 - January 2019),» CAIDA, 3 décembre 2019. [En ligne]. Available: https://www.caida.org/catalog/datasets/passive_dataset/. [Accès le 16 juin 2021].

[125] P. Bojovic, I. Bašicevic, S. Ocovaj et M. Popovic, «A practical approach to detection of distributed denial-of-service attacks using a hybrid detection method,» *Computers and Electrical Engineering,* vol. 73, pp. 84-96, 2019.

[126] S. Daneshgadeh, T. Kemmerich, T. Ahmed et N. Baykal, «An Empirical Investigation of DDoS and Flash Event Detection Using Shannon Entropy, KOAD and SVM Combined,» chez *2019 International Conference on Computing, Networking and Communications (ICNC)*, Honolulu, HI, USA, 2019.

[127] S. Daneshgadeh, T. Ahmed, T. Kemmerich et N. Baykal, «Detection of DDoS Attacks and Flash Events Using Shannon Entropy, KOAD and Mahalanobis Distance,» chez *2019 22nd*

Conference on Innovation in Clouds, Internet and Networks and Workshops (ICIN), Paris, France, 2019.

[128] M. Idhammad, K. Afdel et M. Belouch, «Detection System of HTTP DDoS Attacks in a Cloud Environment Based on Information Theoretic Entropy and Random Forest,» *Security and Communication Networks,* vol. 2018, 2018.

[129] M. Ring, S. Wunderlich, D. Grüdl, D. Landes et A. Hotho, «Flow-based benchmark data sets for intrusion detection,» chez *European Conference on Cyber Warfare and Security*, 2017.

[130] S. Behal, K. Kumar et M. Sachdeva, «D-FACE: An anomaly based distributed approach for early detection of DDoS attacks and flash events,» *Journal of Network and Computer Applications,* vol. 111, pp. 49-63, 2018.

[131] P. Kumar, M. Tripathi, N. Nehra, M. Conti et C. Lal, «SAFETY: Early Detection and Mitigation of TCP SYN Flood Utilizing Entropy in SDN,» *IEEE Transactions on Network and Service Management,* vol. 15, n° 4, pp. 1545-1559, 2018.

[132] S. Behal et K. Kumar, «Detection of DDoS attacks and flash events using novel information theory metrics,» *Computer Networks,* vol. 116, pp. 96-110, 2017.

[133] M. H. Bhuyan, D. K. Bhattacharyya et J. K. Kalita, «E-LDAT: a lightweight system for DDoS flooding attack detection and IP traceback using extended entropy metric,» *Security and Communication Networks,* vol. 9, n° 16, pp. 3251-3270, 2016.

[134] M. Sachdeva, K. Kumar et G. Singh, «A comprehensive approach to discriminate DDoS attacks from flash events,» *Journal of Information Security and Applications,* vol. 26, pp. 8-22, 2016.

[135] I. Basicevic, S. Ocovaj et M. Popovic, «Use of Tsallis entropy in detection of SYN flood DoS attacks,» *Security and Communication Networks,* vol. 8, n° 18, pp. 3634-3640, 2015.

[136] J. David et C. Thomas, «DDoS Attack Detection Using Fast Entropy Approach on Flow-Based Network Traffic,» *Procedia Computer Science*, vol. 50, pp. 30-36, 2015.

[137] H. Rahmani, N. Sahli et F. Kamoun, «Distributed denial-of-service attack detection scheme-based joint-entropy,» *Security and Communication Networks*, vol. 5, n° 9, pp. 1049-1061, 2012.

[138] Y. Xiang, K. Li et W. Yang, «Low-Rate DDoS Attacks Detection and Traceback by Using New Information Metrics,» *IEEE Transactions on Information Forensics and Security*, vol. 6, n° 2, pp. 426-437, 2011.

[139] T. M. Cover et J. A. Thomas, «Entropy, Relative Entropy, and Mutual Information,» chez *Elements of Information Theory, 2nd Edition*, Hoboken, John Wiley & Sons, Inc., 2006, pp. 13-55.

[140] C. Ngô, «Information et entropie,» chez *Énergie, Entropie, Information, Cryptographie et Cybersécurité*, EDP Sciences, 2019, pp. 87-102.

[141] H. Wang, C. Jin et K. G. Shin, «Defense Against Spoofed IP Traffic Using Hop-Count Filtering,» *IEEE/ACM Transactions on Networking*, vol. 15, n° 1, pp. 40-53, 2007.

[142] A. Mukaddam, I. Elhajj, A. Kayssi et A. Chehab, «IP Spoofing Detection Using Modified Hop Count,» chez *2014 IEEE 28th International Conference on Advanced Information Networking and Applications*, Victoria, BC, Canada, 2014.

[143] N. Hubballi et N. Tripathi, «An event based technique for detecting spoofed IP packets,» *Journal of Information Security and Applications*, vol. 35, pp. 32-43, 2017.

[144] «Apache HTTP Server Project,» Apache Software Foundation, [En ligne]. Available: https://httpd.apache.org/. [Accès le 4 juin 2021].

[145] J. Chen, «Mobile Banking,» Investopedia, 25 août 2020. [En ligne]. Available: https://www.investopedia.com/terms/m/mobile-banking.asp. [Accès le 5 juillet 2021].

[146] E. Fernando, Surjandy et Meyliana, «An Investigation Effective Factors Usage of Smartphone for Use Mobile Banking Services Case: Student University Customers,» chez *2020 International Conference on Information Management and Technology (ICIMTech)*, Bandung, Indonesia, 2020.

[147] K. F. Arisya, Y. Ruldeviyani, R. Prakoso et A. L. Fadhilah, «Measurement of Information Security Awareness Level: A Case Study of Mobile Banking (M-Banking) Users,» chez *2020 Fifth International Conference on Informatics and Computing (ICIC)*, Gorontalo, Indonesia, 2020.

[148] M. Boutin et M. Chouinard, «Cybersecurity in the Canadian Financial Sector as a National Economic Security Issue,» 13 mai 2019. [En ligne]. Available: https://www.ourcommons.ca/Content/Committee/421/SECU/Brief/BR10481660/br-external/InFidem-e.pdf. [Accès le 11 avril 2022].

[149] F. Laricchia, «Market share of mobile operating systems worldwide 2012-2022,» Statista, 7 février 2022. [En ligne]. Available: https://www.statista.com/statistics/272698/global-market-share-held-by-mobile-operating-systems-since-2009/. [Accès le 7 avril 2022].

[150] «About the OWASP Foundation,» OWASP, 2021. [En ligne]. Available: https://owasp.org/about/. [Accès le 29 septembre 2021].

[151] «App security best practices,» Google, 22 septembre 2021. [En ligne]. Available: https://developer.android.com/topic/security/best-practices. [Accès le 27 septembre 2021].

[152] S. Chen, T. Su, L. Fan, G. Meng, M. Xue, Y. Liu et L. Xu, «Are Mobile Banking Apps Secure? What Can Be Improved?,» chez *Proceedings of the 26th ACM Joint European Software Engineering Conference and Symposium on the Foundations of Software Engineering (ESEC/FSE '18)*, Lake Buena Vista, FL, USA, 2018.

[153] S. Bojjagani et V. Sastry, «VAPTAi: A Threat Model for Vulnerability Assessment and Penetration Testing of Android and iOS Mobile Banking Apps,» chez *2017 IEEE 3rd*

International Conference on Collaboration and Internet Computing (CIC), San Jose, CA, USA, 2017.

[154] X. Zheng, L. Pan et E. Yilmaz, «Security analysis of modern mission critical android mobile applications,» chez *ACSW '17: Proceedings of the Australasian Computer Science Week Multiconference*, Geelong, Australia, 2017.

[155] R. Chanajitt, W. Viriyasitavat et K.-K. R. Choo, «Forensic analysis and security assessment of Android m-banking apps,» *Australian Journal of Forensic Sciences*, vol. 50, n° 1, pp. 3-19, 2018.

[156] R. Kaur, Y. Li, J. Iqbal, H. Gonzalez et N. Stakhanova, «A Security Assessment of HCE-NFC Enabled E-Wallet Banking Android Apps,» chez *2018 IEEE 42nd Annual Computer Software and Applications Conference (COMPSAC)*, Tokyo, Japan, 2018.

[157] S. Bojjagani et V. Sastry, «STAMBA: Security Testing for Android Mobile Banking Apps,» chez *Thampi S., Bandyopadhyay S., Krishnan S., Li KC., Mosin S., Ma M. (eds) Advances in Signal Processing and Intelligent Recognition Systems. Advances in Intelligent Systems and Computing*, 2016.

[158] J.-H. Jung, J. Y. Kim, H.-C. Lee et J. H. Yi, «Repackaging Attack on Android Banking Applications and Its Countermeasures,» *Wireless Personal Communications*, vol. 73, n° 4, pp. 1421-1437, 2013.

[159] B. Reaves, J. Bowers, N. Scaife, A. Bates, A. Bhartiya, P. Traynor et K. R. B. Butler, «Mo(bile) Money, Mo(bile) Problems: Analysis of Branchless Banking Applications,» *ACM Transactions on Privacy and Security*, vol. 20, n° 3, pp. 1-31, 2017.

[160] D. Bassolé, G. Koala, Y. Traoré et O. Sié, «Vulnerability Analysis in Mobile Banking and Payment Applications on Android in African Countries,» chez *International Conference on Innovations and Interdisciplinary Solutions for Underserved Areas. InterSol 2020*, 2020.

[161] S. Castle, F. Pervaiz, G. Weld, F. Roesner et R. Anderson, «Let's Talk Money: Evaluating the Security Challenges of Mobile Money in the Developing World,» chez *ACM DEV '16: Proceedings of the 7th Annual Symposium on Computing for Development*, Nairobi, Kenya, 2016.

[162] T. Chothia, F. D. Garcia, C. Heppel et C. M. Stone, «Why Banker Bob (Still) Can't Get TLS Right: A Security Analysis of TLS in Leading UK Banking Apps,» chez *International Conference on Financial Cryptography and Data Security. FC 2017*, Sliema, Malta, 2017.

[163] H. Darvish et M. Husain, «Security Analysis of Mobile Money Applications on Android,» chez *2018 IEEE International Conference on Big Data (Big Data)*, Seattle, WA, USA, 2018.

[164] O. Osho, U. L. Mohammed, N. N. Nimzing, A. A. Uduimoh et S. Misra, «Forensic Analysis of Mobile Banking Apps,» chez *Computational Science and Its Applications – ICCSA 2019*, 2019.

[165] A. A. Uduimoh, O. Osho, I. Ismaila et S. M. Abdulhamid, «Forensic Analysis of Mobile Banking Applications In Nigeria,» *i-manager's Journal on Mobile Applications & Technologies,* vol. 6, n° 1, pp. 9-20, 2019.

[166] M. Egele, D. Brumley, Y. Fratantonio et K. Kruegel, «An empirical study of cryptographic misuse in android applications,» chez *CCS '13: Proceedings of the 2013 ACM SIGSAC conference on Computer & communications security*, Berlin, Allemagne, 2013.

[167] M. Zarifopoulos et A. A. Economides, «Evaluating Mobile Banking Portals,» *International Journal of Mobile Communications,* vol. 7, n° 1, pp. 66-90, 2009.

[168] «2021 CWE Top 25 Most Dangerous Software Weaknesses,» MITRE, 26 juillet 2021. [En ligne]. Available: https://cwe.mitre.org/top25/archive/2021/2021_cwe_top25.html. [Accès le 24 septembre 2021].

[169] C. Thompson, R. Leininger et R. Bhatt, «Mobile banking applications: security challenges for banks,» Accenture and NowSecure, Inc, 2017. [En ligne]. Available: https://www.accenture.com/_acnmedia/PDF-115/Accenture-Mobile-Banking-Apps-Security-Challenges-Banks.pdf. [Accès le 24 septembre 2021].

[170] M. Dworkin, «Recommendation for Block Cipher Modes of Operation Methods and Techniques,» NIST Special Publication 800-38A, Gaithersburg, 2001.

[171] M. S. Turan, E. Barker, W. Burr et L. Chen, «Recommendation for Password-Based Key Derivation Part 1: Storage Applications,» NIST Special Publication 800-132, 2010.

[172] «M5: Insufficient Cryptography,» OWASP, 2021. [En ligne]. Available: https://owasp.org/www-project-mobile-top-10/2016-risks/m5-insufficient-cryptography. [Accès le 11 octobre 2021].

[173] W. Zhijun, L. Wenjing, L. Liang et Y. Meng, «Low-Rate DoS Attacks, Detection, Defense, and Challenges: A Survey,» *IEEE Access*, vol. 8, pp. 43920-43943, 2020.

[174] S. Behal, K. Kumar et M. Sachdeva, «Discriminating Flash Events from DDoS Attacks: A Comprehensive Review,» *International Journal of Network Security*, vol. 19, n° 5, pp. 734-741, 2017.

[175] Oracle, «Java Card Platform Specification Release Notes, Version 3.1,» mars 2021. [En ligne]. Available: https://docs.oracle.com/en/java/javacard/3.1/specnotes/index.html. [Accès le 28 mai 2022].

ANNEXE A CLASSIFICATION DES PROTOCOLES D'AUTHENTIFICATION MUTUELLE DE LA LITTÉRATURE

Tableau A.1 Classification des protocoles d'authentification mutuelle de la littérature

Protocoles	**Techniques cryptographiques**	**Autorité de confiance**	**Matériel supplémentaire**	**Lieu de stockage du secret**	**Identité du client mobile**	**Identité du serveur**
[16]	Couplages	Oui	Oui	Carte à puce	Identifiant/mot de passe et empreinte digitale	Identifiant
[21]	Symétrique	Non	Non	N/A	Identifiant/mot de passe et OTP	OTP
[59]	Asymétrique	Non	Oui	Carte USIM	Signature numérique et second facteur (empreinte digitale, code PIN, etc.)	Signature numérique
[60]	Asymétrique	Oui	Non	N/A	Identifiant/mot de passe et clé privée	N/A
[61]	Asymétrique	Oui	Non	N/A	Identifiant/mot de passe et numéro de téléphone	Identifiant

Protocoles	Techniques cryptographiques	Autorité de confiance	Matériel supplémentaire	Lieu de stockage du secret	Identité du client mobile	Identité du serveur
[63]	Symétrique	Non	Non	N/A	Identifiant/mot de passe	N/A
[64] [65]	Symétrique	Non	Non	Périphérique mobile	Identifiant/mot de passe	Signature numérique
[66]	Symétrique	Non	Non	Périphérique mobile	Identifiant/mot de passe	Signature numérique
[67]	Courbes elliptiques	Non	Non	N/A	Identifiant	Identifiant
[68]	Courbes elliptiques	Non	Oui	Carte à puce	Identifiant/mot de passe	Identifiant
[69]	Courbes elliptiques	Non	Oui	Carte à puce	Identifiant/mot de passe	N/A
[70]	Courbes elliptiques	Oui	Non	Périphérique mobile	Identifiant/mot de passe	Identifiant
[71]	Couplages	Oui	Non	Périphérique mobile	Identifiant et empreinte digitale	Identifiant

Protocoles	Techniques cryptographiques	Autorité de confiance	Matériel supplémentaire	Lieu de stockage du secret	Identité du client mobile	Identité du serveur
[72]	Couplages	Oui	Oui	Carte à puce	Identifiant/mot de passe et biométrie	Identifiant
[73]	Couplages	Oui	Non	N/A	Identifiant/mot de passe et signature vocale	Identifiant
[74]	Couplages	Oui	Oui	Carte à puce	Identifiant/mot de passe et empreinte digitale	Identifiant
[79]	Fonctions de hachage	Non	Oui	Carte à puce	Identifiant/mot de passe	Identifiant
[80]	Fonctions de hachage	Oui	Non	Périphérique mobile	Identifiant/mot de passe et biométrie	Identifiant