Security and Trust in Mobile Cloud Computing

Thinh Le Vinh

le cnam

École doctorale Informatique, Télécommunications et Électronique (Paris)

Centre d'Études et de Recherche en Informatique et Communications

THÈSE DE DOCTORAT

présentée par : **Thinh LE VINH**

soutenue le : **14 décembre 2017**

pour obtenir le grade de : **Docteur du Conservatoire National des Arts et Métiers**

Spécialité : **Informatique**

Security and Trust in Mobile Cloud Computing

THÈSE dirigée par

Mme. SAMIA BOUZEFRANE	*Maître de Conférences/HDR, Conservatoire National des Arts et Métiers*

RAPPORTEURS

M. DAMIEN SAUVERON	*Maître de Conférences/HDR, Université de Limoges*
M. GONGXUAN ZHANG	*Professeur, Nanjing University of Science and Technology*

PRÉSIDENT

M. HOSSAM AFIFI	*Professeur, Telecom SudParis*

EXAMINATEURS

Mme. VÉRONIQUE LEGRAND	*Professeur, Conservatoire National des Arts et Métiers*
Mme. HANENE MAUPAS	*PhD, IDEMIA*

Abstract

As living in the cyber era, we admit that a dozen of new technologies have been born every day with the promises that making a human life be more comfortable, convenient and safe. In the forest of new technologies, mobile computing is raise as an essential part of human life. Normally, mobile devices have become the best companions in daily activities. They have served us from the simple activities like entertainment to the complicated one as business operations. As playing the important roles, mobile devices deserve to work in the environment which they can trust for serving us better. In this thesis, we investigate the way to secure mobile devices from the primitive security level (Trusted Platforms) to the sophisticated one (bio-inspired intelligence). More precisely, after addressing the challenges of mobile cloud computing (MCC), we have studied the real-case of mobile cloud computing, in terms of energy efficiency and performance, as well as proposed a demonstration of particular *MCC* model, called Droplock system. Moreover, taking advantages of trusted platform module functionality, we introduced a novel schema of remote attestation to secure mobile devices in the context of Mobile-Cloud based solution. To enhance the security level, we used fuzzy logic combining with ant colony system to assess the trust and reputation for securing another mobile cloud computing model based on the cloudlet notion.

Keywords : trusted computing, mobile cloud computing, attestation, energy efficiency, fuzzy logic, ant colony system.

Résumé

Nous vivons aujourd'hui dans l'ère cybernétique où de nouvelles technologies naissent chaque jour avec la promesse de rendre la vie humaine plus confortable, pratique et sûre. Parmi ces nouvelles technologies, l'informatique mobile se développe en influençant la vie de l'utilisateur. En effet, les plates-formes mobiles (smartphone, tablette) sont devenues les meilleurs compagnons de l'utilisateur pour mener à bien ses activités quotidiennes, comme les activités commerciales ou de divertissement. En jouant ces rôles importants, les plates-formes mobiles doivent opérer dans des environnements de confiance. Dans cette thèse, nous étudions la sécurité des plates-formes mobiles en passant d'un niveau de sécurité primitif qui s'appuie sur les plates-formes de confiance, à un niveau plus sophistiqué qui se base sur de l'intelligence bio-inspirée. Plus précisément, après avoir abordé les défis du cloud computing mobile (MCC), nous avons développé une étude de cas appelée Droplock pour le cloud mobile et nous avons étudié son efficacité énergétique et ses performances pour illustrer le modèle MCC. En outre, en s'appuyant sur les plates-formes de confiance (comme les TPM), nous avons introduit un nouveau schéma d'attestation à distance pour sécuriser les plates-formes mobiles dans le contexte du cloud mobile. Pour améliorer le niveau de sécurité et être adaptatif au contexte, nous avons utilisé de la logique floue combinant un système de colonies de fourmis pour évaluer la confiance et la réputation du cloud mobile basé sur la notion de cloudlets.

Mots clés : informatique de confiance, cloud mobile, attestation, efficacité énergétique, logique floue, algorithme de fourmis.

Acknowledgements

It is my deep thankfulness to all the people without whom this thesis could not be possible.

I would like to express my sincere gratitude to my supervisor, Associate-Professor/HDR Mrs. Samia Bouzefrane, for her constant supports, guidance and motivation. I take also this opportunity to thank the staff members in the Department of Computer Science, Conservatoire National des Arts et Métiers (CNAM), Paris for assisting me in all administrative details regarding my thesis. I would also like to thank the Ministry of Education, Vietnam, Education University of Ho Chi Minh City for sponsoring my research scholarship in Paris.

In addition, I would like to thank Dr. Soumya Banerjee, Mr. Youcef Ould Yahia and Mr. Aghiles Adjaz for the helpful discussions we had together and the advice to fulfill this project.

I am also grateful for the support of Uncle Tuan who has supported and given me a lot of advice on many stubborn problems during the time I am in Paris.

I would also like to deeply thank my parent, my parents in law, my grandparents, my grandparents in law, my older brother and sister, my younger brother in law and the rest of my family member for their endless support to all my endeavors.

Finally, I would give my deepest appreciation to my wife and my 2 little boys for their love, encouragement and sacrifice without which I could not have strengths to overcome all the difficulties during my overseas study time.

Contents

List of Tables

List of Figures

Résumé de la thèse

Les appareils mobiles, comme nous le savons aujourd'hui, n'existent pas depuis longtemps. Ce n'est que le 3 avril 1973 que Martin Cooper, ingénieur principal chez Motorola, a fait le premier appel téléphonique au monde à sa compagnie de télécommunication en leur disant qu'il parlait via un téléphone portable. Depuis lors, l'appareil mobile a énormément évolué, de l'usage personnel à l'entreprise. Avec la prolifération rapide des appareils mobiles, nous sommes conscients que chaque année une grande variété d'appareils avec différentes propriétés, des capacités améliorées, et plus d'intelligence sont disponibles sur le marché. Selon la prévision de Cisco, le nombre d'appareils mobiles, y compris les modules machine-to-machine (M2M), atteindra 1,5 par habitant en 2021, soit près de 12 milliards d'appareils connectés, contre 8 milliards et 1,1 par habitant en 2016. En outre, le trafic par appareil d'utilisateur mobile atteindra globalement 5657 mégaoctets par mois d'ici 2021, contre 977 mégaoctets par mois en 2016, soit un taux de croissance annuel composé de 42%. Les appareils mobiles vont de plus en plus faire face à de lourdes tâches de calcul car l'importance de l'énorme taille des données sera de plus en plus manipulée. En conséquence, les appareils mobiles remplaceront les PC en tant qu'appareils d'accès les plus courants pour des usages multiples.

Avec l'émergence des nouvelles technologies de l'information et de la communication, les appareils intelligents combinés à des infrastructures de communication telles qu'Internet et les réseaux mobiles doivent fournir des services fiables. à cette fin, de nombreux domaines d'application à fort impact social et commercial, tels que les soins de santé personnels, la domotique, le paiement mobile, peuvent utiliser des appareils mobiles qui s'appuient sur un environnement de confiance. Cet environnement de confiance peut être fourni ou basé sur de nombreuses plates-formes matérielles ou logicielles telles que des cartes SIM, des

éléments sécurisés pour la communication en champ proche (NFC), l'émulation de carte hôte (HCE), TEE (Trusted Execution Environment) ou TPM. Même si ces plateformes fiables sont extrêmement limitées en termes d'énergie, de puissance de calcul et de mémoire, elles sont conçues pour résoudre les principales contraintes de sécurité.

D'un autre côté, le cloud computing doit rechercher comment gérer le stockage à distance et les ressources partagées par des multi-utilisateurs dans un environnement virtualisé et isolé [5]. Pour être plus précis, le cloud computing fournit les derniers services; à savoir le réseau en tant que service (NaaS), l'infrastructure en tant que service (IaaS), la plateforme en tant que service (PaaS), le logiciel en tant que service (SaaS) et le stockage en tant que service (STaaS) pour abstraire tous les types de ressources informatiques en tant que services. [6]. Par conséquent, les services d'infrastructure de cloud partagé fonctionnent comme des utilitaires: les utilisateurs disposant de leur téléphone intelligent peuvent accéder aux services dont ils ont besoin à partir de ressources hébergées à distance sur Internet et payer pour ces services. Ces services ne se mettent pas seulement à niveau automatiquement, ils évoluent également facilement vers le haut ou vers le bas. On peut voir clairement que les avantages du cloud computing ont apporté des avantages aux appareils mobiles pour surmonter les limites de la technologie mobile; à savoir le stockage de données limité, la puissance de traitement et la durée de vie de la batterie. Généralement, le terme de Mobile Cloud Computing (*MCC*) vient de la combinaison des technologies de l'informatique mobile et du cloud computing.

Dans cette thèse, après avoir étudié le cas réel des plateformes de confiance et des modèles de *MCC*, nous nous concentrons principalement sur le défi de la sécurité dans l'écosystème du cloud mobile avec le soutien de plateformes de confiance en proposant notre intérêt sur l'authentification, la confiance et la réputation dans le contexte du cloud computing mobile.

Description du problème: Calcul sécurisé pour le Cloud Computing Mobile

Dans le cloud computing mobile, la tendance actuelle est d'externaliser les données et le calcul des appareils mobiles vers le cloud afin d'étendre les limites de travail des

équipements à ressources limitées et de bénéficier de capacités cloud illimitées telles que le stockage. En outre, *MCC* doit offrir des mécanismes pour garantir la protection et la sécurité des données dans un environnement non sécurisé. Avec l'utilisation croissante des *MCC*, les problèmes de sécurité restent un défi. Notre travail consiste à rechercher principalement sur les plateformes de confiance existantes pour répondre aux questions suivantes:

- Qu'est-ce qu'un environnement de confiance ?
- Une solution de sécurité pure est-elle suffisante pour créer un environnement de confiance ? et
- Comment les appareils mobiles peuvent-ils coopérer avec le Cloud pour créer un environnement de confiance ?

En outre, en prenant en charge le matériel pour fournir des primitives de sécurité indépendantes des autres composants du système, le concept d'informatique de confiance n'est pas une notion nouvelle et est décrit dans de nombreuses recherches [7; 8; 9; 10]. Un exemple spécifique de technologie informatique de confiance est l'attestation à distance (Remote Attestation) qui permet à un système informatique de mesurer les propriétés d'un système distant de telle sorte qu'un système distant sera détecté s'il est compromis. Les propriétés d'une plate-forme ou d'une application peuvent être utilisées pour définir les exigences de sécurité afin de satisfaire le terme de base C.I.A (Confidentialité, Intégrité, Authentification) de la sécurité du système. Dans l'environnement Cloud, l'un des problèmes pouvant survenir est la gestion de la clé de chiffrement. Plusieurs clients du Cloud peuvent avoir leur propre méthode de cryptage et la gestion des clés de sécurité est un autre problème à résoudre dans le contexte des données cryptées [11]. En prenant en charge la protection des clés cryptographiques, la génération de nombres aléatoires, la liaison cryptographique des données à certaines configurations système, l'encapsulation des données dans la configuration de l'application et l'authentification de la plateforme/application, nous proposons d'utiliser Trusted Platform Module (TPM), supporté par le Trusted Computing Group, pour surmonter cette situation. Le module TPM (Trusted Platform Module), un exemple spécifique de plates-formes de confiance, a été spécifiquement conçu pour

constituer un bloc de construction pour l'informatique de confiance. C'est une utilisation importante dans l'industrie et le gouvernement, par exemple: Bitlocker pour le cryptage de périphérique de Microsoft et les solutions de sécurité pour ordinateur portable dans le département de la défense des États-Unis [12]. Avec ses fonctionnalités, le TPM protège le système informatique au-delà du contrôle de l'utilisateur contre une attaque délibérée ou accidentelle. En considérant les fonctionnalités du TPM, nous proposons des suggestions sur un nouveau schéma d'attestation basée sur la propriété, utilisable avec l'authentification par jeton.
Depuis, la sécurité et la confiance sont des aspects complémentaires nécessaires pour construire une solution complète et adaptative. Par conséquent, l'intelligence basée sur la confiance et la réputation, comme la logique floue et le système de colonie de fourmis, est considérée ici dans le contexte de l'environnement de cloud mobile.

Contribution de la thèse

Contribution 1

La thèse commence par l'étude de cas réelle de Cloud Computing Mobile. Nous décrivons d'abord les différents modèles d'architecture de *MCC* et nous illustrons ces architectures en examinant leur efficacité énergétique avec le travail expérimental [13]. Cette approche est compétente pour adapter les deux contributions suivantes lorsque nous analyserons la performance des différentes approches. Ensuite, pour comprendre l'application du modèle *MCC* avec le niveau de sécurité standard de base dans un cas réel, nous proposons un système Droplock [14]. L'objectif actuel de l'approche Droplock est d'améliorer la sécurité de la prestation de services du fournisseur au client. Ce travail nous a inspiré pour développer une nouvelle approche de sécurité dans la deuxième contribution.

Contribution 2

En tirant parti de la combinaison de technologies et de tendances, telles que des plateformes de confiance, le cloud computing et l'apport de votre propre appareil, nous introduisons l'attestation de jeton basée sur la propriété (Property-based Token Attestation)

pour sécuriser l'utilisateur mobile dans l'environnement cloud d'entreprise. Bien que l'architecture de *MCC* puisse être classifiée en plusieurs catégories [15], pour simplifier, nous avons sélectionné le modèle Client-Serveur de *MCC* pour simuler la solution de sécurité proposée.

Contribution 3

En considérant le protocole de sécurité traditionnel (infrastructure à clé publique (PKI), etc.) comme insuffisant pour sécuriser fermement l'écosystème de *MCC*, nous proposons un schéma d'évaluation de l'intelligence bio-inspirée fondée sur la confiance et la réputation (e.g. logique floue et système de colonies). En tant qu'addition de la deuxième contribution, le schéma proposé convient à une relation un-à -un (Client-Serveur) ainsi qu'à une relation un-à -plusieurs (Client-Cloudlets) oà les clients peuvent sélectionner dynamiquement les bons fournisseurs de services pour adapter leurs propres exigences.

Plan de la thèse

Le reste de ce manuscrit est organisé comme suit :

♢ Le chapitre 2 rappelle plusieurs techniques de virtualisation et compare la virtualisation basée sur des conteneurs avec les machines virtuelles.

♢ Le chapitre 3 introduit diverses définitions du cloud computing mobile, y compris ses structures et les modèles associés.

♢ Le chapitre 4 fournit un aperçu des plates-formes de confiance existantes et détermine leur pertinence pour les utilisateurs mobiles sécurisés dans le contexte du cloud computing mobile.

♢ Le chapitre 5 aborde les problèmes de sécurité liés au cloud computing mobile.

♢ Le chapitre 6 décrit un scénario pour mesurer la confiance et la réputation, qui consiste également en l'état de l'art de la littérature correspondante liée à l'intelligence computationnelle.

♢ Le chapitre 7 présente la première contribution en abordant les défis récents de *MCC* en termes d'énergie et de sécurité dans le cadre d'études de cas et de travaux expérimentaux réels.

♢ Le chapitre 8 expose la seconde contribution en introduisant un nouveau cadre pour l'attestation, à savoir l'attestation de jeton basée sur la propriété, dans *MCC*. La performance, y compris la consommation d'énergie et la sécurité, est également analysée ici pour faire une comparaison entre le modèle actuel et les modèles connexes.

♢ Le chapitre 9 fournit la dernière contribution du travail de thèse pour la gestion de la confiance et de la réputation dans le contexte des multi-cloudlets.

♢ Le chapitre 10 analyse la performance entre les modèles proposés et les autres modèles contemporains

♢ Le chapitre 11 conclut et résume l'ensemble du travail et présente également la perspective première de ce travail.

Avec l'expréssion «Pas de Cloud sans virtualisation», la technologie de virtualisation est largement utilisée pour partager les capacités des ordinateurs physiques en répartissant les ressources entre les systèmes d'exploitation. En 1964, le concept de machines virtuelles (VM) est utilisé dans le projet d'IBM appelé système CP/CMS. Le programme de contrôle (CP) a agi pour diviser la machine physique en plusieurs copies (VM), chaque copie ayant son propre système d'exploitation. En 1974, Goldberg et Popek ont publié un document [16] qui introduit un ensemble de conditions suffisantes pour que l'architecture informatique supporte efficacement la virtualisation du système. En 1999, la société VMware a présenté son produit VMware Virtual Platform pour l'architecture x86-32. En 2006, Intel et AMD ont proposé des extensions pour prendre en charge la virtualisation.

Les techniques de virtualization

La couche de virtualisation appelée hyperviseur ou VMM (Virtual Machine Monitor) gère les ressources matérielles entre les différentes machines virtuelles. Le système d'exploitation

invité est le système d'exploitation qui s'exécute dans une machine virtuelle. Actuellement, il existe plusieurs techniques de virtualisation que nous résumons ci-dessous.

Virtualisation complète

Dans la virtualisation complète, le code source du système d'exploitation invité n'est pas modifié. Ainsi, lorsqu'il est exécuté, le système d'exploitation invité n'est pas conscient d'être virtualisé. La traduction binaire (BT) est utilisée pour émuler un petit ensemble d'instructions, c'est-à -dire que des instructions privilégiées sont capturées et sensibles mais des instructions non privilégiées sont détectées puis traduites pour montrer une fausse valeur. Le reste des instructions est directement exécuté par le CPU hôte. Il existe deux types d'hyperviseurs qui gèrent la virtualisation complète. Le type 1 correspond à un hyperviseur natif ou autonome qui s'exécute directement sur le matériel. Type 2 est un hyperviseur qui s'exécute sur le système d'exploitation hôte (comme une application ordinaire). L'inconvénient de la virtualisation complète est que le VMM doit utiliser des techniques spéciales pour virtualiser le matériel pour chaque machine virtuelle, ce qui génère un surcoût supplémentaire lors de l'accès au matériel [17].

Para-virtualisation

La para-virtualisation fait référence à la collaboration entre le système d'exploitation invité et l'hyperviseur pour améliorer les performances. Cette collaboration implique de modifier le code source de système d'exploitation invité pour appeler directement, en utilisant des hypercalls, l'hyperviseur pour exécuter des instructions privilégiées. Le système d'exploitation invité est donc conscient d'être virtualisé. L'inconvénient de cette technique est la mauvaise compatibilité et la portabilité des systèmes d'exploitation.

Virtualisation assistée par matériel

Pour simplifier les techniques de virtualisation, les fournisseurs de matériel tels que Intel et AMD ont introduit de nouvelles extensions pour prendre en charge la virtualisation. Ainsi, deux fonctionnalités en mode CPU sont introduites dans Intel Virtualization Technology (VT-x) et AMDV d'AMD pour que le VMM fonctionne en mode root sous le ring 0 (c.-à -d.

Ring 1) pendant que le SE invité fonctionne en mode non-root (ring 0). L'état du système d'exploitation invité est stocké dans des structures de contrôle de machine virtuelle (pour Intel VT-x) ou des blocs de contrôle de machine virtuelle (pour AMDV). L'avantage de la virtualisation assistée par matériel est la réduction du surcharge provoquée par le modèle trap-and-emulate [18].

Virtualisation du matériel

Pour exécuter avec succès les systèmes virtualisés, la couche de virtualisation (VMM) doit partager le processeur, la mémoire et les périphériques. Plusieurs mécanismes sont mis en œuvre pour le faire.

Processeur

Généralement, le système d'exploitation est conçu pour s'exécuter directement sur la machine physique et utiliser pleinement les ressources de l'ordinateur. L'architecture x86 traditionnelle comporte quatre niveaux de privilèges (appelés anneaux 0 à 3) dédiés au système d'exploitation et aux applications afin de gérer l'accès au matériel. Les applications utilisateur s'exécutent dans l'anneau 3 (mode utilisateur), tandis que le système d'exploitation s'exécute dans l'anneau 0 (mode noyau) pour avoir tous les privilèges. Dans les nouvelles générations de processeurs, une extension pour la virtualisation appelée Virtual Machine Extension (VMX) a été ajoutée. Dans l'architecture Intel VT-X, il existe deux niveaux d'exécution: le mode root VMX utilisé pour exécuter les fonctions VMM; et le mode non-root VMX qui correspond à un privilège réduit pour exécuter les systèmes d'exploitation invités.

Memoire

Dans les systèmes d'exploitation traditionnels, une mémoire physique est allouée à chaque processus défini avec son propre espace d'adressage virtuel. Lorsque le processus veut accéder à la mémoire, l'adresse virtuelle du processus est traduite par l'unité de gestion de la mémoire (MMU) qui repose sur la table de pages de processus. Lorsqu'un système d'exploitation invité s'exécute sur un hyperviseur, un niveau de traduction supplémentaire

est ajouté. En fait, le système d'exploitation invité mappe des adresses virtuelles invitées (GVA) avec des adresses physiques invitées (GPA), et la traduction de GPA en adresses physiques de machine (MPA) est effectuée par l'hyperviseur [19].

Périphériques

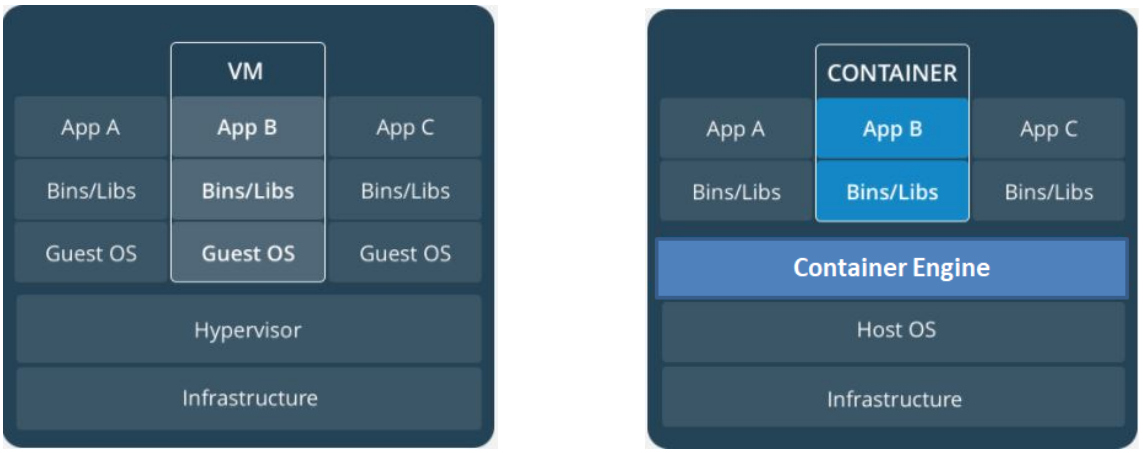
Le VMM doit présenter des périphériques au système d'exploitation invité, tels qu'un disque, une interface réseau, un port USB et des minuteurs. Puisque les pilotes du système d'exploitation invité ne sont pas conscients d'être virtualisés, toutes les instructions générées par les pilotes doivent être capturées et émulées, ce qui est la source de surcharge. Cependant, si les pilotes sont conscients d'être virtualisés, un chemin plus direct pour gérer les périphériques à l'intérieur de la VM peut être utilisé en installant des pilotes particuliers qui intéragissent avec les périphériques para-virtuels (comme l'interface virtuelle sous Linux) [20].

Isolation par conteneurs

Dans les solutions basées sur la virtualisation, la planification de l'hyperviseur fonctionne à deux niveaux, comme expliqué ci-dessous. La première planification se produit au niveau de l'hyperviseur où les machines virtuelles sont planifiées et le second au niveau de la machine virtuelle où le système d'exploitation invité planifie les processus. En prenant cela en compte, il est évident que cette approche crée un surcoût significatif. Une autre approche de virtualisation, basée sur des conteneurs, a été proposée pour réduire significativement ce surcoût, et ainsi fournir de meilleures performances, notamment en termes d'élasticité et de densité au sein d'un data center faible.

Conteneurs

Dans la virtualisation basée sur conteneur, la planification se produit uniquement à un niveau. Les conteneurs contiennent des processus, tandis que le noyau et les bibliothèques sont partagés entre les conteneurs. Par conséquent, le planificateur est utilisé uniquement pour planifier les processus. Le comportement est similaire à un système d'exploitation classique; la seule différence est que, dans la virtualisation par conteneur, les processus sont



(a) Hyperviseur basé sur des machines virtuelles

(b) Virtualisation basée sur les conteneurs

Figure 1: Architecture de la machine virtuelle par rapport aux architectures Docker Container

affectés à un conteneur, et entre chaque conteneur, une isolation est fournie par le système d'exploitation.

Nous pouvons définir un conteneur comme un environnement confiné dans l'environnement global. Le système d'exploitation Linux fournit certains mécanismes pour créer ces environnements ou conteneurs confinés, et oblige chaque conteneur à suivre une politique prédéfinie. L'isolation peut se produire à deux niveaux: l'isolation entre les processus et l'isolation entre les processus et le matériel [21].

Virtualisation basée sur les conteneurs par rapport aux machines virtuelles

Comme le montre la Figure 1, contrairement aux machines virtuelles qui dé pendent d'un hyperviseur, les conteneurs sont considérés comme des applications dans le système d'exploitation hôte. Par consé quent, comme indiqué dans [22], les performances offertes par l'utilisation de conteneurs sont meilleures que celles des machines virtuelles, ce qui est assez é vident puisque les conteneurs n'entraînent pratiquement aucune surcharge par rapport aux machines virtuelles. La table 1 présente la comparaison entre les conteneurs et les machines virtuelles en termes de taille des images, de dé pendance à la plate-forme et de migration.

	Containers	VMs
Require	A host OS	A hypervisor of type 1 or type 2
Size of images	Images are lightweight (for example, basic Ubuntu image for Docker containers has 180MB)	A VM hosts a guest OS that is a very big file (for example Ubuntu 14.04.1 has more than 1GB).
Platform dependency	Dependent-platform: the container is viewed as a bunch of processes vis-à -vis the host OS.	Independent-platform: a guest OS is hosted by a VM that is managed by a hypervisor. A guest OS is not aware of that and behaves as an OS that has total access to the physical resources.
Migration	Migration is possible only if the source and the destination platforms are Linux based. Even if Docker container engine is provided for Windows platform for example, this engine uses Linux-specific kernel features and requires a VM.	Migration is suitable for VMs viewed as files easily migratable and manageable by a hypervisor.

Table 1: Difference between Containers and VMs

Au cours des dernières années, le Cloud Computing est un domaine émergent qui affecte l'infrastructure informatique, les services réseau et les applications en fournissant un cadre tel que la population d'Amazon Web Service, de Window Azure et de Google App Engine. Le cloud computing est un élément tiers avec des serveurs de stockage à grande échelle et des centres de données utilisés pour fournir des infrastructures, des plates-formes de développement de logiciels et de distribution à faible coût dans la technologie informatique. L'avantage du Cloud Computing est venu avec trois grands modèles populaires qui sont offerts aujourd'hui en tant que services. La Figure 2 représente les modèles de service populaires dans l'environnement Cloud.

Logiciel en tant que service (SaaS)

SaaS ou logiciel à la demande est un modèle de distribution de logiciels permettant aux clients d'accéder à des logiciels spécifiques sur Internet tels que Google App, Office 360. En mode SaaS, le fournisseur est connu sous le nom de fournisseur SaaS hébergeant l'application

dans ses centres de calcul et l'interface de l'application est généralement accessible via un navigateur Web. SaaS offre ses avantages à un usage privé ou d'entreprise. Par exemple, au lieu d'acheter la licence de l'application, l'utilisateur final peut obtenir les mêmes fonctions qu'un service hébergé fourni par des serveurs distants. Puisque l'application est gérée à distance et de manière centralisée par les fournisseurs, la complexité de l'installation, de la maintenance et de la mise à niveau du logiciel a été réduite sans intervention de l'utilisateur.

Plate-forme en tant que service (PaaS)

Alors que SaaS fournit les fonctions fixes, PaaS offre des interfaces de programmation d'applications (API) et un environnement de programmation sur lesquels les développeurs d'applications peuvent créer leurs propres applications et les héberger sur l'infrastructure du fournisseur PaaS. En fournissant des serveurs virtualisés et des services associés, le service PaaS permet à un utilisateur d'exécuter des applications existantes, ou de concevoir, développer, tester, déployer, déboguer et héberger des applications sur la plate-forme logicielle comme cadre de développement. En outre, l'élasticité et l'évolutivité de la plate-forme matérielle et logicielle sous-jacente sont garanties de manière transparente par la plate-forme PaaS pour l'application en cours d'exécution. Le fournisseur PaaS est responsable de la surveillance de la livraison des applications. Google Application Engine (GAE), Amazon Web Services et Microsoft Azure sont des exemples particuliers de PaaS. Le prix pour PaaS peut se calculer sur une licence de développeur par application et sur une base de sièges hébergés [23]. Dans la hiérarchie des modèles, PaaS a un plus grand contrôle de l'utilisateur que SaaS.

Infrastructure en tant que service (IaaS)

En dessous de PaaS, on trouve l'IaaS qui fournit les facilités de communication et de stockage à travers la virtualisation. En offrant l'infrastructure virtualisée comprenant le matériel, le stockage, les serveurs, le centre de données et les composants réseau, les clients peuvent installer leur propre système d'exploitation, base de données et services informatiques sur les machines virtuelles provisionnées [23]. Au lieu d'acheter leurs propres

composants de matériel informatique, les clients achètent généralement des IaaS en fonction d'une base de calcul de l'utilisation ou de l'utilité, comme le temps CPU, l'espace de stockage et la bande passante du réseau. Amazon EC2, Cisco Metapod ou Google Compute Engine sont quelques exemples populaires du modèle IaaS.

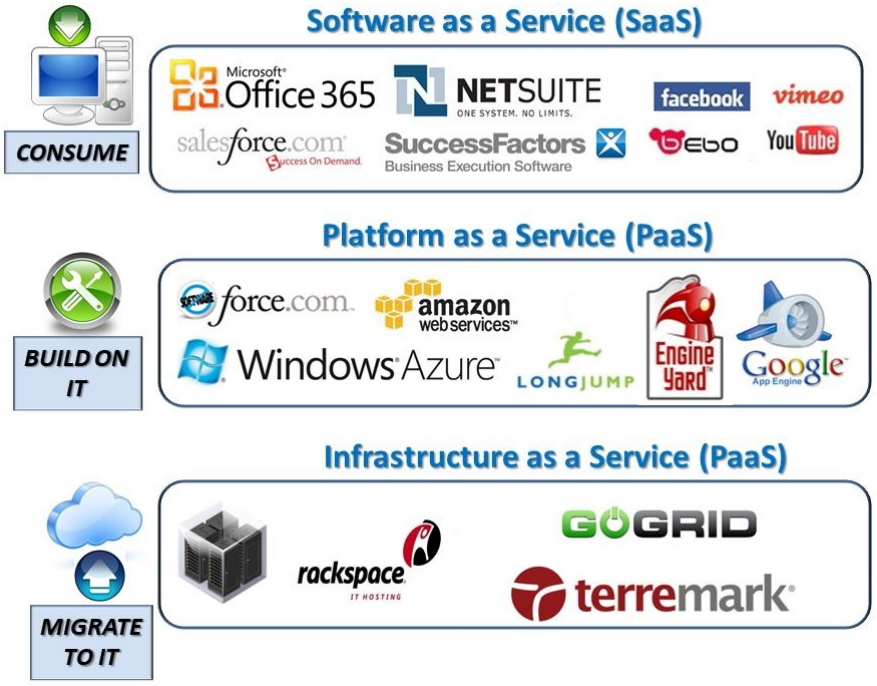


Figure 2: Modèles de service du cloud computing

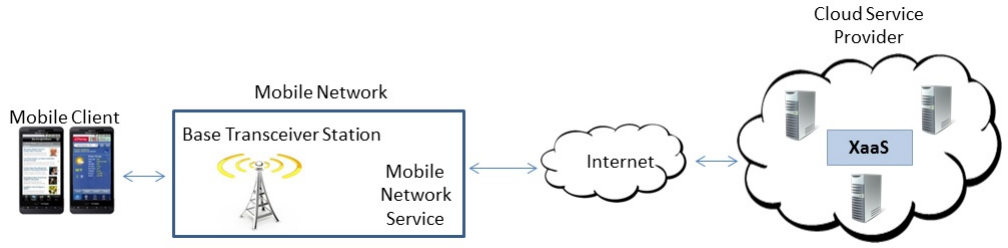
Cloud Computing Mobile

Parallèlement à cette évolution, on suppose que les appareils mobiles hétérogènes ou les objets en réseau (depuis les smartphones, les ordinateurs portables et les dispositifs portables jusqu'aux objets intégrés) ont la capacité de partager des données [24]; Des études pointent vers l'existence de plus de 50 milliards d'objets d'ici 2020 [25]. L'hétérogénéité vient du point de vue du logiciel, du matériel et de l'architecture, comme expliqué dans [26]. De nos jours, l'informatique mobile est une partie essentielle de la vie humaine qui rend la vie quotidienne plus commode et efficace indépendamment du temps et de l'endroit. En offrant de manière souple des services à la demande aux utilisateurs, les applications mobiles

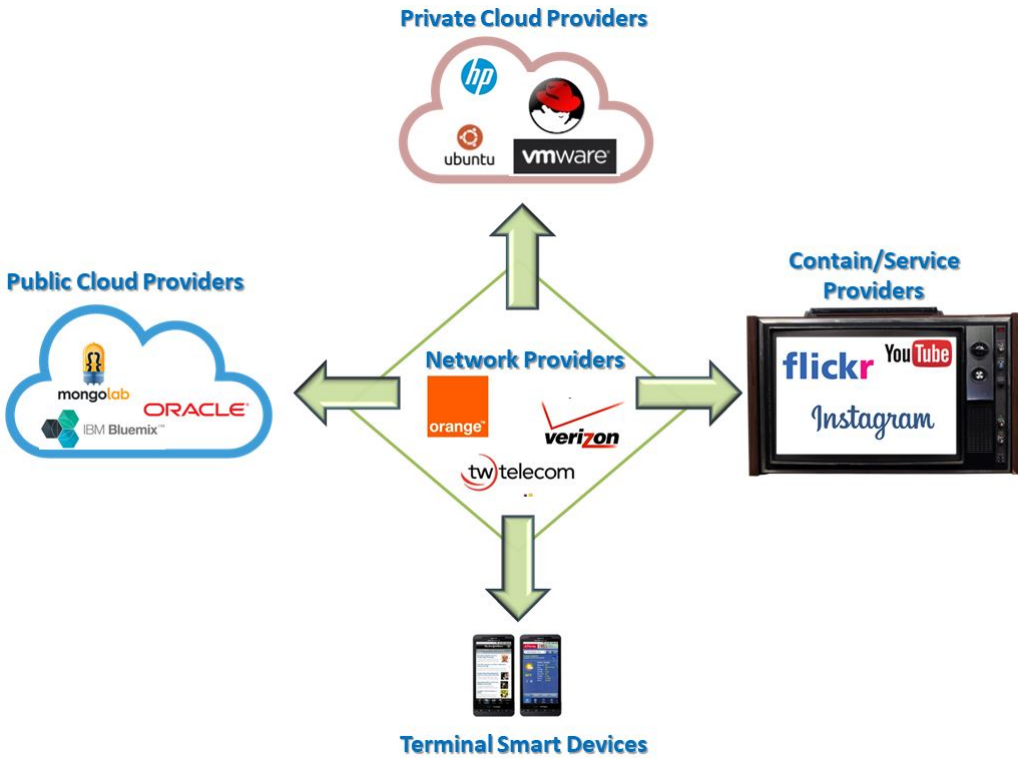
basées sur le Cloud Computing peuvent être rapidement provisionnées et libérées avec un minimum d'efforts de la part des fournisseurs de services et de la gestion [24]. Cependant, le Cloud Computing Mobile ne peut pas être simplement défini comme une fusion entre les technologies mobiles et le Cloud Computing. Il existe plusieurs définitions du Cloud Computing Mobile qui doivent être prises en compte avant d'examiner ses modèles et les problèmes de sécurité restants. Selon IBM, "Le Cloud Computing Mobile est une nouvelle plateforme combinant les appareils mobiles et le cloud computing pour créer une nouvelle infrastructure, par laquelle le Cloud effectue la partie lourde des tâches informatiques et stock des quantités massives de données. Dans cette nouvelle architecture, le traitement et le stockage des données se font en dehors des appareils mobiles". De même, Cisco Internet Business Solutions Group (IBSG) [27] définit le Cloud Mobile comme "des services mobiles et des applications livrés depuis un centre de données centralisé (et peut-être virtualisé) vers un appareil mobile, tel qu'un smartphone. Les clients accèdent à ces services à la demande en utilisant un navigateur ou un client léger sur leurs appareils mobiles. Ce qui contraste avec les clients "plus gros" qui sont téléchargés et résident (et s'exécutent) sur l'appareil mobile. Les services de Cloud mobile sont agnostiques au type d'appareil sur lequel ils s'exécutent". Une autre définition de Techopedia estime que : "Le cloud computing mobile est une technique ou un modèle dans lequel les applications mobiles sont construites, alimentées et hébérgées utilisant les technologies du Cloud Computing. Une approche cloud mobile permet aux développeurs de créer des applications conçues spécifiquement pour les utilisateurs mobiles sans être liés par le système d'exploitation mobile et la capacité de calcul ou de mémoire du smartphone. Le cloud computing mobile est généralement accessible via un navigateur mobile depuis un serveur web distant, typiquement sans avoir besoin d'installer une application client sur le téléphone destinataire". L'architecture MCC et ses partenaires importants sont représentés dans les figures 3a et 3b respectivement.

Les modèles MCC

Pour souligner la motivation derrière le Cloud Computing Mobile, différentes architectures dans la littérature [28; 29; 30; 31] ont été définies afin de répondre à différents cas d'utilisation dans le cloud computing mobile. Le principal avantage du Cloud Comput-



(a)



(b)

Figure 3: L'architecture MCC et ses parties prenantes importantes

ing pour les périphériques mobiles est de permettre l'exécution d'applications entre des périphériques à ressources limitées et le Cloud Internet. Par conséquent, les périphériques à ressources limitées peuvent externaliser sur le Cloud les opérations couteuses en calcul/-communication/ressources. Les applications de Cloud mobile basées sur la collaboration en Cloud mobile peuvent déployer leurs composants dans différents endroits, y compris le téléphone intelligent local, les machines virtuelles dans le Cloud ou les Cloudlets. Dans ce qui suit, nous décrivons les modèles possibles pour le Cloud computing mobile.

Le modèle Server à Client

Comme nous l'avons mentionné précédemment, le Cloud Computing Mobile vise à prolonger les capacités des périphériques à stockage/calcul limités et à fournir un accès transparent aux données/applications sur un serveur distant riche en ressources de n'importe quel endroit. Un serveur de Cloud distant agit en tant que fournisseur de services pour les appareils mobiles. La connectivité réseau entre l'appareil et le serveur Cloud doit être optimisée pour garantir la qualité du service avec un handover transparent (voir Figure 4). Avec le terme "No Cloud without Virtualization", pour réduire le temps de traitement et améliorer l'efficacité énergétique, de nombreuses solutions prennent en charge cette architecture en utilisant la technique de virtualisation telle que la virtualisation basée sur les machines virtuelles.

Le modèle Cloud Virtuel

Une autre approche [32], [33] consiste à créer un Cloud avec des périphériques mobiles interconnectés pour le stockage et le traitement des données afin que les périphériques mobiles soient des fournisseurs de ressources d'un Cloud virtuel. Dans cette architecture (voir Figure 5), les appareils mobiles fonctionnent comme fournisseurs de services ou consommateurs, et chaque appareil mobile peut ainsi coopérer avec ses voisins pour collecter/distribuer ses informations environnantes à des fins spécifiques telles que les informations sur le trafic, la surveillance de l'état de santé d'un patient, etc.

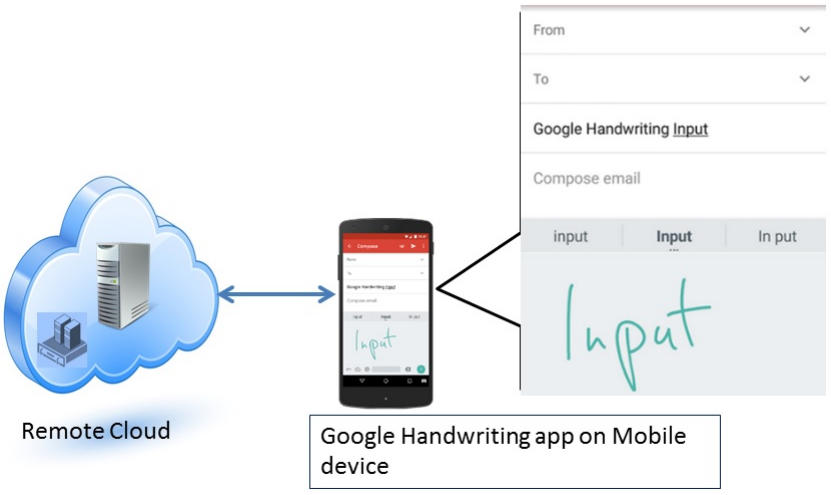


Figure 4: Modèle Server-Client

Figure 5: Modèle Cloud Virtuel

Le modèle Cloudlet

Le concept de Cloudlets tel que proposé dans [34], est une architecture alternative de *MCC* dans laquelle la Cloudlet est représentée comme des éléments de déchargement intermédiaires dans une structure à trois niveaux: Appareil mobile-Cloudlet-Cloud. Dans la Figure 6, la Cloudlet est considérée comme un ordinateur puissant et bien connecté, installé dans l'infrastructure publique avec une connectivité au serveur Cloud. Elle peut être implémentée grâce à des serveurs hotspots Wi-Fi qui prennent en charge les hyperviseurs pour gérer les machines virtuelles ou qui peuvent correspondre à de puissantes stations de base dans un environnement de périphérie mobile. Ceci est utile pour que le périphérique mobile proche externalise sa charge de travail tout en assurant un faible délai et une bande passante élevée. Prenant la dé tection d'é vé nement en temps réel par exemple, quand un touriste é tranger visite un musé e local, il fait face au problème des langues pour comprendre toutes les œuvres d'art. Si aucune Cloudlet n'est disponible à proximité, il doit se connecter à un Cloud distant avec un réseau mobile 3G/4G coûteux ou pas toujours disponible. Cependant, grâce à la Cloudlet qui est déployée partout comme hotspot Wi-Fi, il peut utiliser les services de Cloudlet pour la traduction, le traitement d'image ou l'envoi de flux vidéo à ses amis avec une faible latence, une bande passante élevée et un traitement puissant.

Simanta et al. dans [35] a proposé un exemple de mise en œuvre du concept de Cloudlets dans le but d'améliorer le traitement et la conservation de l'énergie de la batterie dans les appareils mobiles, notamment dans les environnements hostiles où les réseaux sont peu fiables. La caractéristique clé de l'architecture est que les éléments déchargés sont sans état. La communication entre la Cloudlet et le Cloud central est effectuée uniquement pendant l'installation et l'approvisionnement. Une fois la Cloudlet configurée, il fonctionne en mode de déconnexion avec le Cloud central et en mode de connexion avec le périphérique mobile. Lorsqu'un périphérique mobile est connecté à une Cloudlet, une surcouche d'application est externalisée du périphérique mobile vers la Cloudlet. Une surcouche d'application représente la différence entre une machine virtuelle de base avec seulement un système d'exploitation installé et la même machine virtuelle avec l'application installée. La Cloudlet peut héberger plusieurs machines virtuelles. Actuellement, la Cloudlet est proposée à

l'utilisation dans de nombreux domaines tels que les réseaux véhiculaires [36], le domaine militaire [37].

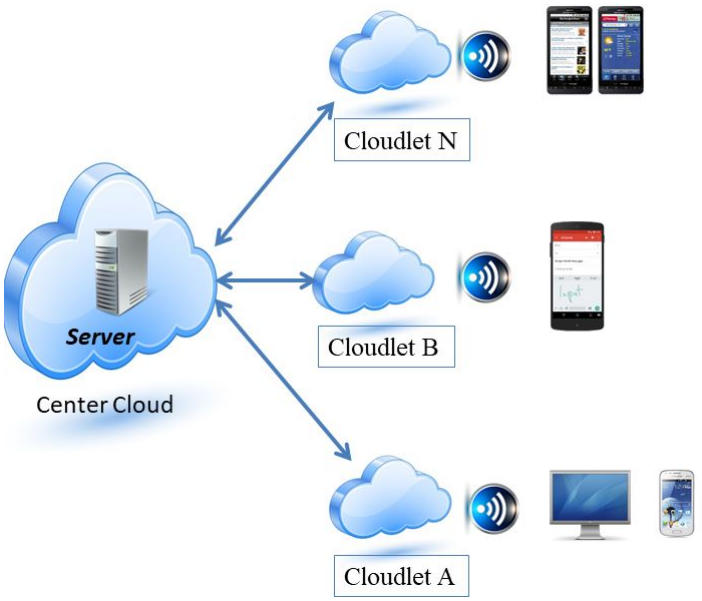


Figure 6: Modèle Cloudlet

Pour la mise en œuvre du concept de Cloudlet, les auteurs de [38] présentent l'approche qui utilise une application overlay sans avoir besoin d'un environnement pré-requis. Dans cette approche, la superposition de machine virtuelle fait référence à la différence binaire compressée entre une image de machine virtuelle de base et une image de machine virtuelle complète. L'image de machine virtuelle complète est une machine virtuelle de base dans laquelle l'application de superposition est installée. L'utilisateur mobile ne transporte que les superpositions qui peuvent être calculées hors ligne ou obtenues à partir du Cloud via Cloudlet. Comme présenté dans la Figure 7, lorsqu'un périphérique mobile est connecté à une Cloudlet, une surcouche d'application, telle que la réalité augmentée, la reconnaissance de visage ou d'objet, est externalisée dans la Cloudlet au lieu d'externaliser une VM trop lourde à transferer. Du côté de la Cloudlet, une instance de VM est créée à partir de la superposition reçue et une VM de base possédée par la Cloudlet. Ce processus est appelé synthèse VM. L'appareil mobile peut par conséquent utiliser cette instance de machine virtuelle pour ses opérations d'éxternalisation.

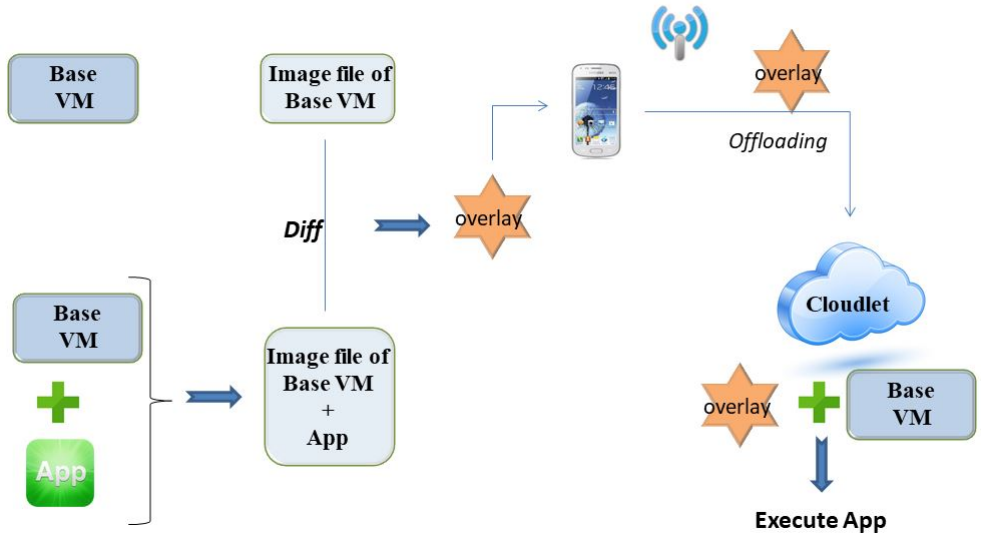


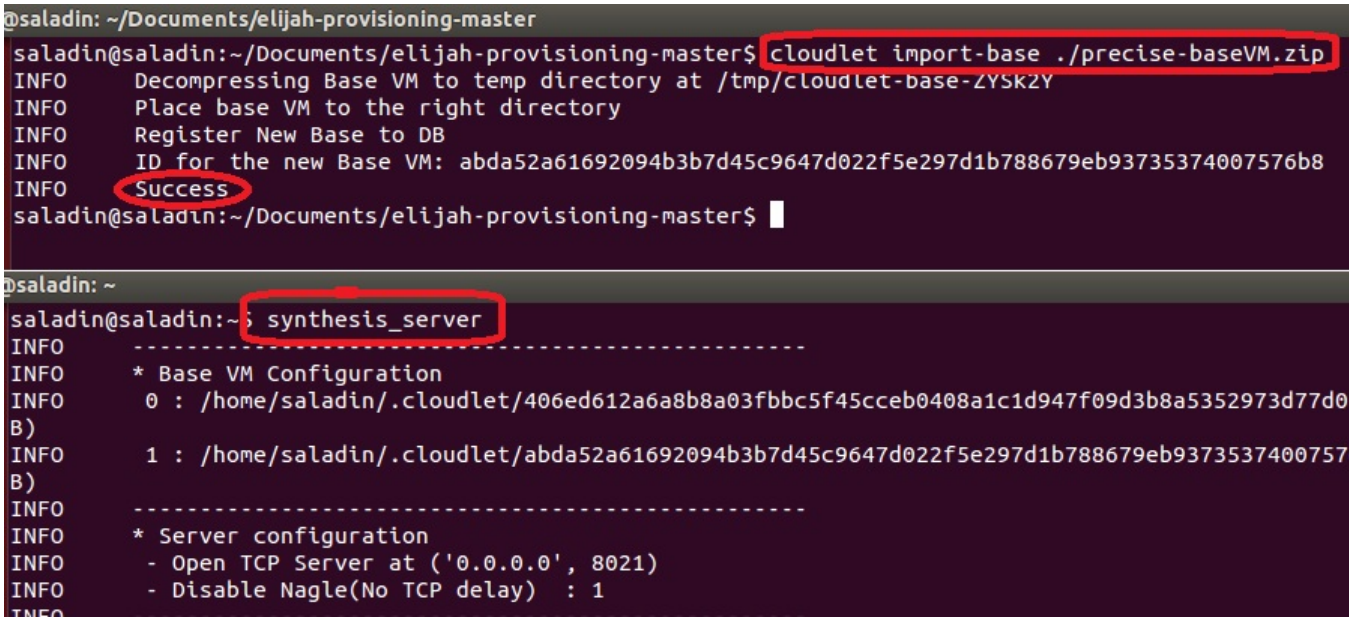
Figure 7: Surcouche de VM dans l'architecture Cloudlet

Sur la base de cette approche, nous avons également simulé ce modèle de Cloudlet [1] pour générer et transférer une superposition entre deux systèmes. La performance de cette simulation est analysée pour la comparaison avec d'autres approches du Chapitre 7, où nous présentons les contributions. Dans cette simulation, nous avons la superposition VM qui contient le programme Geany [2] du côté client. Cette superposition VM est transférée et reconstruite du côté serveur. Comme le montre la Figure 8, la Figure 8a montre la VM basée sur l'importation et le serveur Cloudlet, tandis que la Figure 8b illustre l'envoi réussi de la superposition côté client.

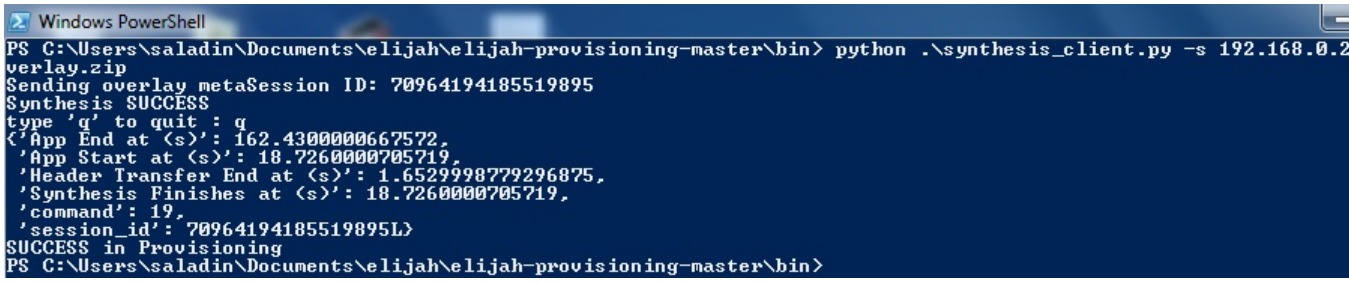
Après avoir présenté le MCC et la virtualisation, le reste des chapitres de cette thèse est résumé comme suit:

Au chapitre 4, nous étudions les plates-formes de confiance existantes et l'une des fonctionnalités les plus importantes, à savoir l'attestation à distance. Nous examinons également les travaaux récents pour déterminer leurs avantages et leurs inconvénients. La plupart de ces travaux ont souligné les inconvénients existants des mécanismes actuels d'attestation à distance. En ce qui concerne l'attestation d'application, les déficiences de BA ont été clairement montrées dans beaucoup de recherches discutées dans le chapitre.

[1] https://github.com/cmusatyalab/elijah-cloudlet
[2] https://www.geany.org/



(a)



(b)

Figure 8: Simulation Elijah-cloudlet

Pour aborder le rôle des plates-formes sécurisées dans *MMC*, le chapitre 5 présente des problèmes de sécurité pour les périphériques mobiles et le côté cloud. En effet, *MMC* profite non seulement des installations de cloud computing, mais hérite également des menaces de sécurité du cloud computing conventionnel. Outre les défis actuels liés au traitement de la fourniture de services Web aux appareils mobiles tels que perte de connexion, bande passante/latence et ressources limitées, l'utilisation du cloud pour les appareils mobiles pose également un problème de sécurité et de fiabilité. Les défis actuels de *MCC* peuvent être énumérés ici, à savoir les logiciels malveillants, les virus, la confidentialité, la fraude financière, la protection du contenu, les données d'entreprise et l'espace sécurisé. De

ce fait, il y a une prise de conscience et une accélération de l'adoption des solutions de sécurité. Afin de calculer la réputation basée sur la confiance pour *MCC*, en particulier pour l'architecture basée sur cloudlet, différentes méthodes avec une précision élevée, telles que le système de logique floue (FLS) et l'intelligence bio-inspirée ont été utilisées. Dans le chapitre 6, nous rappelons la littérature et l'avantage des systèmes de colonies de fourmis après avoir présenté les avantages des systèmes de logique floue.

Au chapitre 7, nous soulignons l'étude de cas de *MCC* en examinant l'efficacité énergétique et la démonstration dans un contexte réel. Bien que la préoccupation liée à *MCC* soit vaste, nous présentons des domaines pertinents pour l'objectif de cette thèse. En outre, la méthode de sécurité simple (c'est-à-dire le jeton) dans Droplock nous incite à développer un nouveau jeton logiciel basé sur une solution de sécurité.

Au chapitre 8, nous introduisons le jeton consistant en une propriété aléatoire, le modèle de sécurité proposé non seulement diminue le démérite restant des mécanismes d'attestation à distance, mais s'adapte également au contexte. Selon le contexte, le mécanisme d'attestation a ses propres avantages et inconvénients. Le modèle proposé décrit un nouveau schéma d'attestation basé sur les mécanismes d'attestation existants. Afin de prouver l'exactitude, nous avons vérifié le protocole proposé sous Scyther et l'avons analysé avec les preuves de sécurité. De plus, le présent protocole permet aux parties impliquées, à savoir l'entreprise et le tiers de confiance, de déployer et de gérer de manière transparente des jetons numériques pour appareils mobiles non seulement pour accomplir l'authentification, mais aussi pour sécuriser l'accès aux réseaux d'entreprise.

L'évaluation de la confiance est significative dans tout processus d'échange de cloudlet. Cependant, la détection d'intrusions stéréotypées et les mesures associées ne permettent pas d'assurer la confiance et la validation de plusieurs capteurs et dispositifs intelligents sous le cloudlet. Au chapitre 9, nous proposons une hybridation élaborée et spécifique de mesures de confiance informatisées du point de vue de l'utilisateur et de la qualité du service. Le calcul flou de la proposition est adopté pour aborder l'incertitude dans les inférences humaines tout en faisant confiance à un cloudlet spécifique. Par la suite, la performance du réseau améliore également la validation de confiance et donc les paramètres sont mesurés grâce à l'algorithme d'optimisation des colonies de fourmis. Dans le chapitre

10, afin de prouver l'avantage de notre solution, nous avons analysé la performance de nos approches par rapport aux autres. Les résultats actuels sont comparés et il est évident que le schéma hybride de la colonie de fourmi et de la logique floue peut concevoir une mesure intelligente et complète de la confiance dans le paradigme du cloudlet.

Pour finaliser la dissertation, une conclusion est donnée dans le chapitre 11.

Part I

Introduction

Chapter 1

Introduction

Mobile devices, as we know today, had not been around for a long time. It was not until on April 3, 1973 that Martin Cooper, a senior engineer at Motorola, made the world's first mobile phone call to his telecommunication company and told them he was speaking via a mobile phone. Since then, the mobile device has evolved tremendously to human life from the personal use to the enterprise purposes. With the quickly proliferating of mobile devices, we are aware that each year a wide variety of devices in different factors, improved capabilities, and more intelligence are available in the market. According to Cisco's prediction, the number of mobile devices, including machine-to-machine (M2M) modules, in use will reach 1.5 per capita by 2021 that nearly 12 billion mobile-connected devices, up from 8 billion and 1.1 per capita in 2016. In addition, mobile traffic per mobile-connected end-user device will reach globally 5,657 megabytes per month by 2021, up from 977 megabytes per month in 2016, a compound annual growth rate of 42%[1]. Mobile devices will more and more deal with heavy computation tasks because the importance of the huge size of data that will be more and more manipulated. As a result, mobile devices will replace PCs as the most common access devices for multipurpose.

With the emergence of new information and communication technologies, smart devices combined with communication infrastructures such as Internet and mobile networks need to provide reliable services. For this purpose, many application domains with high social and business impact such as personal healthcare, home automation, mobile payment, may use mobile devices that rely on trusted environment. This trusted environment may be provided

[1]http://www.cisco.com/assets/sol/sp/vni/forecast highlights mobile/

or based on many hardware or software platforms such as SIM cards, secure elements for Near Field Communication (NFC), Host Card Emulation (HCE), TEE (Trusted Execution Environment), or TPM (Trusted Platform Module). Even if these trusted platforms are extremely constrained in terms of energy, computational power and memory, they are designed to solve major constraints of security.

On the other hand, cloud computing is to research how to manage remote storage and resources that are shared by multi-users in a virtualized and isolated environment [5]. To be more specific, cloud computing delivers the latest services; namely Network as a Service (NaaS), Infrastructure as a Service (IaaS), Platform as a Service (PaaS), Software as a Service (SaaS) and Storage as a Service (STaaS) to abstract all types of computing resources as services [6]. As a result, the shared cloud infrastructure services work as the utility: the users with their smart phones can access and pay for what services they need from remotely hosted resources over the Internet. These services not only upgrade automatically but also easily scale up or down. It can be clearly seen that the advantages of cloud computing have brought benefits to mobile devices to overcome the limitations of mobile technology; namely limited data storage, processing power and battery life. Generally, the term of Mobile Cloud Computing (*MCC*) has come from the combination between mobile and cloud computing technologies.

In this thesis, after investigating the real case-study of trusted platforms as well as *MCC*'s models, we focus mainly on the challenge of security in the mobile cloud computing ecosystem with the support of trusted platforms by proposing our interest on authentication, trust and reputation in mobile cloud computing context.

1.1 Problem Statement: Trusted computing for Mobile Cloud Computing

In mobile cloud computing, the trend nowadays is to outsource data and computation of mobile devices to the cloud in order to extend the working boundaries of resource-constrained devices and to benefit from unlimited cloud capacities such as storage. In addition, *MCC* must offer mechanisms to guarantee data protection and security in untrusted environment. With the increasing use of *MCC*, security issues remain a challenge. Our work is to research

mainly on the existing trusted platforms to answer the following questions:

- What is a trust environment?
- Is a pure security solution sufficient to build a trust environment? and
- How can mobile devices cooperate with the Cloud to build a trust environment?

Furthermore, by supporting hardware to provide security primitives independent from other system components, the concept of trusted computing is not a novel notion and is described in many researches [7; 8; 9; 10]. A specific example of trusted computing technology is Remote Attestation which enables a computing system to measure properties of a remote system in such a way that remote system will be detected if it is compromising. The properties of a platform or application can be used to define security requirements in order to satisfy the basic term C.I.A (Confidentiality, Integrity, Authentication) of system security. In cloud environment, one of the problems that can occur is the management of encryption key. Several cloud customers may have their own encryption method and managing the security keys is another issue to address in the context of encrypted data [11]. By supporting protection of cryptographic keys, random number generation, cryptographically binding data to certain system configuration, sealing data in the configuration of the application and platform/application authentication, we propose to use the Trusted Platform Module (TPM), which is promoted by Trusted Computing Group, to overcome this situation. The Trusted Platform Module (TPM), a specific example of trusted platforms, has been specifically designed to be a building block for trusted computing. It is a significant use in industry and government, for example: Bitlocker for device encryption in Microsoft and security solution for laptop in the United States Department of Defense [12]. With its functionalities, the TPM protects the computing system beyond the user's control to against a deliberate or accidental attack. By considering the TPM's functionalities, we offer suggestions about a novel schema of property-based attestation suitable for use with the token based authentication.

Since, security and trust are complementary aspects that are required to build a complete and adaptive solution. Hence, the trust and reputation based intelligent, such as fuzzy logic and ant colony system, is considered here in the context of mobile cloud environment.

1.2 Thesis contribution

To be adapted to the above discussion, the main contribution of this work is threefold as presented in the following:

1.2.1 Contribution 1

The PhD dissertation starts by investigating the real case study of mobile cloud computing. We first describe the different architecture models of *MCC* and we illustrate these *MCC* architectures by examining their energy efficiency with the experimental work [13]. This approach is competent to tailor for the two subsequent contributions when we analyse the performance of various approaches. Then, for understanding the application of *MCC* model with the basic standard security level in real case, we propose a Droplock system [14]. The current objective of Droplock's approach is to improve the security of the service delivery from the supplier to the client. This work has inspired us to develop a novel security approach in the second contribution.

1.2.2 Contribution 2

By taking advantage of the combination of technologies and trends, such as trusted platforms, cloud computing and bring your own device, we introduce Property based Token Attestation for securing the mobile user in the enterprise cloud environment. Although *MCC*'s architecture can be classified into many categories [15], to simplify, we have selected the Client-Server model of *MCC* to simulate the proposed security solution.

1.2.3 Contribution 3

While considering the traditional security protocol (e.g public key infrastructure (PKI) etc.) as insufficient to secure *MCC*'s ecosystem robustly, we propose a schema for evaluating the trust and reputation based bio-inspired intelligence (e.g fuzzy logic and ant colony system). As the addition of the second contribution, the proposed schema is suitable for one-to-one relationship (Client-Server) as well as one-to-many relationship (Client-Cloudlets) where the clients can select dynamically the right service providers for fitting

their own requirements.

1.3 Thesis outline

The rest of this manuscript is organized as follows:

♢ Chapter 2 recalls several virtualization techniques and compares the virtualization based on containers with virtual machines.

♢ Chapter 3 introduces various definitions of mobile cloud computing, including its structures and related models.

♢ Chapter 4 provides a survey of existing trusted platforms and figures out how they are suitable for secure mobile-users in the context of mobile cloud computing.

♢ Chapter 5 addresses the security issues related to mobile cloud computing.

♢ Chapter 6 describes a scenario for measuring trust and reputation which also consist of the state of the art of the corresponding literature related to computational intelligence.

♢ Chapter 7 presents the first contribution by addressing the recent challenges of *MCC* in terms of energy and security under real case study and experimental work.

♢ Chapter 8 exposes the second contribution by introducing a novel framework for attestation, namely Property based Token Attestation, in *MCC*. The performance, including energy consumption and security, is also analyzed here to make a comparison between the present model and related models.

♢ Chapter 9 delivers the last contribution of the PhD work for the trust and reputation management in the context of multi-cloudlets.

♢ Chapter 10 analyses the performance between the proposed models and the other contemporary models

♢ Chapter 11 concludes and draws a summary of whole work and also presents the prime perspective of this work.

Part II

Context and State of the Art

Chapter 2

Virtualization: The beginning

With the term "No Cloud without Virtualization", virtualization technology is widely used to share the capabilities of physical computers by splitting the resources among operating systems. In 1964, the concept of virtual machines (VMs) is used in the IBM's project called CP/CMS system. The Control Program (CP) acted to split the physical machine to several copies (VMs), each copy with her own OS CMS. In 1974, Goldberg and Popek published a paper [16] which introduces a set of sufficient conditions for computer architecture to efficiently support system virtualization. In 1999, the company VMware presented its product VMware Virtual Platform for the x86-32 architecture[1]. In 2006, Intel and AMD proposed extensions to support virtualization. In this chapter, we will recall the basic principles of virtualization.

2.1 Virtualization techniques

The virtualization layer called hypervisor or Virtual Machine Monitor (VMM) manages the hardware resources between the different VMs. The guest OS is the OS that is running within a VM. Currently, there are several virtualization techniques that we summarize in the followings.

[1]VMWare Inc, Introducing vmware virtual platform, technical white paper (February 1999)

2.1.1 Full virtualization

In full virtualization, the source code of the guest OS is not modified. So, when executed, the guest OS is not aware to be virtualized. Binary translation (BT) is used to emulate a small set of instructions, i.e., privileged instructions are trapped and sensitive but non-privileged instructions are detected then translated to show a fake value. The rest of the instructions are directly executed by the host CPU. There are two types of hypervisors that handle the full virtualization. Type 1 corresponds to a native or standalone hypervisor that runs directly on the bare hardware. Type 2 is hypervisor that runs on top of the host OS (like an ordinary application). The drawback of full virtualization is that the VMM must use special tricks to virtualize the hardware for each VM, which generates an additional overhead when accessing the hardware [17].

2.1.2 Para virtualization

It refers to the collaboration between the guest OS and the hypervisor to improve the performance. This collaboration involves modifying the source code of guest OS to call directly, using hypercalls, the hypervisor to execute privileged instructions. So the guest OS is aware to be virtualized. The drawback of this technique is the poor compatibility and portability of OSs.

2.1.3 Hardware-assisted virtualization

To simplify the virtualization techniques, hardware vendors such as Intel and AMD introduced new extensions to support virtualization. Hence, two mode features for CPU execution are introduced in Intel Virtualization Technology (VT-x) and AMD's AMD-V so that the VMM runs in a root mode below ring 0 (i.e., ring -1) while the guest OS runs in non-root mode (ring 0). The state of the guest OS is stored in Virtual Machine Control Structures (for Intel VT-x) or Virtual Machine Control Blocks (for AMDV). The advantage of the hardware-assisted virtualization is the reduction of the overhead caused by the trap-and-emulate model[2] [18].

[2]https://www.vmware.com/techpapers/2007/understanding-full-virtualization-paravirtualizat-1008.html

2.2 Virtualization of hardware

In order to run successfully the virtualized systems, the virtualization layer (VMM) needs to share CPU, memory and devices. Several mechanisms are implemented to do so.

2.2.1 Processor virtualization

Generally, the OS is designed to run directly on the physical machine and fully has use of the computer resources. The traditional x86 architecture has four levels of privileges (called rings 0 to 3) dedicated to the operating system and the applications in order to manage access to the computer hardware. User applications run in Ring 3 (user mode), while the operating system runs in Ring 0 (kernel mode) to have all privileges. In the new CPU generations, an extension for virtualization called Virtual Machine Extension (VMX) has been added. In the Intel VT-x architecture, there are two execution levels: VMX root mode used to execute the VMM functions; and VMX non-root mode which corresponds to a reduced privilege to run guest OSs.

2.2.2 Memory

In traditional operating systems, a physical memory is allocated for each process defined with its own virtual address space. When the process wants to access memory, the process virtual address is translated by memory management unit (MMU) that relies on the process page table. When a guest OS runs on a hypervisor, an additional translation level is added. In fact, the guest OS maps guest virtual addresses (GVA) with guest physical addresses (GPA), and the translation from GPA to machine physical addresses (MPA) is done by the hypervisor [19].

2.2.3 Devices

The VMM has to present devices to the guest OS such as disk, network interface, USB and timers. Since the drivers of the guest OS are not aware to be virtualized, all the instructions generated by the drivers have to be trapped and emulated, which is the source of overhead. However, if the drivers are aware, a more direct path to handle devices inside

the VM can be used by installing particular drivers which talk to the para-virtual devices (like virtual interface in Linux) [20].

2.3 Isolation by containers

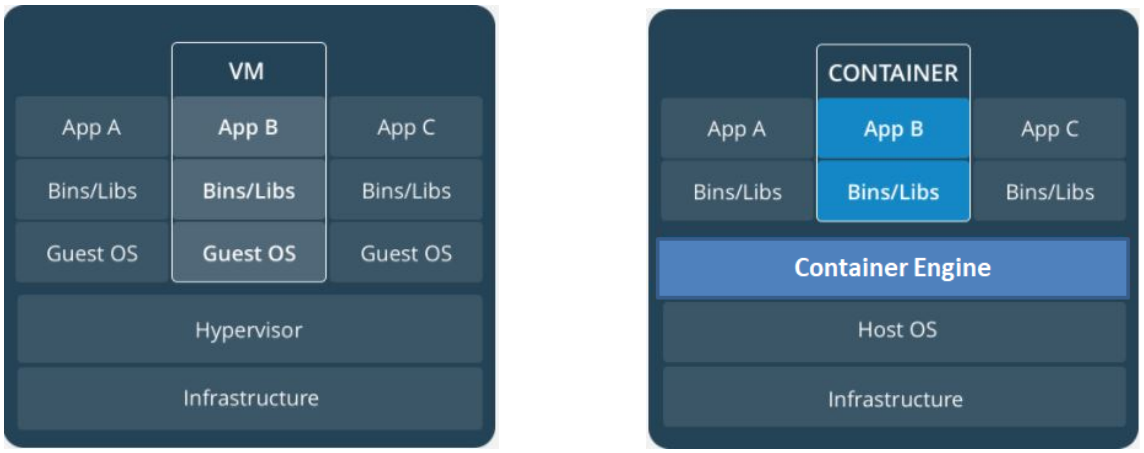
In the virtualization based solutions, the hypervisor scheduling operates at two levels as explained in the following. The first scheduling occurs at the hypervisor level where the hypervisor schedules VMs, and the second one at the virtual machine level where the guest operating system schedules processes. Taking that into account, it is obvious that this approach creates a significant overhead. Another virtualization approach, based on containers, has been proposed to reduce significantly this overhead, and thus providing better performances, especially in terms of elasticity and density within a lean data center.

2.3.1 Containers

In the container based virtualization, scheduling occurs only at one level. The containers contain processes, while the kernel and the libraries are shared between the containers. Hence, the scheduler is used only at process level to schedule processes. The behavior is similar to a classical operating system; the only difference is that, in container-based virtualization, processes are affected to a container, and between each container, an isolation is provided by the operating system.
We can define a container as a confined environment under the global environment. The Linux operating system provides some mechanisms to create these confined environments or containers, and somehow forces each container to follow a predefined policy. The isolation can occur at two levels: Isolation between processes, and Isolation between processes and the hardware [21].

2.3.2 Virtualization based on containers versus virtual machines

As shown in figure 2.1, unlike VMs that rely on a hypervisor, containers are viewed as applications within the host operation system. Consequently, as stated in [22], the performances given by using containers are better than those for virtual machines, which is a little obvious since containers add almost no overhead comparing to VMs. The information



(a) Virtual machines based hypervisor

(b) Virtualization based on containers

Figure 2.1: Virtual Machine versus Docker Container architectures

in table 2.1 present the comparison between the containers[3] and virtual machines in terms of size of images, platform dependency and migration.

[3]https://docs.docker.com/installation/windows/

	Containers	VMs
Require	A host OS	A hypervisor of type 1 or type 2
Size of images	Images are lightweight (for example, basic Ubuntu image for Docker containers has 180MB)	A VM hosts a guest OS that is a very big file (for example Ubuntu 14.04.1 has more than 1GB).
Platform dependency	Dependent-platform: the container is viewed as a bunch of processes vis-à-vis the host OS.	Independent-platform: a guest OS is hosted by a VM that is managed by a hypervisor. A guest OS is not aware of that and behaves as an OS that has total access to the physical resources.
Migration	Migration is possible only if the source and the destination platforms are Linux based. Even if Docker container engine is provided for Windows platform for example, this engine uses Linux-specific kernel features and requires a VM.	Migration is suitable for VMs viewed as files easily migratable and manageable by a hypervisor.

Table 2.1: Difference between Containers and VMs

Chapter 3

Annals of Mobile Cloud Computing

3.1 Introduction

In the recent year, cloud computing is an emerging area that affects IT infrastructure, network services, and applications by providing the framework such as the population of Amazon Web Service, Window Azure and Google App Engine. Cloud computing is a third party with large-scale storage servers and data centers used to provide infrastructures, software development and distribution platforms with low costs in the computing technology. The advantage of cloud computing has come with three popular major models that are offered today as services. The figure 3.1 represents the popular service models in the cloud environment.

3.1.1 Software as a Service (SaaS)

SaaS or on-demand software is a model for the distribution of software which provides customers access to specific software over the Internet such as Google App, Office 360. In SaaS, the provider is known as the SaaS Provider that hosts the application at its data center and the interface to the application is usually through a standard web browser. SaaS delivers its benefits to either private use or enterprise. For example, instead of purchasing the application's license, the end-user can obtain the same functions as a hosted service provided by remote servers. Since the application is managed remotely and centrally by

the providers, the complexity of software installation, maintenance and upgrade has been lessened without user intervention.

3.1.2 Platform as a Service (PaaS)

While SaaS provides the fixed functions, PaaS offers application programming interfaces (API) and programming environment on which application developer can build their own applications and host them on the PaaS provider's infrastructure. By providing virtualized servers and associated services, the PaaS service enables a user to run existing applications, or to design, develop, test, deploy, debug and host applications on the software platform work as development framework. In addition, the elasticity and scalability of underlying hardware and software platform is guaranteed transparently by the PaaS platform for the running application. The PaaS provider is responsible for monitoring application-delivery. Google Application Engine (GAE), Amazon Web Services and Microsoft Azure are particular examples of PaaS. Pricing for PaaS can be on a per-application developer license and on a hosted-seats basis [23]. In the hierarchy of models, PaaS has a greater degree of user control than SaaS.

3.1.3 Infrastructure as a Service (IaaS)

Below PaaS is the IaaS which provides the communication and storage facilities through the virtualization. By offering the virtualized infrastructure including hardware, storage, servers, data center and network components, the customers can install their own operating system, database and computing services over the provisioned virtual machines [23]. Instead of buying their own computing hardware components, the customers usually purchase IaaS based on a per-use or utility computing basis such as CPU time, storage space, and network bandwidth. Amazon EC2, Cisco Metapod or Google Compute Engine are some the popular examples of IaaS model.

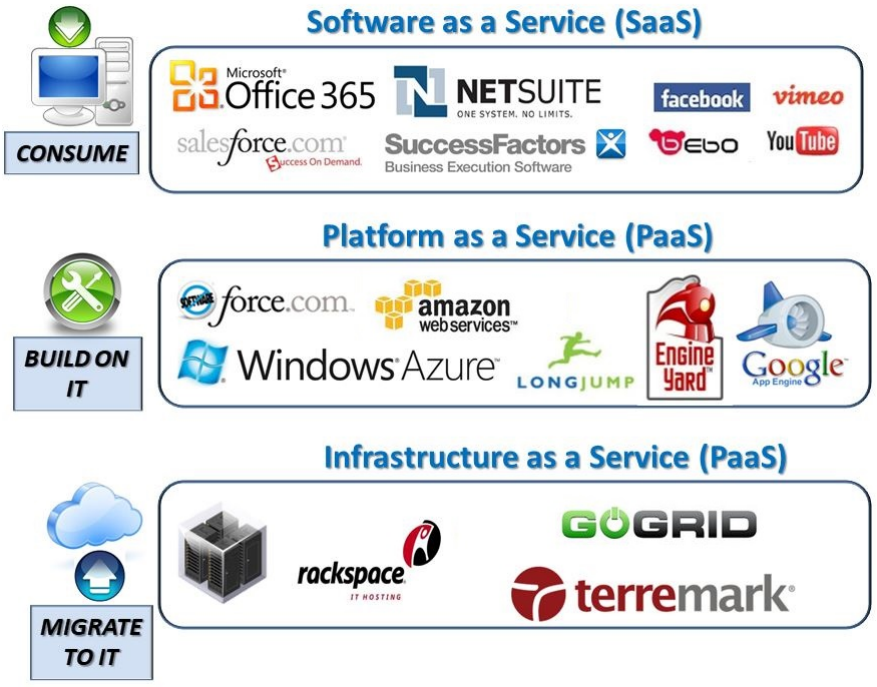


Figure 3.1: Service models of cloud computing

3.2 Mobile Cloud Computing

Along with this evolution, heterogeneous mobile devices or networked objects (from smartphones, laptops and wearable devices to embedded objects) are assumed to have the capability of sharing data [24]; studies point to the existing of more than 50 billion objects by 2020 [25]. The heterogeneity comes from software's, hardware, and architectural point of view as explained in [26]. Nowadays, mobile computing is an essential part of human life which makes daily life more convenient and effective regardless of time and place. By enabling on-demand services to the users elastically, mobile applications over cloud computing can be rapidly provisioned and released with minimal efforts of service providers and management [24]. However, mobile cloud computing cannot be simply defined as merging between mobile and cloud computing technologies. There are various definitions of mobile cloud computing that need to be considered before looking into its models and the remaining security issues. According to IBM[1], "Mobile cloud computing

[1]https://www.ibm.com/blogs/cloud-computing/2013/06/mobile-cloud-computing/

is a new platform combining the mobile devices and cloud computing to create a new infrastructure, whereby cloud performs the heavy lifting of computing-intensive tasks and storing massive amounts of data. In this new architecture, data processing and data storage happen outside of mobile devices". Similarly, Cisco Internet Business Solutions Group (IBSG) [27] defines mobile cloud as "mobile services and apps delivered from a centralized (and perhaps virtualized) data center to a mobile device, such as a smartphone. Customers access these services on-demand using the browser or thin client on their mobile devices. This contrasts to "thicker" clients that are downloaded from app stores and reside (and run) on the mobile device. Mobile cloud services are agnostic about the type of device or operating system on which they run". Another definition from Techopedia[2] stands that: "Mobile cloud computing is a technique or model in which mobile applications are built, powered and hosted using cloud computing technology. A mobile cloud approach enables developers to build applications designed specifically for mobile users without being bound by the mobile operating system and the computing or memory capacity of the smartphone. Mobile cloud computing centered are generally accessed via a mobile browser from a remote webserver, typically without the need for installing a client application on the recipient phone". The MCC architecture and its important partners are represented in figures 3.2a and 3.2b respectively.

3.3 MCC models

To highlight the motivation for mobile cloud computing, different architectures in the literature [28; 29; 30; 31] have been defined to cater to different use cases in mobile cloud computing. The major benefit of cloud computing for mobile devices is to enable running applications between resource-constrained devices and Internet-based Cloud. Hence, resource-constrained devices can outsource computation/communication/resource intensive operations to the cloud. Mobile cloud applications based on mobile cloud collaboration may deploy their components into different places including local smart phone, virtual machines in cloud or cloudlets. In the following, we describe the possible models for mobile cloud computing.

[2]https://www.techopedia.com/definition/26679/mobile-cloud-computing-mcc

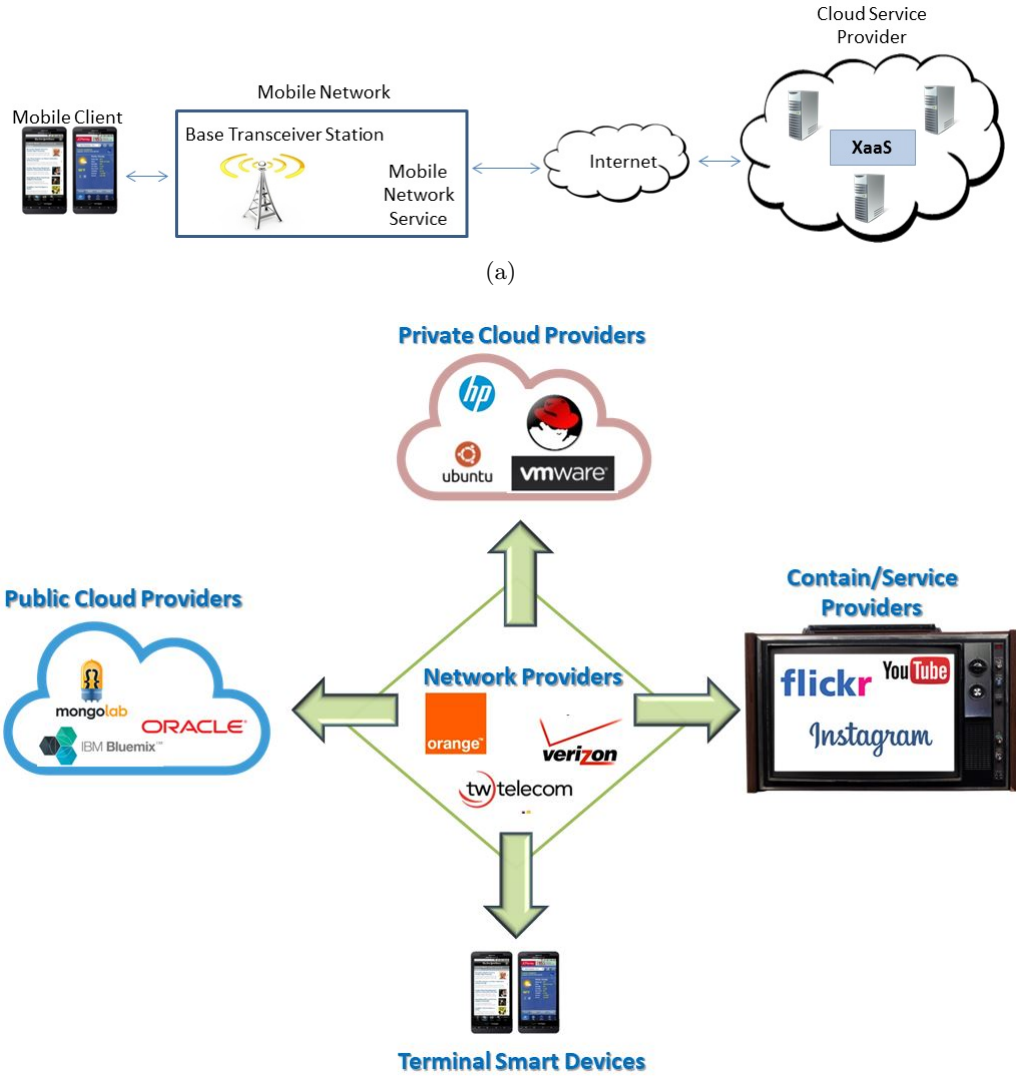


Figure 3.2: MCC architecture and its important stakeholders

3.3.1 Cloud Server – Client model

As we mentioned earlier, mobile cloud computing aims to prolong the capabilities of storage/computation-limited devices and to provide seamless access to data/application on a remote resource rich server from anywhere. A remote cloud server acts as a service provider to mobile devices. The network connectivity from the device to the cloud server needs to be optimized to ensure the quality of service and seamless handover (See figure 3.3). With the term of "No Cloud without Virtualization", to reduce the processing time and improve the efficient energy, there are many existing solutions that support this architecture by using the virtualization technique such as Virtual Machines-based, Container-based virtualization.

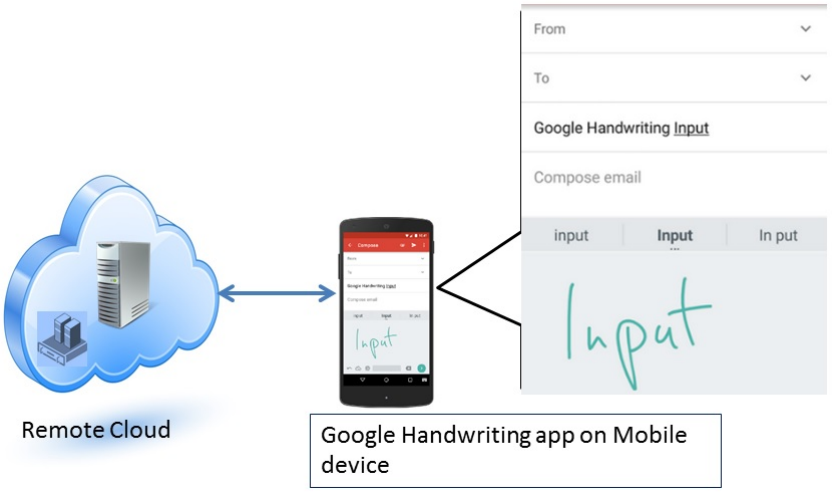


Figure 3.3: Server-Client model

3.3.2 Virtual Cloud

Another approach [32], [33] is to build up a cloud with peer-to-peer connected mobile devices for data storage and processing so that mobile devices are resource providers of a virtual cloud. In this architecture (See figure 3.4), the mobile devices work as either service providers or consumers, and as such each mobile device can cooperate with its neighbors to collect/distribute its surrounding information for specific purposes such as traffic app, healthcare monitor, etc.

Figure 3.4: Virtual Cloud model

3.3.3 Cloudlet model

The concept of cloudlets as proposed in [34], is an alternative architecture of *MCC* in which, the cloudlet is represented as intermediate offload elements in three-tier structure: Mobile device- Cloudlet- Cloud. In figure 3.5, the cloudlet is considered as resource-rich, well-connected and powerful computer installed in the public infrastructure with the connectivity to Cloud server. It can be implemented thanks to Wi-Fi hotspot servers that support hypervisors to manage VMs, or can correspond to powerful base stations in a mobile edge computing. This is useful for the proximate mobile device to offload its workload while ensuring low delay and high bandwidth. Taking real time event detection for example, when a foreign tourist man visits a local museum, he faces with the languages problem for understanding all the art-works. If cloudlet is not available nearby, he has to connect with a distant cloud with 3G/4G mobile network which is costly or not always available. However, thanks to the cloudlet which is deployed everywhere as Wi-Fi hotspot, he can use cloudlet's services for language translation, image processing or sending video streams to his friends with low latency, high bandwidth and powerful processing.

Simanta et al. in [35] proposed an example of implementation of the concept of cloudlets with the objective of enhancing processing and conserving battery power in the mobile devices notably in hostile environments where networks are unreliable. The key feature

of the architecture is that offloaded elements are stateless. Communication between the cloudlet and the center cloud is done only during setup and provisioning. Once the cloudlet is provisioned, it works in a disconnection mode with the center cloud and in a connection mode with the mobile device. When a mobile device is connected to a cloudlet, an application overlay is offloaded from the mobile device to the cloudlet. An application overlay represents the difference between a base VM with only an operating system installed and the same VM with the application installed. The cloudlet may host multiple VMs. Currently, the cloudlet is proposed to use in many areas such as vehicle-to-vehicle [36], military [37].

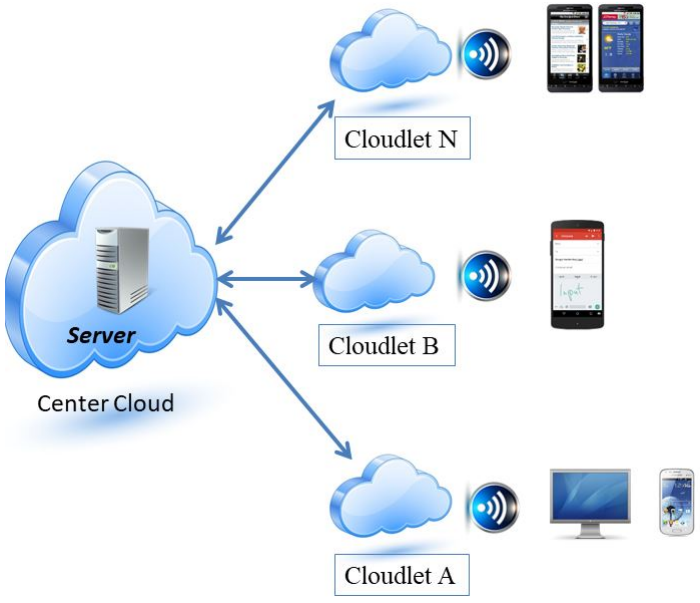


Figure 3.5: Cloudlet model

For the implementation of the concept of cloudlet, the authors in [38] present the approach that uses an application overlay without the need of a pre-requisite environment. In this approach, the VM overlay refers to the compressed binary difference between a base VM image and a complete VM image. The complete VM image is a base VM in which the overlay application is installed. The mobile user carries only the overlays that can be either calculated offline or obtained from the cloud via cloudlet. As presented in figure 3.6, when a mobile device is connected to a cloudlet, an application overlay, such as augmented reality,

face or object recognition, is offloaded to the cloudlet instead of offloading a VM which is too heavy for a transfer. In the cloudlet side, a VM instance is created from the received overlay and a base VM possessed by the cloudlet. This process is called VM synthesis. The mobile device, consequently, can use this VM instance for its offload operations.

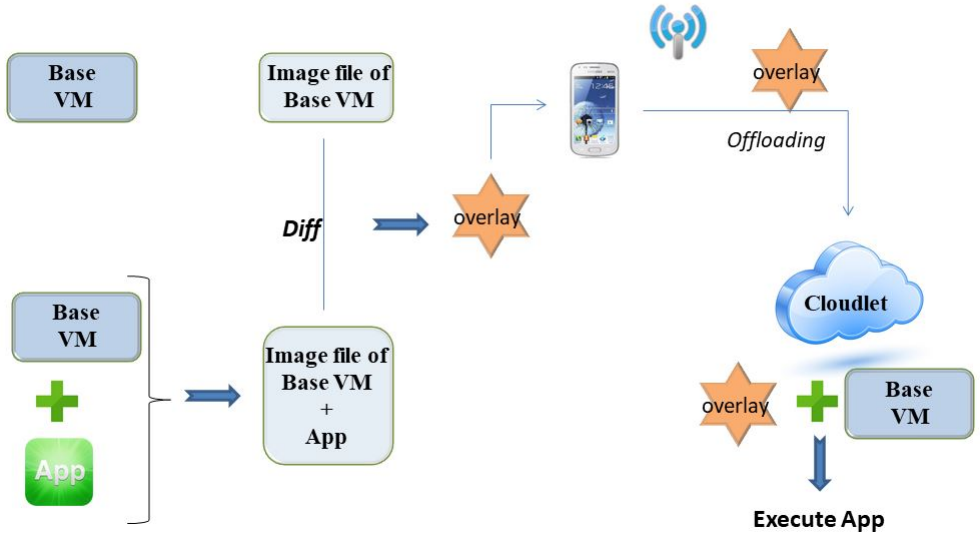


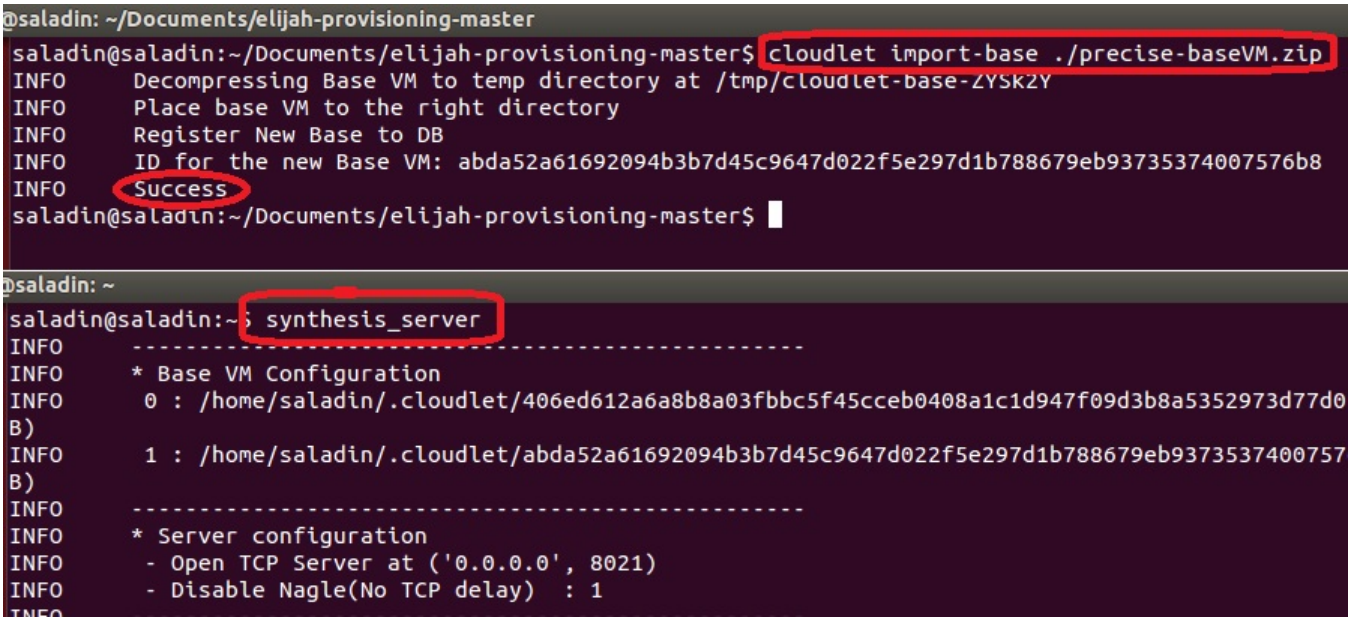
Figure 3.6: VM Overlay in the Cloudlet architecture

Based on this approach, we also have simulated this cloudlet model[3] for generating and transferring an overlay between two computer systems. The performance of this simulation is analyzed to compare with other approaches in chapter 7, where we introduce the contributions. In this simulation, we have the VM overlay that contains the geany program[4] in the client side. This VM overlay is transferred and reconstructed in the server side. As presented in figure 3.7, figure 3.7a shows the import based VM and running the cloudlet server, while figure 3.7b demonstrates the successful sending of the overlay in the client side.

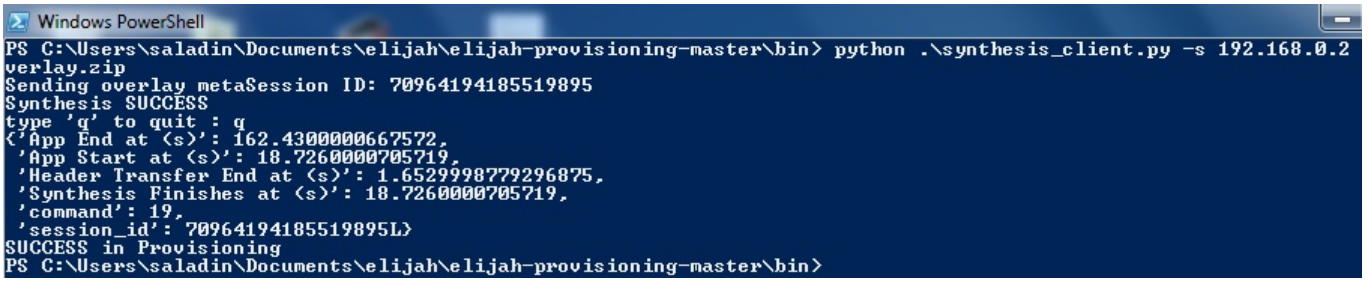
3.4 Summary

In this chapter, we introduced the well-known services of cloud computing and also discussed the existing models of *MCC* that related to the PhD's contributions. In the next

[3]https://github.com/cmusatyalab/elijah-cloudlet
[4]https://www.geany.org/



(a)



(b)

Figure 3.7: Elijah-cloudlet simulation

chapter, Trusted Platforms and their features are presented.

Chapter 4

Trusted Platforms: Computing with Trust

4.1 Introduction

The emergence of a new generation of mobile devices leads to the growing number of security concerns demand attention. Mobile users use their smart-devices for a variety of applications in an open environment. They need the reliable services to reduce the risks associated with the security threats. To do this, mobile devices need Trusted Platforms to meet a proper implemented security. Before discussing the existing Trusted Platforms, a definition is required to understand what is meant by the term "Trusted Platform". There is a wealth of research studies on the concept of trusted computing in a wide variety of publications. For example, the authors in [39] presented the effective survey on the existing solutions of trusted platforms. Generally, the trusted computing is about verifying the system is trustable to another one and reducing the scale, complexity of the entities that need to be trust. The prime goal of a Trusted Platform is to establish the secure foundation for the rest of the system that can be built on. According to Trusted Computing Group (TCG), "a Trusted Computing Platform is a computing platform that can be trusted to report its properties"[1]. TCG also defined the trusted platform as a platform is in such manner as "the expectation that a device will behave in a particular manner for a specific purpose". In the same way, the authors in [40] claimed that "A Trusted Platform

[1]https://www.trustedcomputinggroup.org/wp-content/uploads/TCG_Glossary_Board-Approved_12.13.2012.pdf

is a computing platform that has a trusted component, probably in the form of built in hardware, which it uses to create a foundation of trust for software processes" or "a Trusted Platform is defined as a computing platform that has a trusted component, which is used to create a foundation of trust for software processes". It is worth noting that, a distinction between trusted entity and trustworthy entity in computing system needs to be drawn. While an entity of a system is trusted means that the security of the system depends on it, the entity is trustworthy refers to the entity deserves to be trusted. To be more precise, the author in [41] discussed this distinction as "The proper definition is that a trusted system or component is one whose failure can break the security policy, while a trustworthy system or component is one that will not fail".

Due to the ongoing threats from hackers, viruses, and worms continue to make security to be a top priority for IT and business professionals in both the private and public sectors, we need a standard that allows affordable authentication, encryption, and network access to be accomplished on a variety of computing platforms. To address this worrying problem, the Trusted Platform supports the reliable services to mitigate the risks associated with the security threats even though this does not solve all the attacks that we face.

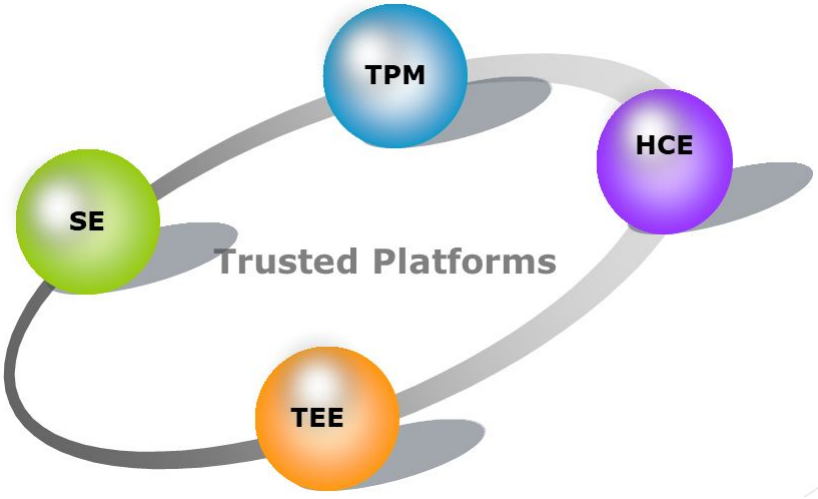


Figure 4.1: Trusted Platforms

In this chapter, we present several well-known types of Trusted Platforms (refer figure 4.1), such as Secure Element, Trusted Execution Environment, Host Card Emulation and Trusted

Platform Module, in either hardware or software form. We also make a comparison among these platforms to show how they can be suitable for securing mobile cloud computing.

4.1.1 Secure Element (SE)

A smartcard is a tamper resistant micro-controller [42] whose security is enforced by multiple software and hardware countermeasures. A secure element (SE) is a smart card embedded in an NFC device such as NFC-enabled smart phone, able to store sensitive data such as PIN code and security keys and perform internal computation such as cryptographic processing when needed by the terminal (phone). A SE has the following physical features: 8, 16 to 32 bits of CPU, 32 to 64 kB of PROM that hosts the operating system, and 1 to 4 kB of RAM to execute programs, 1kB to 128 kB of EEPROM/Flash to store data and applications, while a dedicated crypto-processor is used for cryptographic computation. To interact with a SE, the terminal sends application protocol data unit (APDU) commands and the SE responds using APDU responses.
The SE is generally a Java Card platform (refer figure 4.2) composed of a Java Card Virtual Machine (JCVM), a Java Card API and a Java Card Runtime Environment (JCRE). JCRE implements the Java Card mechanisms that are intrinsic to the smart cards such as transaction management. When the SE is a SIM card, it is enriched with Java Card packages that allow interaction with the Mobile Network Operator (MNO) such as sending SMS, generating a phone call, etc.

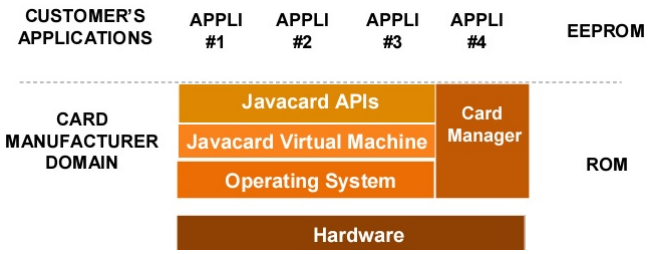


Figure 4.2: Java Card Platform [3]

In addition, Java Card platforms are designed in respect to Global Platform specifications[2]. In other terms, each Java Card platform is composed of isolated EEPROM areas called

[2]https://www.globalplatform.org/specifications.asp

security domains (SD) as in figure 4.3. A SD is assigned to one service provider (SP) and may contain one or several applications belonging to the same SP. Each Java Card application is composed of one or several Java Card applets, generally called Cardlets.

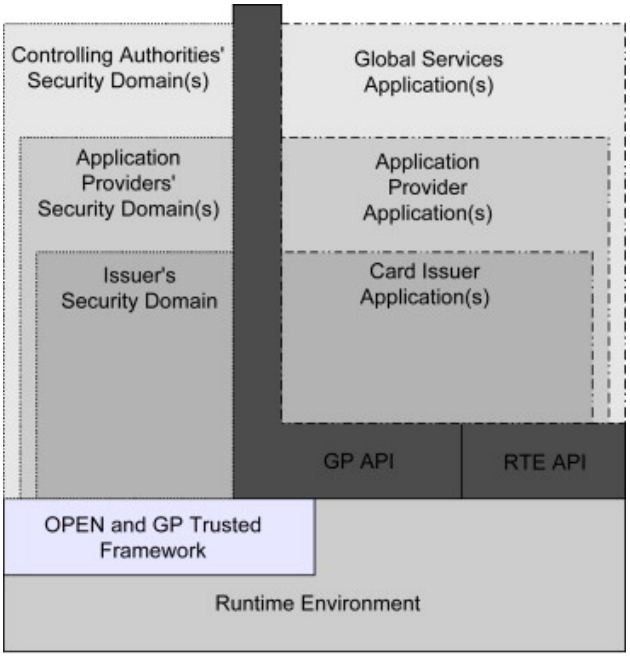


Figure 4.3: Security Domain as defined in GlobalPlatform [4]

As in figure 4.4, each NFC device has a NFC controller that allows a contactless communication with respect to three modes:

- Read/write mode: the application running on the mobile device is able to read and write data on RFID tags according to an NFC Data Exchange Format defined by NFC forum[3]. Since 2010, Google proposed an API to implement this mode.

- Peer to peer mode: called Android Beam in Android platforms, this mode allows a tap of NFC devices together to exchange data between them.

- Emulation card: unlike the other modes, the emulation card mode involves the use of a SE and allows communication between the mobile and a contactless reader. In case SE is a SIM card (refer figure 4.4), a Single Wire Protocol (SWP) connects the SIM

[3]https://nfcforum.org/

card with the NFC controller to perform the communication with the contactless smartcard reader infrastructure.

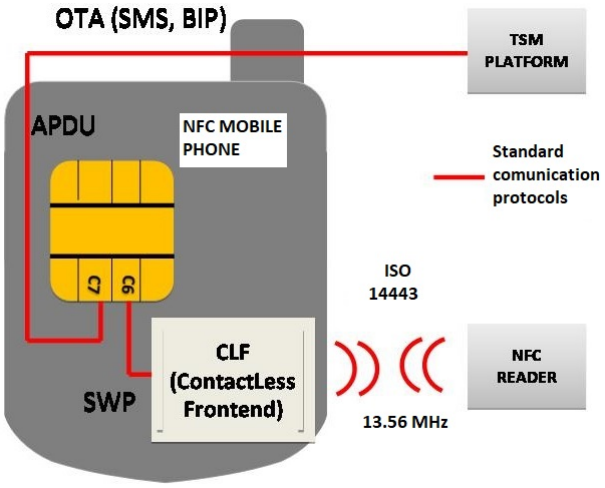


Figure 4.4: NFC mobile device

4.1.2 Host Card Emulation (HCE)

"Card Emulation" has been initially defined by the NFC Forum and integrated as a part of the core set of NFC specifications. HCE (Host Card Emulation) allows the software emulation of a smart card-based application. Prior to December 2013, HCE was deployed firstly in Blackberry OS. Since then, the feature has also been supported by Android OS, following the launch of version 4.4, codenamed "KitKat". It provides an API set that helps developers to control the NFC interface and send commands to NFC-enabled devices. In this case, the data is routed to the host CPU on which Android applications are running directly, instead of routing the NFC protocol frames to a secure element. HCE operation and protocols are shown in figure 4.5[4].

Applications could therefore be developed to emulate any classic contactless smartcard application using the ISO 14443-4 standard, such as loyalty, access control, ticketing or payment applications. In addition, HCE is simply an ordinary Android application without any specific hardware or software-based security services. Security concerns relating to

[4]https://developer.android.com/guide/topics/connectivity/nfc/hce.html

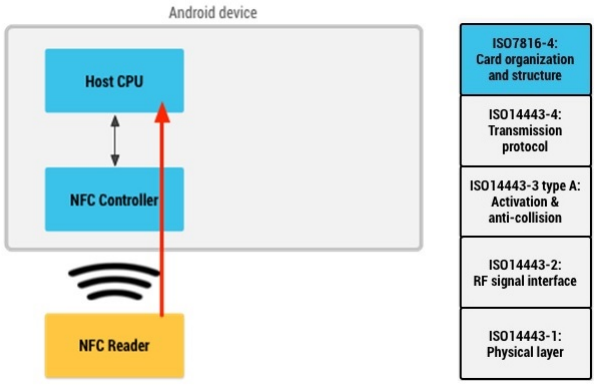


Figure 4.5: NFC card emulation without a secure element HCE supports APDU stack protocol

HCE are further compounded by industry reports indicating that Android is, by some distance, the most attacked of all mobile environments.

According to Simalliance[5], HCE is effective for the NFC ecosystem as a whole; it will make NFC more accessible and versatile to developers, and more familiar to end-users, increasing mass market adoption as a result. It is also best suited to use cases where the user's stored credentials are of low value and where stringent security requirements are not mandatory and the emulated NFC application is not based on direct implementation of a current, pre-existing card application. However, service providers evaluating HCE for payment and other high-value NFC services should proceed with caution. HCE presents a new raft of challenges and has the potential to diminish both the transaction security and the end user's NFC service experience. HCE remains in its infancy. Until more OS providers support the HCE model, deployments will remain vulnerable to the prevailing challenges associated with global Android OS utilization.

As a software solution, HCE is considered as less secure than a physical SE. For example, the HCE cannot handle counter-measures for physical attacks. Unlike secure devices, HCE concept does not target applications with high-level security, and relies on larger resources (memory & processor) because it uses the host OS. In addition, it is independent from SE providers.

[5]http://simalliance.org/

4.1.3 Trusted Execution Environment (TEE)

The GlobalPlatform Trusted Execution Environment (TEE) defines a standardized isolation environment for mobile devices in which sensitive code, data and resources are processed away from the main operation environment, software and memory on the device. This isolation is enforced by hardware architecture and the boot sequence uses a hardware root of trust in the system on chip package making it highly robust against software and probing attacks. In addition, code running in the TEE and using protected resources (known as "Trusted Application") is cryptographically verified prior to execution, leading to high integrity assurance. Because it provides an isolated runtime environment entirely inside the mobile device, the TEE enables advanced device or peripheral security use cases such as securing the user interface, or controlling access to an NFC chip. As such, the TEE can be used as a distinct security coprocessor or provide a trusted "bridge" between the user and other security technologies such as secured user interface or OS user permissions on one side, and Secure Element access control on the other. The main operating system and rich applications then run as normal on the device, accessing the functionality of the Trusted Applications via a standardized "Client API". Trusted Applications are written to an "Internal API" which ensures portable trustworthy access to secure resources, cryptographic operations and secure storage regardless of the underlying SoC hardware. Figure 4.6[6] shows the architecture of the TEE. Furthermore, the TEE manages and executes trusted applications built-in by device makers as well as trusted applications installed as people demand them. Trusted applications running in a TEE have access to the full power of a device's main processor and memory, while hardware isolation protects these from user installed apps running in a main operating system. Software and cryptographic isolation inside the TEE protect the trusted applications like ARM Trustzone platform[7].

Device and chip makers (such as ARM) use TEEs to build platforms that have trust built in from the start, while service and content providers rely on integral trust to start launching innovative services and new business opportunities. Hence, the Trusted Execution Environment (TEE) is a secure area that resides in the application processor of

[6]https://www.globalplatform.org/mediaguidetee.asp
[7]https://www.trustonic.com/

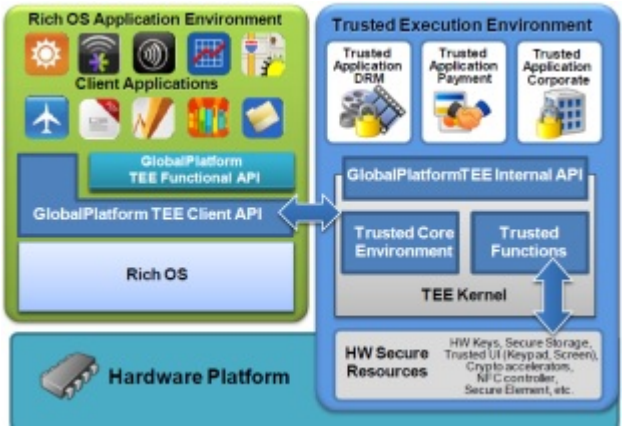


Figure 4.6: Architecture of GlobalPlatform TEE

an electronic device. Devices with a TEE will provide consumers with more secure, user-friendly experiences that simplify and speed up how they interact with their digital world. This will enable them to use their smart, connected devices more frequently to access an increasing range of applications and services in a secure way. This includes mobile payment, enterprise productivity and mobile banking applications, as well as online commerce and premium content services. SierraVisor and SierraTEE (ARM), ARM TrustZone, and Trustonic TEE are some examples of platforms implementing the TEE concept. The motivation of TEE is to serve as a combination between mobile-device OS (called rich OS) performance and SE security; because SE is resilient to attacks but limited by its resources and the rich OS is vulnerable to attacks. Figure 4.7[8] shows the shape of arrows including width and height to represent security and usability characteristics in Rich OS, TEE and SE. Taking "Attack Resistant" as a specific example, arrow's height indicates that the Attack Resistant of the three is acceptable; the width shows that SE is the best and TEE, Rich OS are fine and poor respectively.

With larger resources than a SE, the RAM size of a TEE may reach 1MB with a Flash memory shared with a Rich OS. TEE may cooperate with SE to guarantee a high level of security.

[8]https://www.globalplatform.org/mediaguidetee.asp

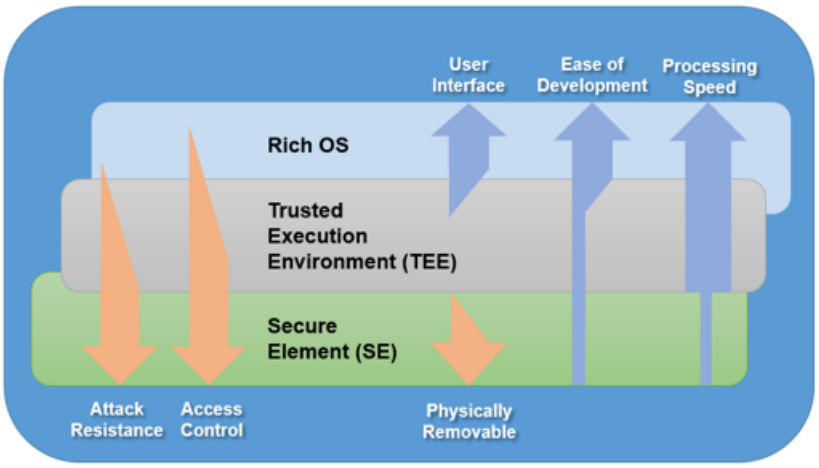


Figure 4.7: Rich OS, TEE and SE positioning

4.1.4 Trusted Platform Module (TPM)

TPM is a hardware platform with encryption computing unit and secure storage component. TPM is dedicated to PCs, servers, printers, or mobile phones to enhance security in an ordinary and non-secure computing platform and to convert them into trust environments. For mobile devices, Mobile Trusted Module (MTM) [43] refers to this secure hardware chip. TPM is the core component of Trusted Computing Group (TCG) which is a consortium of companies [44]: Compaq, HP, IBM, Intel, Microsoft, AMD, etc. The TCG requires three Roots of Trust in a trusted platform: Root of Trust for Measurement (RTM), Root of Trust for Storage (RTS), and Root of Trust for Reporting (RTR). It has recently released the TPM 2.0 library specification that provides a support for additional cryptographic algorithms, enhancements to the availability of the TPM to applications, enhanced authorization and management mechanisms. According to TCG, "specifications will detail how the TPM can be implemented in various platforms through TCG platform specific specifications. These future specifications include the TPM Software Stack specification (TSS) and separate specifications for PCs, mobile, embedded and virtualized platforms". Moreover, by supporting protection of cryptographic keys, random number generation, cryptographically binding data (refer figure4.8[9]) to certain system

[9]https://www.cisco.com/c/en/us/about/press/internet-protocol-journal/back-issues/table-contents-57/153-internet.html

configuration, sealing data in the configuration of the application and platform/application authentication [28], TPM implements mechanisms and protocols to ensure that a platform has loaded its software properly. By protecting the system at hardware level, TPM is also known as the primitive security that allows affordable authentication, encryption, and network access to be executed on a variety of computing platforms. It stores secret keys to encrypt data files/messages, to sign data, etc.

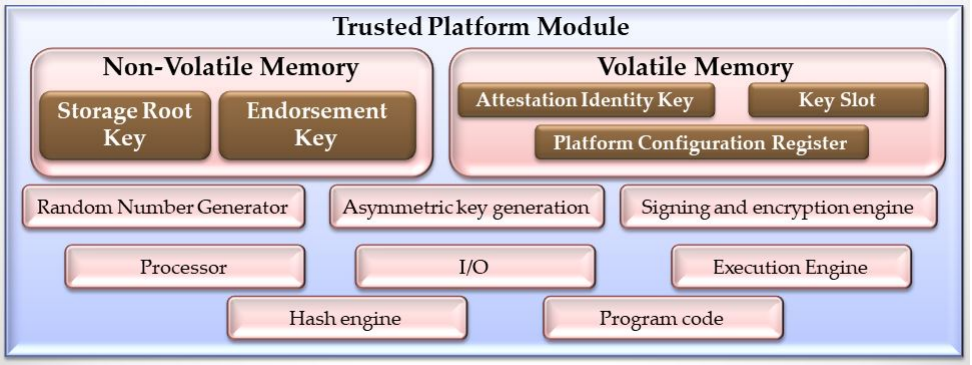


Figure 4.8: Architecture of Trusted Platform Module

In addition, TPM is equipped with a set of special registers like the PCR (Platform Configuration Register) to store integrity metrics. These metrics are used to measure the integrity of any code from BIOS to applications. Therefore, $PCR_0 \rightarrow PCR_n$ (n= 23 with TPM 2.0) provide evidence of a certain state of the system. For example, each time when the system event occurs, TPM computes the hash of a metric value and extends to the PCR associated with the occurred event. The operation to extend new value to the PCR [45] is

$$PCR_{newvalue} = Digest\ of\ (PCR_{oldvalue} || Data\ to\ extend) \tag{4.1}$$

In remote attestation as discussed in next section, it is important to know that the user communicates with a genuine TPM. The computing platform uses its TPM's Attestation Identity Key (AIK) to show its identity and provide its valid evidence with other parties. The private part of AIK is also used to sign message/data.

4.1.5 Remote Attestation

Remote Attestation (RA) is a mechanism for either authenticating to a remote entity or validating the integrity of the application/platform. Currently, the point of view in remote attestation is full of change and variety. Depending on different research aspects, RA can be categorized into two mechanisms and three techniques to serve for the same goal of the remote attestation which is to ensure the trustworthy of the current platform/applications. The former is Binary Attestation (BA) and Property based Attestation (PBA) [46]. The latter is Hardware-based, Software-based, and Hybrid technique [47]. The remote attestation is also one of the most important functionalities supported by the TPM. It is used for software or hardware to prove its identity/trusted state/configuration with the third party. Before delivering to the end user, the TPM is built-in with TPM vendor and platform endorsement keys (EKs) and corresponding certificates by the TPM vendors and platform manufactures. While the TPM vendor certificate claims that "This endorsement key is resident on an authentic TPM manufactured by me", the platform manufacturer certificate shows, "This key is resident on a TPM that is part of my platform, and this platform supports certain TPM features" [9]. By using these evidences, the platform/TPM can prove its trusted state remotely with other entities over a network. The Remote Attestation protocol (See figure 4.9) starts with the Challenger (*Chal*) that challenges the Attestator (*Att*) by sending a request for attestation (step 1). *Chal* is a trusted entity that wants to attest the authenticity of *Att* before communicating any data with it. As, the EK of TPM is unique and cannot be revealed outside TPM, *Att* then generates AIK instead of using EK. *Att* cannot link different AIKs from the same TPM. The public part of AIK, which is signed by EK, is sent along with other identity proofs such as platform certificate, EK certificate, etc. to trusted third party (Privacy Certificate Authorization noted as PCA). This can associate AIK and EK (step 2). After validating the received data, the PCA generates and signs an AIK certificate. The certificate of AIK is sent back to *Att* (Step 3). *Att* now can send this certificate, PCR values signed by AIK and Stored Measurement List (SML), which contain the measured value of platform state, to *Chal*. In the challenger side, it verifies the certificate of AIK and uses AIK to verify the signature of PCR values for recalculating the value as listed in SML. *Chal* compares this recalculated value with

PCRs value. If the PCRs value and SML do not match, it indicates that the platform has been tampered and Chal determines that *Att* is an untrusted system. Otherwise, it decides to trust *Att*. The configuration of *Att* can be stored in whitelist/blacklist if it is the trusted/blacklist system.

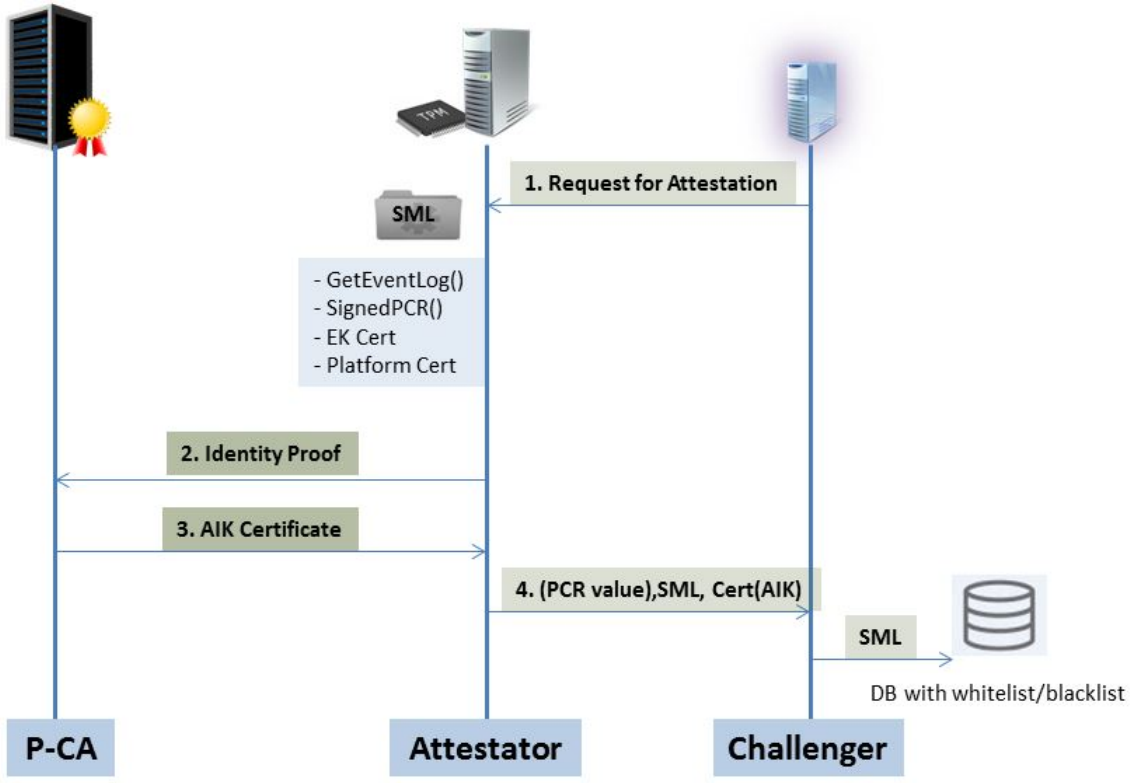


Figure 4.9: Remote Attestation

In the other hand, another approach of attestation, namely Property based Attestation (PBA), is founded on the Binary Attestation (BA). This approach not only relies on the TPM functionalities to verify the evidence of platform but also requires a trusted third party to issue a Certificate of Property. Instead of attesting binary values, platform/application's properties, functions, or behaviors are attested to prevent binary information leaking. The principle of PBA is that various platforms may have the same properties which fulfilling the same requirement regardless a variety of platforms and components. In comparison with BA, Aarthi et al.[46] stated the following advantages of PBA: 1) Properties do not reveal implementation details of a system and can therefore hide system vulnerabilities;

2) Properties may not identify components and may provide a certain level of privacy. 3) Properties of components may not change as often as hash values particularly during updates; 4) Properties are easier to understand and can be useful to write meaningful access control policies rather than using an excess of binary values. Taking the login function of the computing system (e.g. application or web page) for example, despite existing the difference in configuration, implementation, types of application/website or using-purpose, these systems have two common basic factors, such as identity of user and password. These factors can be considered as properties [48]. Thus, if these systems have been updated or changed, these properties still remain for their own purposes.

4.2 Comparison of Trusted Platforms features

After discussing the overview of existing Trusted Platforms, in this part, we compare (refers table 4.1 for the summarization) the different features of trusted platforms as the following criteria.

a. Form and Dependency

   Different from SE and TEE (refer table 4.1), HCE and TPM are only in the software and hardware form respectively. The software is simply the android program where the user's stored credentials are of low value. The latter is a computer chip and considered as the cheapest way to enhance security in an ordinary, non-secure computing platform. In term of dependency, TEE works as the framework for mobile-device security. It is also known as a layer between mobile OS and typical SE.

b. Hardware

   SE, TPM and some implementations of TEE have their own limited hardware resource. In contrast, HCE and TEE have larger resources because they have access to the host resources. Thus, the TEE provides a more powerful processing speed capability and greater accessible memory space than SE. Unlike the other platforms, HCE has at its disposal all the memory resources of the Android platform, but HCE is not a piece of hardware.

c. OS and API support

At the moment, HCE is not isolated from the device's operation system and supported by Android OS in version 4.4. However, in the software form, HCE is considered less secured than others.

d. Security features

In this aspect, the SE is the highest security level for the mobile computing. However, it also required extra cost for the implementation. In some cases, the TEE is sufficient for most applications. Moreover, the TEE can work with another trusted platform to enrich the security level. For the moment, TEE standardization is not yet finalized. By contrast, HCE concept does not target applications with high-level security. Security concerns relating to HCE are further compounded by industry report indicating that Android is, by some distance, the most attackable of all mobile environment [49].

e. Role

The role of HCE is a blur in comparison with other trusted platforms. Some drawbacks deal with HCE such as low power mode, roaming and no data connectivity scenarios and transaction speed. As stated in [49], HCE still remains in the beginning stage and needs a long way to go before it can establish a certification schema that is comprehensive and robust enough to gain the confidence of service providers and end-users.

f. Physical / Software attack

The SE's physical protection depends on the type of chips. Most of them have strong protection. TEE, SE and TPM are designed to prevent from unauthorized software attacks. They work effectively in many use cases including e-commerce, citizen to government applications, online banking, and other fields where a high security level is required.

g. Privacy

In order to be against the increasing of the security threats, the TPM supports covering privacy protection and interoperability on multi-platforms including PCs and mobile devices. Hardware manufacturers can flexibly implement the architecture of TPM chips. The owner can also flexibly deploy this architecture depending on

the security solutions. Regarding the software implementation of available trusted platforms, HCE, support strongly by Google, has gained some attention in the NFC technology. With SE-less, HCE faces with many security risks in comparison to SE and also requires more attention from the developer side. However, it enriches some factors; namely development costs, time to market and the need to cooperate with other parties.

4.3 Virtual Trusted Platforms

With the advantage of virtualization technology [17; 50; 51], the convergence between trusted computing and virtualized computing enables a novel security method. In the context of virtual environment, because the physical TPM that is linked to the underlying physical machine cannot attest directly guest OSs, it has been virtualized to provide its functionalities for each virtual machine (VM), which is running on a single platform. This makes VM feel it has its own private TPM. As a result, by virtualizing, a single physical TPM works as the Root of Trust which can be served by multi virtual environments on the trusted platform. For this reason, there are different implementation models for TPM virtualization [52; 53; 54; 55]. As we will see it in the next subsections, normally the TPM virtualization can be implemented by migrating the real TPM to the virtual machines or emulating the TPM in software [55]. Table 4.2 presents the difference between the physical TPM (noted pTPM) and virtual TPM (noted vTPM).

4.3.1 TPM virtualization via Virtual Machine Monitors

This implementation is proposed by many authors for the migration of virtual TPMs (vTPMs) and their associated VMs to provide secure storage and cryptographic operations. Berger et al. [52] introduced an architecture where the TPM's specifications are available in virtual environment. Sadeghi et al. [56] proposed a vTPM architecture, which is based on Berger's approach, to improve the maintainability and applicability of VMM. England and Loeser, in [53], proposed a para-virtualized TPM sharing the approach in which a physical TPM can be shared among virtualized hosts. To enhance the security, Danev et

Criteria	SE	TEE	TPM	HCE
Form	Comprised of software and tamper resistant hardware.	Made up of software and hardware.	Hardware-Embedded in a host platform	Software-Android Applications.
Dependency	Autonomous entity (like a little PC).	A secure area that resides in the main processor mobile devices.	Bounded with the host platform	User defined or OS Library defined.
Hardware	Processor, cryptoprocessor, EEPROM, RAM.	Processor, cryptoprocessor, EEPROM, RAM. The memories may be shared with the host device.	Cryptoprocessor, EEPROM,RAM	Not available.
OS	Within the chip.	Separate and run in parallel with the host OS	Bootstrap in the TPM, OS in the host platform.	Operate in the host OS.
API support	Limited	Limited	Limited	Rich API support
Security features	Stores sensitive data (PIN code, IMSI), secret keys, generate session keys.	Provides a framework for security within the device.	Trust root, generates asymmetric key pair, binding, identity attestation, protected objects.	Security is dependent on device OS.
Role	Used for identification/authentication, for data encryption during communication	Used for robust, hardware-backed, scalable-consistent, OS-independent security.	Used to secure the host platform, to remote communication.	Used for the software emulation of a smart card-based application.
Physical attacks	Many studies (the chip is endowed with counter-measures).	Depending on hardware features of hosting platform.	No study	No counter-measures for physical attacks.
Software attacks	Many studies (the chip is endowed with counter-measures)	Resistant to software attacks.	Resistant to software attacks.	Low resistance to software attacks.
Privacy	Controlled by the user.	Controlled by the user.	No control by the user.	Controlled by the user.

Table 4.1: Comparison of features of Trusted Platforms

al. [57] delivered a new vTPM key hierarchy by proposing an intermediate layer of keys between pTPM and vTPM. In what follows, we introduce a brief background on TPM virtualization in terms of architecture and migration.

Critera	Physical TPM (pTPM)	Virtual TPM (vTPM)
Resources	Endorsement Key (EK), Storage Root Key (SRK), Attestation Identity Key (AIK) and Platform Configuration Registers (PCRs)	vEK, vSRK,vAIK and vPCRs
Specifications	Standardized by Trusted Computing Group (TCG)	Imitate the functionality of the pTPM
Security	Trust anchor, high security level	Low security level in comparison with pTPM
Operation platform	One to one	Multi to one

Table 4.2: Difference between pTPM and vTPM

a. vTPM architecture: Berger et al. [52] designed a vTPM architecture that enables a physical TPM run on systems running parallel multiple operating systems. In this architecture, the vTPM manager is responsible for multiplexing requests and creating a guest vTPM instance corresponding to its guest VM with configured TPM support. Each vTPM instance imitates the interface and functionality of the hardware TPM. Figure 4.10 demonstrates the generic architecture of vTPM.

b. vTPM migration: vTPM migration protocol is one of the most important features on TPM virtualization based VMM. To keep the correct operation of guest applications, the vTPM must be migrated to the corresponding VM. A secure migration relies on the TPM key migration facilities which is supported by the existing TPM standard. It also requires synchronizing VM-vTPM state during the migration. In [52], the authors presents vTPM migration procedure by using asymmetric and symmetric key. In the protocol, the state of vTPM is encrypted and packaged on the source platform and decrypted on the destination platform. Instead of using a migratable

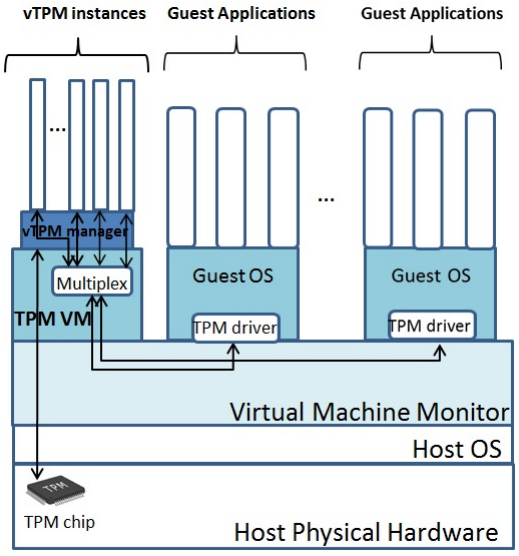


Figure 4.10: Architecture of virtual TPM

TPM storage key, the authors in [56] proposed to use the migration procedure in a trusted channel, which is used to create a secret encryption key related to the TPM on the destination and to the configuration of trusted computing base.

4.3.2 TPM based Software Emulator

As mentioned above, one limitation associated with the security properties of the physical TPM is that it only permits one TPM per platform. Since Berger et al. [52] introduced their vTPM architecture, their works have become a foundation solution for other authors. The TPM based software emulator is another solution to overcome the limitation of TPM. In virtualization environment, each vTPM is implemented as software based emulator that has its own emulated specifications. Because of software form, the solution has less security guarantees in comparison with hardware form. It neither keeps the hardware based root of trust nor supports vTPM migration. Although remaining some drawbacks, this solution is currently reasonable for study purposes. We present here two examples of emulators.

1. TPM emulator: Strasser and Stamer [55] presented an open source TPM emulator. It becomes a new standard for other approaches using software TPM. The solution

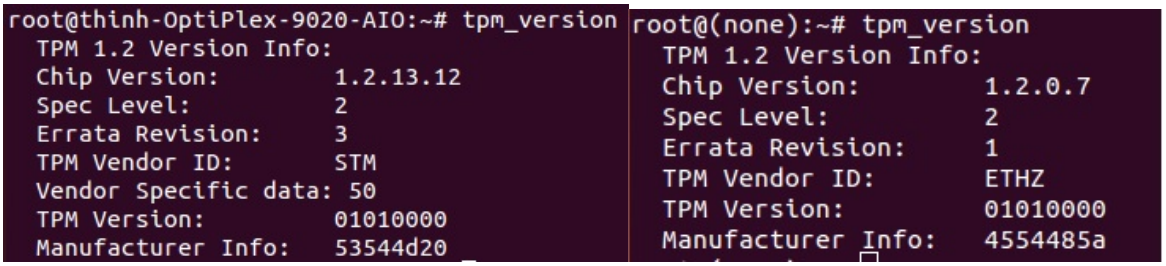
provides the ability to run more than one TPM emulator instance on a single platform and execute TPM based software in a trusted virtualization environment.

2. XEN- embedded TPM emulator: XEN is an open source hypervisor that enables for multiple VMs to run on a single system. It provides a service to support TPM functionality to VMs by embedding a TPM emulator in it. This allows guest application to interact with a vTPM in a same way they interact with a hardware TPM [22]. The figure 4.11 presents the pTPM in host system and vTPM in virtual machine. Each VM has its own software TPM, which is simulated by TPM simulator. vTPM manager domain manages TPM specifications and is responsible for interacting between vTPM and pTPM. The implementation model of vTPM in XEN has the following features[10]: non transparent vTPM, vTPM's secret bound to pTPM, configurable TPM ownership and Storage Root Key authentication, pass through of certain Platform Configuration Register (PCR), extension of Chain of Trust from the host machine to the virtual. However migration of vTPM is not supported. In figure 4.12, we simulate the vTPM based on virtual machines in XEN[11]. This simulator enables each VM to interact with its own vTPM as the same way the host machine works with its unique TPM. There is a special VM, called vTPM manager, that manages a particular vTPM for each VM. As presented in figure 4.12a, the vtpmmgr manages two vTPM, namely domu-vtpm and domu-vtpm02. Finally, within each VM, we can interact with its corresponding vTPM. Figure 4.12b demonstrates the content of PCRs of vTPM in a specific VM.

4.3.3 Virtual Trusted Execution Environment (vTEE)

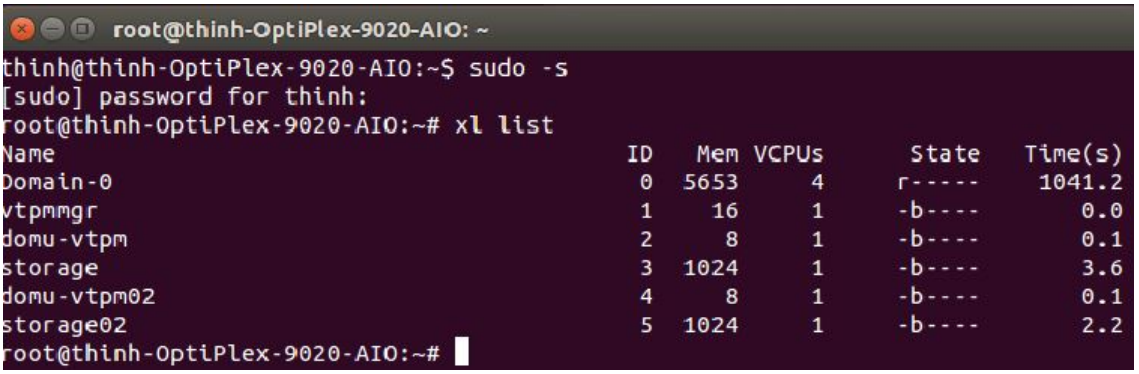
In addition, with the advantage of TEE, TEE[12] has been deployed widely in volumious mobile devices. Due to the limited access created by mobile manufacturers and the cost, hardware based TEE makes the difficulties for programming developers who want to

[10]Virtual Trusted Platform-XEN. http://wiki.xenproject.org/wiki
[11]https://xenbits.xen.org/docs/unstable/man/xen-vtpm.7.html
[12]https://www.globalplatform.org/mediaguidetee.asp

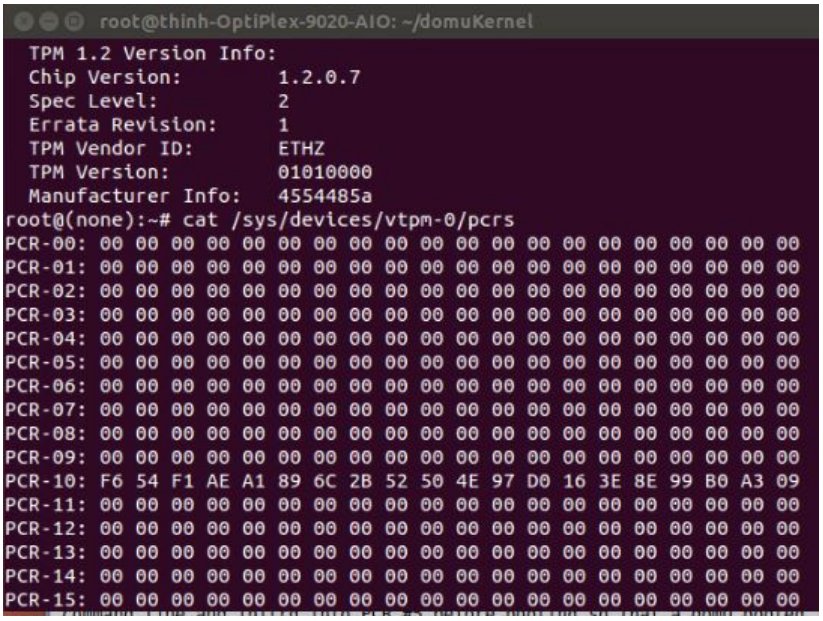


(a) physical TPM (b) virtual TPM

Figure 4.11: Example of vTPM for virtual machine relies on physical TPM's specification



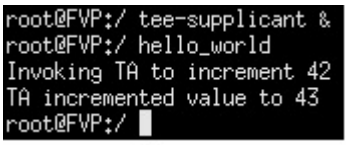
(a)



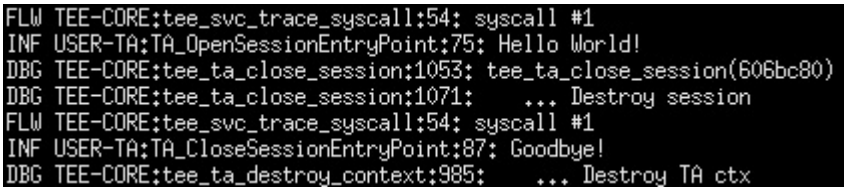
(b)

Figure 4.12: vTPM simulation

interact or access to the TEEs. The authors in [58] have built an Open-TEE or virtual TEE in software to overcome this situation. Figure 4.13 represents an example of vTEE and its trusted application[13]. Where the simple trusted application "hello_world" is executed its sensitive part in trusted environment and its result is displayed in Rich OS (e.g. Android OS, iOS)



(a) In Rich Environment



(b) In Trusted Environment

Figure 4.13: Demonstration of vTEE

4.4 The use of TPM based attestation in mobile devices and cloud computing

The proliferation of mobile devices has changed the requirement of computing ecosystem. The end-users would rather use their personal mobile devices than personal computers for either entertainment or work. However, the mobile device is still considered as the resource-constraint device in terms of energy, storage, processing power, and especially for security. To fulfill the requirement of security, there are many useful tools, antivirus application, and cryptography techniques in use. However, they require a trusted computing platform to run.

Trusted Computing Group (TCG) with its TPM provides a standard that enables authentication, authorization, encryption and integrity to be achieved on a variety of computing platforms. As we discussed earlier in subsection 4.1.5, remote attestation is one of the im-

[13]https://wiki.linaro.org/WorkingGroups/Security/OP-TEE

portant functionalities provided by TPM, namely Binary Attestation (BA). Many authors have used the advantage of BA to prove the trustworthiness of their research objects such as platforms, applications, and agents [59; 60; 61]. In other hand, some researchers argue that this attestation mechanism has its own shortcomings, such as privacy, flexibility and scalability [46; 62; 63; 64]. For this reason, the literature [48; 63; 65] proposed Property-Based Attestation (PBA) which is established on Binary Attestation. This mechanism attests not only binary values but also security properties, functions or behavior of systems to overcome the limitation of BA. Based on this notation, Xin et al. [66] proposed a model of PBA oriented to cloud computing that enables users to attest the security property of cloud service platform before exchanging data or performing tasks to the cloud. Another approach is to combine the certificate of Attestation Identity Key of TPM, which is delivered by Privacy Certificate Authorization and Secure Socket Layer certificate to form Platform Property Certificate as presented in [67]. To consider the problems of trust in cloud monitoring system, Awad et al. [68] introduced a trusted framework to establish a chain of trust for the clients in the cloud environment by relying on PBA. In addition, to improve the flexibility, Liu et al. [69] with their model, CORA, allow the cloud tenants not only choose the node in cloud that matches to their own security requirements but also verify the trustworthiness of this node dynamically.

To improve the security model for virtual machines and services in the cloud, Vijay et al. [70] brought in a new trust model to detect and prevent security attacks in cloud environment by using PBA. In their model, the authors considered the basic communication properties between the tenant virtual machine and the cloud user as the security properties for attesting, such as the source address, the traffic and the state validation of the tenant virtual machine.

In the mobile computing context, although without the existence of hardware root of trust, Nauman et al. [71] implemented BA for Android platform. By emulating the TPM as a part of the kernel, the authors created a root-of-trust to establish the chain-of-trust from this root-of-trust to the Dalvik virtual machine, and then to the entire entities within the virtual machine. Similarly, the authors in [72] investigated the practicable of remote attestation for low cost embedded computing devices without trusted hardware. To support

secure remote attestation, they claimed that only the essential and sufficient properties for a low cost device were identified and mapped into a minimal collection of system component. Meanwhile, Kostiainen et al. [73] applied PBA for the in-vehicle communication system enabling mobile devices to exchange data to car head units. The authors presented a new model of property-based attestation that bootstraps from existing mobile application certification infrastructure. In their work, they omitted a trusted party which is responsible for translation between software measurement and properties.

4.5 Conclusion

In this chapter, we study the existing trusted platforms and investigate one of the most important features, that is remote attestation. We also look into the recent work to figure out their pros and cons. Most of them have pointed out the existing drawbacks of current remote attestation mechanisms. Regarding to application attestation, the deficiencies of BA have been shown clearly in many discussed above researches. However, in term of property-based attestation, the questions that what useful properties need to be attested; and the way to generate them automatically have not been answered sufficiently by the above mentioned work. In the next chapter, we address the recent security issues related to *MCC* by reviewing various approaches.

Chapter 5

Security issues in Mobile Cloud Computing: Related Work

MMC not only takes advantage of the cloud computing facilities but also inherits the security threats of conventional cloud computing. Besides the existing challenges facing the processing of providing web service to mobile devices such as loss of connection, bandwidth/latency and limited resources, using the cloud for mobile devices also raises a challenge of security and trust issues. The current challenges of *MCC* can be listed here, namely malware, virus, privacy, financial fraud, content protection, enterprise data and secure space. Due to this matter, there is a rise of awareness and an acceleration of adoption of security solutions. To address the role of Trusted Platforms in *MMC*, we present security concerns for either the mobile devices or the cloud side.

5.1 Securing the mobile devices

In terms of Security for mobile users, the available Trusted Platforms have played their security role well in mobile device environment. In addition, these trusted platforms can work together to meet the security requirements. For example, TEE can serve as a complementary security countermeasure and integrating nucleus for service that depends on partial identities distributed across SEs, such as Active Stickers, Secure Micro SD Cards, UICCs, and Embedded SEs. The TEE combines these into integrated solutions which assure seamless interaction and security of processes that are executed in the periphery of

the respective[1]. The combination of TEE and TPM Mobile is another typical example. Because of the advantage in on-processor technologies such as the TEE combining with the flexibility of the TPM protocols, this means it is possible to implement the TPM as an integrated solution or in firmware. As a result, the TPM Mobile functionality is implemented as a Trusted Application in the TEE.

What's more, mobile apps and data must be considered to secure mobile devices. Firstly, since apps are executed in untrusted mobile devices, trusted platforms are essential to overcome security constraints. Depending on the security level that is required, apps may rely on physical or software trusted platforms. For example, Roland developed, in [74], a prototype that emulates secure elements within Android platforms. Such an open emulator is used for debugging and rapid prototyping of secure element applications. The emulator is a complete Java Card API implemented within Android. Comparing to HCE of Google, the emulated secure element allows security and trust, and is proposed to replace the physical SE only for long-term testing and for showcasing of applications. Secondly, confidentiality and integrity are required to secure data stored on the mobile device and to allow transfer through the Internet. Well-established protocols such as SSL/TLS should be used for data transfer while confidentiality and integrity may be computed by the trusted platforms of the mobile device since they integrate a crypto-processor with trust storage, and implement cryptographic algorithms. Depending on data sensitivity, data may be encrypted or not and may be stored in a distinct storage memory. In fact, if data is sensitive like a PIN code, a generated security key, or a password, it can be stored in the Flash memory or the EEPROM of the trusted platform (SE or TEE). If data size exceeds the memory available in trusted platforms, data can be stored in the mobile device memory that is protected by the trusted platform like in TEE, or even outsourced if the Cloud provides a trusted environment. In case data is not sensitive, it can be stored in the untrusted mobile device or on the Cloud.

[1] https://globalplatform.org/specifications.asp

5.2 Securing the Cloud

With regard to the security of apps and VMs in the cloud environment, the authors, in [75], proposed the concept of a Cloud of Secure Elements (CoSE) where the secure services are hosted by servers rather than by smartphone Secure Elements. They discuss the use of CoSE for mobile payments, and illustrate how an NFC smartphone may be efficiently used as a bridge between an NFC reader and an Internet server of secure microcontroller that hosts EMV applications. The advantage of this solution is to assign dynamically a physical secure element on the cloud for a sensitive application. In cloud computing, the service provider must achieve auditability, for attesting whether security setting of applications has been tampered or not. As stated in [76], auditability can be achieved using remote attestation techniques that require a trusted platform module (TPM). The TPM contains an endorsement private key that uniquely identifies the TPM. At boot time, the host computes its system state (boot sequence, namely the BIOS, the bootloader, and the software implementing the platform) as a sequence of hashes securely stored within the TPM. A remote party can use this sequence of hashes to authenticate and trust the host platform.

In traditional systems, the operating system is linked to the hardware machine regarding the attestation process. However, in a virtualized environment like the clouds, VMs can dynamically migrate from one location to another, making the remote attestation mechanism not sufficient. Santos et al. [77] proposed to secure the hardware layer using hardware TPM, and secure VMs using a secure virtual machine monitors. VM migration is possible if both source and destination servers are trusted. In [78], the authors designed efficient protocols for trust establishment and management. Additionally, Patidar et al. [79] proposed a system in which cloud computing system is combined with trusted computing platform using a TPM. Security mechanisms like authentication, confidentiality and integrity are provided to illustrate the use of TPM. Coupled with the security of apps and VMs, how to secure data is another important aspect in the cloud ecosystem. To this end, many authors propose security mechanisms for Cloud storage. For example, Hsueh et al. [80] proposed hybrid cryptography, and digital signatures to provide security requirements for data storage of mobile phones. Their solution is used to avoid malicious

attackers from illegal access and to share desired information with targeted friends by distinct access rights. Unlike this kind of solution, the hardware-based encryption can take advantage of the TPM and does not require user intervention or impact system performance. For instance, Trusted Computing Group[2] has published a Trusted Storage specification to provide a manageable, enterprise-wide means for implementing full-disk encryption using TPM. This trusted storage notion may be extended to target the storage provided by the data centers of the Cloud.

In terms of digital right management (DRM), Zou et al. [81] proposed Phosphor, a cloud-based mobile DRM scheme with a subscriber identity module (SIM) card in mobile phone, to avoid illegal distribution and piracy of digital contents (like video, audio, e-book, etc.). The SIM card hosts a license state word (LSW) protocol that decrypts each digital content received from the cloud using the decryption key stored on the SIM card. The drawback of this solution is the sharing of a secret key between the Cloud and the SIM card. In other words, the Cloud has to maintain a key for each user

5.3 Authentication

The mobile devices require strong authentication, secure storage and device integrity. That is to say, authentication and access control are more crucial than ever since the cloud services and the cloud storage may be accessible to anyone over the Internet. The TPM can easily provide stronger authentication than username and password. For instance, to improve mobile security, PIN Access or Biometric Lock (eyeprint/ fingerprints/ facial recognition) has been integrated into mobile devices. By running at the backend of TPM 2.0, if the user has no a precise password or unfollow a right verifying process, TPM locks the user and protects sensitive data.

In [82], the authors present a protocol to authenticate a nearby Cloudlet by a mobile device that handles NFC applications. For authentication, the mobile device uses a SE and involves the Mobile Network Operator and a trusted service management that belongs to a Cloud computing.

[2]https://trustedcomputinggroup.org/tcg-storage-architecture-core-specification/

5.4 Data Privacy

Apart from data security, privacy is a means to allow the user having a complete control on his personal data hosted on the Cloud computing. In some MCC frameworks, like in MobiCloud, Huang et al. [83] introduced the notion of Extended Semi-Shadow Image (ESSI) that was not exactly the same as a virtual image because an ESSI could be an exact clone, a partial clone, or merely an image that has extended functions of the physical device. CloneCloud [84] was another MCC platform that uses offloading technique to perform computing on a resourceful server. For this purpose, the authors used device clones, and assumed that the clone VM was by default a trusted one, and mention establishing trust as future work. The drawback of the MobiCloud and CloneCloud platforms, which rely on device clones on the Cloud, is that no privacy means to be provided to either the user to let him choose the personal data to outsource or to the device clone. If a TEE based solution is adopted for example in this context, all the personal data that the user does not want to outsource can be stored in a secure manner by the TEE that has larger memory storage than a SE.

5.5 Summary

There is considerable consensus that the trusted platforms are helpful in cloud security. We addressed the remaining security challenges of *MCC*, in this chapter. In the next chapter, the literature of computation based intelligence is presented to assess the trust and reputation.

Chapter 6

Computational Intelligence: Scenario for measuring Trust and Reputation

In order to calculate the trust based reputation for *MCC*, especially for cloudlet based architecture, different methods with high-ranking accuracy, such as fuzzy logic system (FLS) and bio-inspired intelligence (i.e ant colony optimization) have been involved. In this chapter, we recall the literature and the advantage of ant colony system after presenting the benefits of fuzzy logic system.

6.1 Fuzzy Logic

6.1.1 Introduction

Fuzzy logic system (FLS) is a proved mathematical concept. It has been considered as the mathematical model of vagueness and imprecision which is originally proposed by Professor Lofti Zadeh in 1965. Its basic concept has been represented in many books [85; 86]. Moreover, there are many researchers who have applied this concept to their work to manage the uncertainties, ambiguities and incomplete information associated with the trust measurement in cloud environment [87; 88; 89; 90]. Generally, FLS is a rule based system [91] in which a mathematical model allows to solve difficult simulated problems with variety input/output parameters. Coupled with the linguistic variables, we can mimic human decision making based on the fuzzy control rules. The figure 6.1 demonstrates

the global function of fuzzy inference system (FIS). The FIS begins with the process of converting the input crisp sets into fuzzy sets, which include the membership functions and linguistic values, namely fuzzification. The Fuzzy Inference applies the rule-base. This is essentially the control strategy of the system and presented as a set of *IF-THEN* rules, to generate the fuzzy output. Typically, the rule can be expressed as the following example: $IF\ state(x)\ is\ Good\ Reputation\ (GP)\ THEN\ output(y)\ is\ High\ Level\ (HL)$, where *state(x), output(y)* are linguistic variables while GP and HL are the linguistic values. Finally, the crisp output is derived by mapping the fuzzy output to crisp input in defuzzification [86].

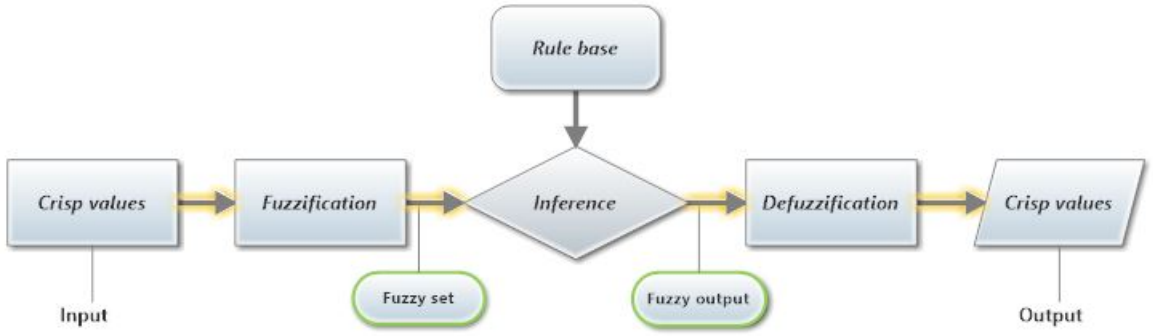


Figure 6.1: General diagram of basic fuzzy logic functions

6.1.1.1 Crisp sets versus Fuzzy sets

In crisp set, everything is bivalence with two values such as True/False, Yes/No or 1/0. The uncertain value is not existed in crisp set. For example,
let $x \in A; A\ can\ be\ \mathbb{N}\ or\ \mathbb{R}$
then

$$\mu(x) = \begin{cases} 1, & (x \in A) \\ 0, & (x \notin A) \end{cases} \tag{6.1}$$

It can be clearly seen that $\mu(x)$ can only receive 1 or 0 value with no allowing middle value. In contrast to the uncertainty problem solving of crisp set, all items in fuzzy sets have different degrees (i.e. multivalence from 0 to 1), called degrees of membership, and sum of degree of membership is 1.

6.1.1.2 Membership functions (MFs)

MFs are used to expressed the degrees of membership. The basic types of membership functions are classified as the following [86]:

- Rectangular Membership Function:

$$\mu(x) = \begin{cases} 1, left \leq x \leq right \\ 0, otherwise \end{cases},$$

where *left, right* are left and right sides value.

For example, $left = 3$ and $right = 5$ (refer figure 6.2).

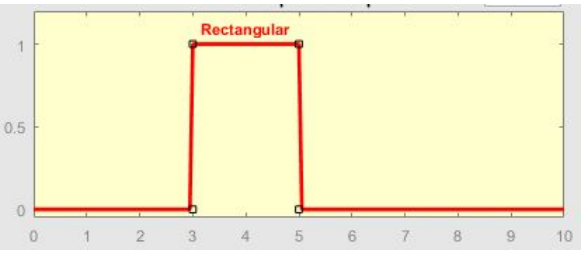


Figure 6.2: Rectangular MF

- Triangular Membership Function:

$$\mu(x) = \begin{cases} 1 - \frac{x - center}{right - center}, & center < x < right \\ 1 - \frac{center - x}{center - left}, & left < x < center \\ 1, & center = x \\ 0, & otherwise \end{cases}$$ (refer figure 6.3)

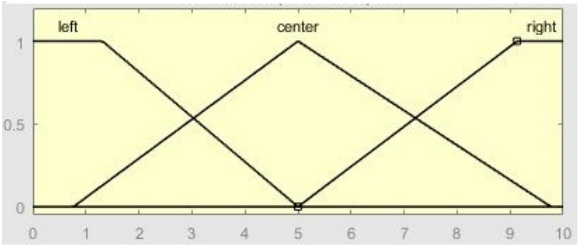


Figure 6.3: Triangular MF

- Gaussian Membership Function:

$\mu(x) = \varepsilon(-\frac{(x-center)^2}{2\theta^2})$,

where ε, θ are the exponential and the width of the membership function, respectively. (refer figure 6.4)

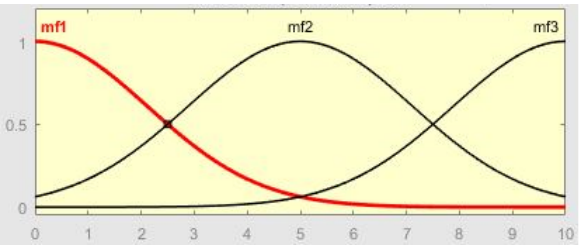


Figure 6.4: Gaussian MF

More precisely, considering the example of evaluation of trust level based on performance and security factors by using triangular MF.
We assume that the linguistic value sets are presented as:

- TrustLevel (Bad, Acceptable, Trust) (Fig. 6.5a)
- Performance (Poor, Medium, Good) (Fig. 6.5c)
- Security (Low, Average, High) (Fig. 6.5b)

Then, the rules are:

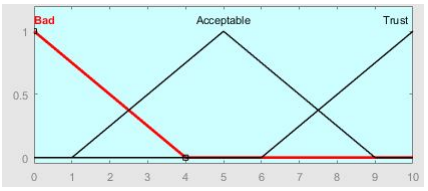
1 If (Performance is Poor) and (Security is Low) then (TrustLevel is Bad)

2 If (Performance is Medium) and (Security is High) then (TrustLevel is Acceptable)

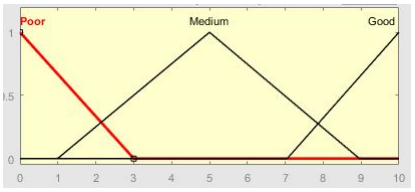
3 If (Performance is Good) and (Security is High) then (TrustLevel is Trust)

4 If (Performance is Good) or (Security is High) then (TrustLevel is Acceptable)

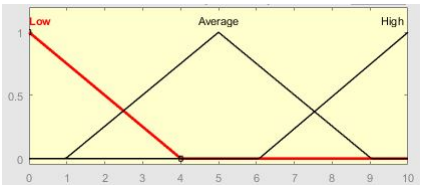
Finally, the figure 6.6 shows the output of fuzzy logic based on the mentioned rules. Taking two value of Security and Performance randomly, as 2 and 2. It can clearly seen that $Trustlevel \approx 1.68$ which is referred as Bad level.



(a) Trust level MF



(b) Performance MF



(c) Security MF

Figure 6.5: Trust evaluation based Performance and Security

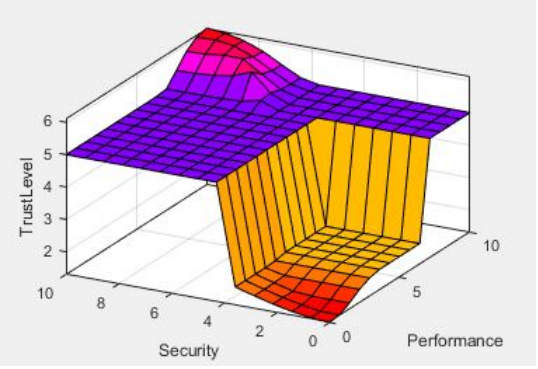


Figure 6.6: Output of trust evaluation

6.1.2 Trust and Reputation using fuzzy theory for Cloud computing: related work

Trust and reputation for cloud computing have recently received an attention of the research community. In this part, we highlight several previous approaches related to trust and reputation evaluation based fuzzy theory. In order to evaluate the security a cloud service provider, the authors, in [92], proposed a fuzzy based approach that allows users to determine the trustworthy of cloud service providers. The trustworthiness of service providers and the willingness of users are analyzed as two important factors for building the trust model. This approach also takes the advantages of fuzzy logic to deal with the uncertainties associated with measuring trust. The authors proved the effectiveness of

their approach by the case study. However, without using real dataset, this approach is less practical to evaluate the trust and reputation. Similarly, Wu et al. [88] present their trust model to handle uncertainty, fuzziness and incomplete information in cloud trust reports. According to the authors, this approach will help clients determine the level of trust that can be placed on any cloud service provider. In grid and peer to peer network environment, Saeed et al. [90] applied fuzzy logic for proposing a reputation model for trust management, namely FR Trust. The objective of this approach is to compute a peer trust level with a linguistic scale, such as Low, Medium or High. The authors consider the chain of trust like: "If A has trust x in B and B has trust y in C, then A must have some trust z in C" to construct their model. Accordingly, nodes are grouped into different categories based on the semantic similarity between their resources. In each group, a super node will be responsible for trust management. Likewise, Fenming et al. [93] developed a cloud model for trust evaluation in peer to peer network system by dividing the node trust into reputation trust and transaction trust. The contribution of this approach is to use a specific tectonic operation to gain the transformation from qualitative concepts to quantitative computation. Regarding the description of the uncertainty trust, the authors have used the entropy (En) and hyper-entropy (He) values. The former is defined as the degree of uncertainty for qualitative concepts. The latter is mentioned about the degree of uncertainty entropy. The following equation, for instance, represents the computation of the trust evaluation of n nodes in the cloud.

$$T(Ex, En, He) = (\omega_1 \times T_1) \oplus (\omega_2 \times T_2) \oplus \cdots \oplus (\omega_n \times T_n) \tag{6.2}$$

where $\omega_1 \ldots \omega_n$ are weight factors of the corresponding trust nodes $T_1(Ex_1, En_1, He_1) \ldots T_n(Ex_n.En_n, He_n)$. The expected value Ex, En and He are calculated as:

$$Ex = \frac{1}{n}\sum_{i=1}^{n}\beta_i\alpha_i \tag{6.3}$$

$$En = \sqrt{\frac{1}{n}\sum_{i=1}^{n}(\beta_i\alpha_i - Ex)^2}$$

$$He = \sqrt{\frac{1}{n-1}\sum_{i=1}^{n}(\beta_i\alpha_i - Ex)^2 - En^2}$$

where $\beta_i \in [0, 1]$ is an interval scale to present the value trust levels in trust space and α_i denotes the time of evaluation between two nodes.
Since reputation based user feedback is considered as subjective testimony, Liu at al. [87] proposed a fuzzy logic based reputation to prevent unfair ratings. To improve the accuracy of the evaluation system, the author considered combining the temporal, similarity and quality aspects of the user ratings based on fuzzy theory into their trust model. The authors divided the involved parties into four different categories. *Trusters* are responsible for evaluating the reputation of entities known as *Trustees*. These *Trustees* provide the ratings which are referred as *Witnesses*. The ratings provided regarding to the entities as *Testimonies*. For example, the reputation (R) of trustee (I) is calculated fairly as: $R = \frac{\sum_{i=1}^{N}}{N}$, where truster U of an online system is assessing the reputation of I with a rating set $\{r_1, \ldots, r_i, \ldots, r_N\}$ and r_i is a numerical integer value.

To extend the existing Certain Trust Model (CTM), Nafi et al. [94] applied the output of CTM with fuzzy logic by proposing two more parameters such as trust T and behavioral probability P. CTM is a decentralized trust model which enables entities to choose trustworthy partners for risky engagements [95]. The objective of this model is to serve as a representation for evidence-based trust and uncertain probabilities [96]. The expectation value of CTM is represented as the following formula:

$$E(t, c, f) = t \times c + (1 - c) \times f \tag{6.4}$$

Where t is an average rating, c and f are certainty and initial expectation, respectively. However, according to [94], this model failed to apply fuzzy logic as well as probabilistic logic because of developing on the basis of human understanding. Thus, to overcome this limitation, the authors proposed that the output of CTM is applied with fuzzy logic along with two new proposed parameters. Such as,

$$Trust_T = \frac{c \times t}{Highest\ value\ of\ rating\ range} \times 100\% \tag{6.5}$$

and

$$P = \frac{T - f}{f} \times 100\% \tag{6.6}$$

Another approach based fuzzy logic for trust evaluation is hierarchical fuzzy inference system for service selection. In this approach, Qu et al. [97] have analyzed past benchmark results to identify services that meet all the quality of service requirements. To evaluate trust of cloud service providers, Trust Evaluation Service has taken user requirements and the services' past performance result as input. The output is a list of trust services. Then, the clients can select the most suitable service for their own purposes. Although bias-rating can be overcome, the drawback of past experience was not mentioned in this approach. For example, new service providers cannot be considered due to lacking of past benchmark values. In this section, we have examined the relevant approaches based fuzzy theory in order to present our proposed model, which is described in Chapter 7, for verifying the trust and reputation of entity in *MCC* in general and cloudlet context in particular. Since our trust and reputation solution combines fuzzy logic and ant colony, we introduce in the next subsection the principle of ant colony optimization.

6.2 Ant Colony Optimization

6.2.1 Introduction

Ant Colony Optimization (ACO) is a one of the swarm intelligent methods. It is considered as a population-based meta-heuristic that can be used to find approximate solutions to solve optimization problems [98]. According to the history of appearance, ACO has been proposed initially by Marco Dongo and his colleagues in 1991, called Ant System. In 1997, the first major improvement has been presented also by Dorigo Gambardella, namely Ant Colony System. Max-Min Ant System, which has been introduced by Stuzle and Hoos [99] in 2000, is another improvement. The best practice for understanding ACO is to solve the traveling sales man problem (TSM). The objective of TSM is to find the shortest path in a closed tour for linking a number of nodes and each node is visited once and only once. The term ACO reflects the original instinct of ant in natural world: the forward ant explores the path randomly to find the food while depositing a chemical substance to create a pheromone trail that allows other ants to follow the same path. The return ants will update the knowledge with their collected information on the paths they traveled. In ACO algorithm, construction of ant solutions and updating pheromones

are two important components. To construct ant solutions, at each node, ants have to determine where to move next according to a probability that relies on two factors such as pheromone trails and heuristic information factors (refer to eq. 6.7). The former factor reflects the past experience of the colony while the latter assesses the interest of selecting a component with respect to an objective function [100].

$$p_{ij}^{k} = \frac{[\tau_{ij}]^{\alpha}[\eta_{ij}]^{\beta}}{\sum_{l \in N_i^k}[\tau_{ij}]^{\alpha}[\eta_{il}]^{\beta}}, if\ j \in N_i^k \tag{6.7}$$

Where p is the probability that ant k in node i move to node j after taking decision that the node with the largest probability may not be chosen [101]; τ_{ij} represents the amount of pheromone trails on the path between i and j; $\eta = \frac{1}{d_{ij}}$, where d_{ij} presents the distance between i and j, indicates the heuristic value. β and α are the weight factors that control the relative importance of trail against heuristic function and N_i^k is the feasible neighborhood of ant k which refers to the set of non-visited city. $[\tau_{ij}]^{\alpha}[\eta_{il}]^{\beta}$ is the maximum value when ant k is located at node i and move to node l.

After constructing the closed tour, the pheromone trails have been updated by following the rule: the better the tour, the greater the amount of pheromone that the paths will receive [100]. Thus, it is important to note that the purpose of updating pheromone is to raise the pheromone values associated with good solutions, otherwise reducing these values that associated with bad ones [98]. Another point is that the pheromone trails gradually evaporate over time, however the resilience of these trails amasses due to multiple ants using tour. The equation 6.8 represents the local update of pheromone which is applied immediately after passing the path between node i and j of ant k during the tour construction [102].

$$\tau_{ij} = (1 - \rho) \times \tau_{ij} + \rho \times \tau_0 \tag{6.8}$$

Where ρ, $0 < \rho < 1$ is the rate of pheromone evaporation and τ_0 denotes as the initial value for the pheromone trails. After each iteration and evaporation is applied to all paths, the shortest path is determined by the elite ants. Only these ants can update the trails to increase pheromone value, namely global pheromone trail update, as presented in equation 6.9.

$$\tau_{ij} = (1 - \rho) \times \tau_{ij} + \Delta\tau_{ij}^{kbest} \tag{6.9}$$

Where, $\Delta\tau_{ij}^{kbest}$ is the total amount of pheromone which the elite ants deposited on the best route. To simulate ACO algorithm and understand how it works, we have used Wolfram CDF[1] as presented in figure 6.7. We assume to use 20 cities and 25 ants with $\rho = 0.98$ and α, β equal to 1 and 3, respectively. Then, the ants travel around randomly to look for a source of food (see figure 6.8a). As the increasing of pheromone on the trails over time, the shortest path is determined as demonstrated in figure 6.8b.

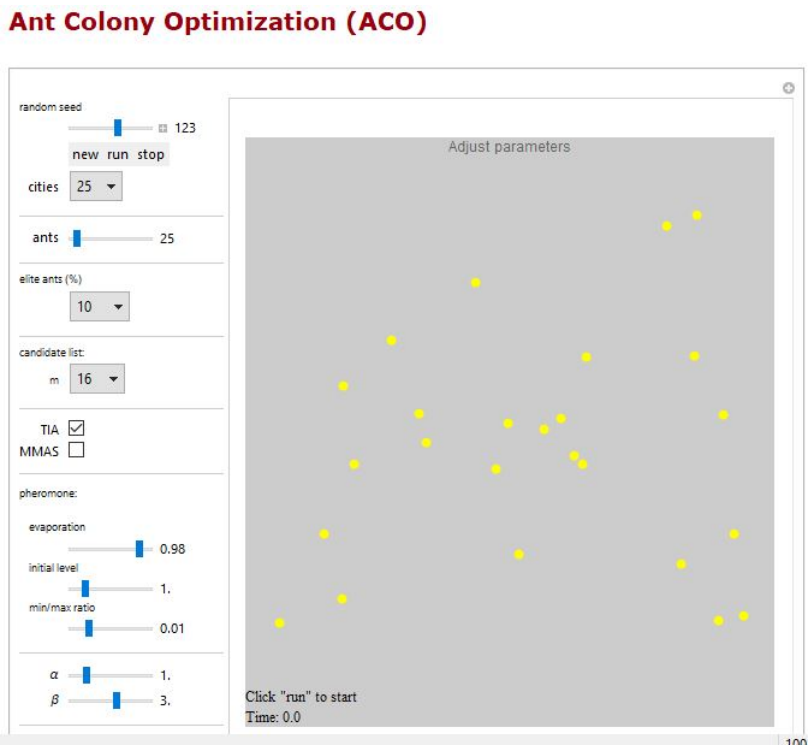
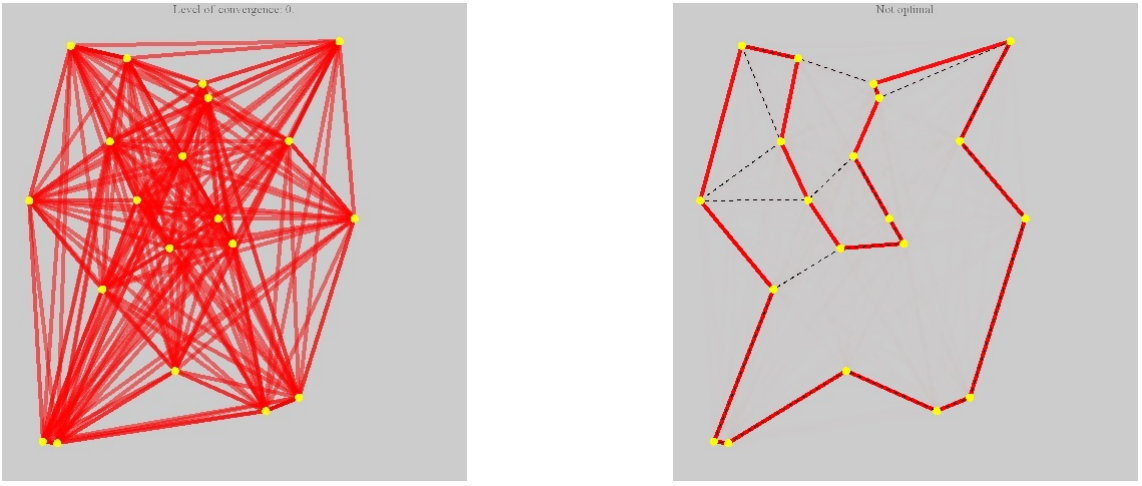


Figure 6.7: Wolfram CDF simulator

6.2.2 Trust models with ACO: related work

Since bio-inspired computing has become important for both the industrial environment as well as the scientific world, the inspiration from swarm intelligence has obtained some highly successful optimization algorithms. In this section, some of the primary approaches related to bio-inspired computing have been presented. Amir et al. [102] introduced the

[1] http://www.wolfram.com/



(a) (b)

Figure 6.8: Process of ACO for finding the shortest path

hybrid approach ACS-TSP combines with fuzzy logic to improve the performance of the algorithms and lead to the faster convergence to the optimal solutions. According to the authors, the ACO's parameter had fixed values throughout the entire run. Thus, they proposed the approach to set these values automatically and also change over the run according to certain performance measures. In this approach, the input values of fuzzy mechanism are the error and variance which are the performance measures. The output values are two important parameters of ACO algorithm, namely β and probability $q_0 \in [0, 1]$. The former refers to the weight of the heuristic information to the pheromone trail. The latter denotes the parameter that controls the exploration against the exploitation in the decision of ants. For example, the fuzzy rule can be expressed as: "IF error is low and variance is medium THEN β is Low". As a result, the modified ACO algorithm has become more flexible than the basic one. By the same token, Castillo et al. [100] presented the approach for dynamic fuzzy logic parameter tuning in ACO. By tuning the α parameter, the objective of this approach is to avoid or slow down full convergence through the dynamic variation of a parameter. In this approach, instead of taking the value of α equal to 1 as many approaches did, the α value is the output of fuzzy control, which the input are the error and change of error. Moreover, fuzzy control can be considered as the translation of external performance specification and observations of a plant behavior into a rule based

linguistic system. For example the rule can be expressed as: "IF (error is P) and (change of error is P) THEN (α increment is N)".

In social network context, Saied et al. [103] have used an alternative ACO algorithm, called the inverted ACO (IACO) [104], instead of the basic one to find trust path between the customers and service provider as well as enhancing load balancing among the customers. Generally, in the IACO, the pheromone scent creates a repulsion effect for other ants to avoid the tour instead of using attraction effect to improve the pheromone value. The IACO algorithm can slip into two phases. Firstly, the selection operation is established among the involved users and the user with higher ρ will be selected. Then, after updating pheromone, the ants in the next interaction will travel in a new route. For instance, if ant k satisfied the user's requirement at time t and then ants in the next interaction (t+1) will move to other service providers. It leads to reduce the waiting time of other users and also improve the load balancing of service providers. Likewise, Shi et al. [105] proposed an ACO-Based Trust Inference algorithm to find trust chain between two nodes in social networks. In this approach, direct trust and trust chain are used to calculate indirect trust value between two nodes. The authors calculated the inferred trust (T_{is}) between node i and s based on the threshold T_{MIN} and the trust value (T_{ij}) of node i and j from the adapted equation [106]:

$$T_{is} = \frac{\sum_{j \in N(i), T_{ij} \geq T_{min}} (T_{ij} \times T_{js})}{\sum_{j \in N(i), T_{ij} \geq T_{min}} T_{ij}} \tag{6.10}$$

then they can get the trust value to the destination node.

Biswas et al. [107] have applied ACO to design an ant colony based trust model for evaluating trust values of a mobile entity in mobile ad-hoc network (MANET). According to the authors, the node is considered as a trusted checkpoint that will never be a malicious or selfish node. This node also has low failure, rate of security attacks and high availability, positive reference to other nodes. In order to decide a specific node as trusted or malicious, the former has the increasing deposited pheromone value while the latter receives the less value. The shortest path between source and target nodes is chosen if there is more than one path which consists of multi-trusted nodes and high pheromone values. Fuzzy-based trusted ant routing protocol (FTAR) [108] is another approach in MANET. In this approach, the

authors have used the dropped packet and time-ratio parameters as the input of fuzzy logic to calculate the trust value for distinguishing between trusted and malicious nodes. They assumed that forward ant (fwa) and backward ant (bwa), a small packet with unique sequence number to prevent duplicate packets, are used to explore the tour. However, fwa will be destroyed when it reaches the destination node. Then, bwa is created for sending back to the source. Here, the authors claimed that their approach can avoid the black hole, gray hole or selective attack by selecting trusted node which is the output of fuzzy logic.

6.3 Summary

Since the bio-inspired computing has been attracting, it increased the attention of many researchers because of its good performance and while resolving optimization problems. Hence, there are many effective approaches which will be discussed in the chapter 7 while making a comparison in terms of performance between our proposed approach and the existing ones. In the next chapter, we will introduce our contributions.

Part III

Contribution

As the result of the technical revolution, mobile devices such as smartphone or tablets computer have been upgraded and made a huge impact on the human's life. From the perspective of security, mobile devices are the soft target of hackers to steal the personal data of the users. Thus, the security concerns need to be addressed in order to establish the trust environment. Due to the limitation of resources, however, mobile devices are inadequately equipped with the sophisticated security solutions compare to converting computing devices. In this part of the PhD thesis, firstly, we present, in Chapter 7, the *MCC* models in real case study and then the new attestation schema is proposed to secure the mobile users in BOYD's context as well as mobile cloud environment in Chapter 8. Last but not least, in order to assess the trust and reputation of involved entities in mobile cloud environment, we propose the novel trust based-reputation model through computational intelligence (i.e fuzzy logic and ant colony optimization) as presented in Chapter 9. Finally, we demonstrate the differences between the mentioned mechanism and other contemporary approaches in Chapter 10.

Chapter 7

Mobile Cloud Computing: Real Case Study

7.1 Introduction

Although the spread of Mobile-Cloud model, using cloudlet architecture can be a better way to externalize applications from mobile devices. Hence, in this chapter, we firstly investigate the performance of the cloudlet based architecture over energy efficiency by examining different implementations, such as Open Service Gateway initiative (OSGi), Cloudlet Overlay and Containerization in section 7.2. Secondly, Droplock system, known as the demonstration of Mobile-Cloud model, is presented in section 7.3.

7.2 Energy efficiency in Mobile Cloud Computing Architectures

In the context of MCC, energy efficiency must be considered not only at the cloud side but also at the mobile side. The former is used to solve the existing limitation of the latter by using remote resource provider rather than processing/storing data locally [28]. In terms of saving energy for mobile devices, the externalization of data and applications is viewed as an effective solution. However, this depends on the MCC architecture that is considered. In fact, if the externalization occurs between the mobile device and the cloud, the application or the data offloading may be costly comparatively to a local mobile computation. More precisely, we conducted some experiments by considering different approaches, all of them

based on the Mobile-Cloudlet model: OSGi, overlay and container based solutions, while comparing the energy efficiency among these architectures to pinpoint their merit and demerit.

To experiment on this work, we have used two devices to implement three discussed Cloudlet based architectures. Performance comparisons is conducted on a Local Area Network (LAN) to overcome Internet Wide Area Network (WAN) latencies that are common to all solutions [109], [13].

- Cloudlet side: Linux Ubuntu 14.04 LTS, 64 bits, CORE i7

- Mobile device side: Windows 7/Linux-based OS 32 bits, DUAL CORE.

Elijah project [38] based on overlay notion. As presented earlier in 3.2.3, the application overlay is calculated as the binary diff (VCDIFF RFC3284) using xdelta3 tool[1], between the complete VM disk image and the base VM disk image. The base VM is then deployed to any platform that will serve as a cloudlet. We assume that we have a VM overlay in the mobile device side. In the cloudlet side, a VM instance is rebuilt by this overlay. The mobile device connects to the cloudlet upon detecting its IP address, using the program "synthesis_client" and supplies the VM overlay. When the mobile device sends the VM overlay to the cloudlet, the cloudlet server starts performing the VM synthesis operation.

Open Service Gateway initiative (OSGi). The OSGi framework provides an environment for the modularization of applications into smaller components called "bundles". These bundles can be either executed on the mobile device or on the Cloud/Cloudlet. The OSGi framework needs to be installed in both mobile device and cloud side. The mobile device has a service consumer which is used to handle interaction with mobile apps that import and consume remote service offered by the Cloud/Cloudlet servers. A service provider has been installed on the Cloud/Cloudlet to implement and export services. For the mobile device, the OSGi framework is installed on an emulator with the following

[1]https://www.systutorials.com/docs/linux/man/1-xdelta3/

characteristics:

Device Nexus S (4.0", 480 * 800: hdpi));

Android 2.2 – API level 8;

Ram 343 MB; VM Heap: 32 MB; Internal Storage: 60 MB; Bundle size: 32 kB.

In the cloudlet side, the OSGi framework is hosted in a specific VM.

Docker container. Even though Docker container engine[2] can be installed on Windows operating system, both the mobile device and the cloudlet operated under Linux- based OS to run Docker containers. In Docker technology, the application is isolated within a container that is managed by a container engine within the mobile device. Instead of offloading the entire container to the cloudlet, that can be heavy (600 MB in our experiments) in size and energy consuming, we offload only the layer that hosts the application.

To compare the difference among these discussed cloudlet implementations, we consider

	Program size			Execution time on local system
	OSGi	Elijah	Docker	
Program 1	3.8 kB	4.96 kB	4.96 kB	∼30 s
Program 2	1.22 MB	1.22 MB	1.22 MB	∼30 s
Program 3	7.922 MB	7.923 MB	7.923 MB	∼30 s
Program 4	15.843 MB	15.844 MB	15.844 MB	∼30 s

Table 7.1: Size and execution time of the tested programs

calculating the execution time locally by examining four different size of programs as presented in table 7.1. We then compare these programs under OSGi, Elijah and Docker platforms to measure their execution time. In this test, communication is established via Wi-Fi to access to the cloudlet server. Figure 7.1 presents time in seconds for downloading and launching the overlay for Elijah, OSGi bundle and Docker layer.

Generally, increasing size of program leads to a rise of execution time for all models. Although transferring an overlay via high speed LAN Wi-Fi, however, Elijah takes more time in comparison with the others especially soar to double time in Program 4 which is the heaviest program. The concept of a VM overlay is similar to copy-on-write virtual disk

[2]https://www.docker.com/what-container

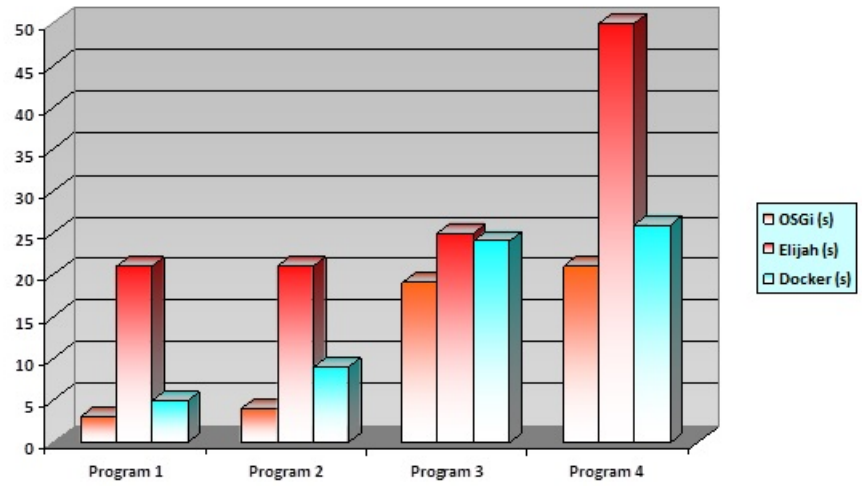


Figure 7.1: Time in second for downloading and executing the overlay, OSGi bundle, and Docker layer

files or VM image hierarchies [38]. It means that the overlay solution is time-consuming task because of the heavy overlay offloading.

On the other hand, with installed Java virtual machine and OSGi framework on each node [109], applications supporting OSGi can interact with a proxy bundle which is generated dynamically with the exported methods without transferring any heavy code. As a result, running a bundle in OSGi framework is the fastest solution in term of time-consuming. The remaining solution, Docker, has insignificant higher execution time than OSGi due to the layer transferring.

Taking the efficiency of offloading for a specific example of energy saving, we consider the following formulas as defined in [110] to apply with our experiments.

- $\frac{M}{S_m}$ denotes the execution time to run the program locally, where M is the amount of computation and S_m is the speed of the mobile device
- $\frac{D}{B} + \frac{M}{S_c}$ refers to the execution time and the offloading time, where D, B and S_c are respectively the size of input data, the bandwidth, and the speed of the remote server.

We consider two cases:

Case#1 $\frac{M}{S_m} > \frac{D}{B} + \frac{M}{S_c}$ if running, a computation takes locally more time than offloading it and running it remotely, then the performance of the whole system is improved by using a specific offloading technique.

Case#2 $\frac{M}{S_m} < \frac{D}{B} + \frac{M}{S_c}$, it means that the offloading does not meet the requirement for computation efficiency.

Adapting these cases to the experimental results, it can be clearly seen that the result of offloading for three architectures is almost positive. Take OSGi based approach for a specific example, the value of $\frac{M}{S_m}$ is always greater than $\frac{D}{B} + \frac{M}{S_c}$ value regardless of the program's size. However, this does not mean that the offload method is always a suitable solution in the context of energy saving. The program 4 in Elijah has proved this drawback of the offloading.

The energy consumed by the mobile device in each MCC model is mainly composed of:

- The computing energy: consumed by the mobile device for execution of local services such as initiation and service request, discovery service, locally executed processes. This energy depends on both the mobile characteristics (computing and battery features) and the performed computing; and
- The communication energy: consumed during the I/O operations using mobile network. It depends on the mobile characteristics, the network characteristics and the size of data to be transferred.

The local computing energy is calculated as in equation (e.q 7.1)

$$ComputingEnergy_{Service} = Mobile_{Capacity} \times Time_{Service} \tag{7.1}$$

Where:

- $ComputingEnergy_{Service}$: is the local energy consumed by a service S (in Joule or kWh).
- $Mobile_{Capacity}$: is the battery capacity of the mobile device (in milliampere or Joule).
- $Time_{Service}$: is the time of the service execution.

Nexus S $(4.0", 480 * 800 : hdpi)$ is used for our experimentation with Android 2.2 system. The energy capacity of this device is 19152 Joules (1440 mAh). Table 7.2 shows the consumed energy measured for each tested solution based on program 1. As shown in

Solution	Execution Time (s)	Mobile Capacity (mAH)	Consumed Energy (Joules)
OSGi	30	1440	0.1596
Elijah	79	1440	0.42028
Docker	30	1440	0.1596

Table 7.2: Consumed energy of the execution of program 1

figure 7.2, increasing the time of program execution (i.e program 1) leads to a rise of the consumed energy on the same device. The launching part (bundle or VM start) is less consuming than the program execution part. For the same program, Elijah consumes more energy in comparison with others solutions. For the ubiquitous environment for *MCC*, crossing architectures also need to be considered. It may lead to increase network latency and transmission rate. In addition, the execution time is based on the application size and the result does not always meet the expectation. For a function partitioning [30], application functions could be determined which part is to be offloaded to the remote cloudlet and which part is processed locally on the device. For example, the appearance of application is displayed locally, while offloading heavy computations to the cloudlet. This takes more energy consumption in case of heavy code offloading. Consequently, the offloading technique is not always considered as the effective solution for *MCC*. Indeed, the OSGi based solution that relies on Client/Server interaction without offloading mechanism can confirm that offloading is not always the suitable solution in terms of energy efficiency.

To sum up, we can clearly say that *MCC* aims to allow the mobile user to overcome its heavy works by providing a seamless and rich functionality regardless of the resource limitations of mobile devices. Although *MCC* becomes the future dominant model for mobile applications, the energy consumption is necessary to consider in order to estimate the efficiency of most of approaches in the context of resource-constraints.

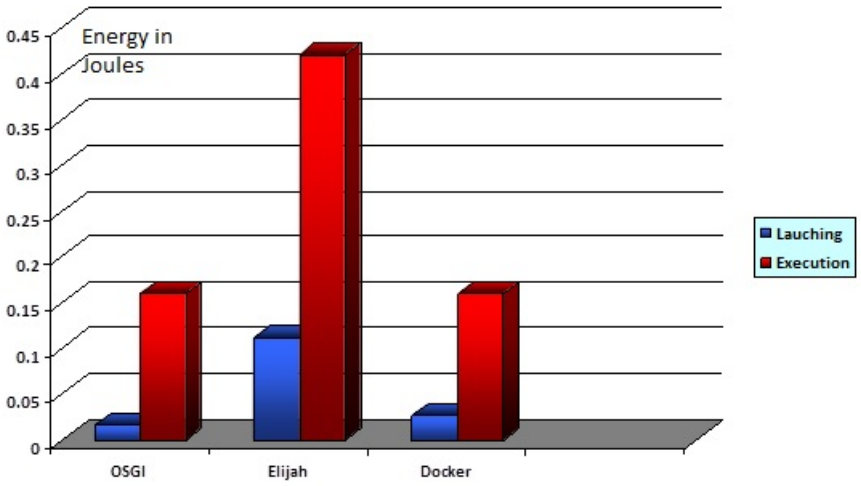


Figure 7.2: Service energy consumption

7.3 Demonstration of Mobile Cloud Computing model in real context

To illustrate the application of *MCC*, we have considered a use case inspired from Smart City, that is, a mechanism that relies on virtual keys instead of real keys of our daily life. Called DropLock, this framework can not only be applied for the product delivery as presented here, but also helps granting authorized access into anything that is locked and that requires unlocking like buildings, hotels, etc. The goal of Droplock approach [14], which is based on Mobile-Cloud model, is to improve the security of the service delivery from the supplier to the client while illustrating the need of *MCC* and Internet of Thing convergence. Currently, if a user orders a product on Amazon website for example, the delivery man will not deliver the product to the user if the latter is not available at the time the package is delivered. Our system overcomes this problem and enables the user to authorize the delivery man to access to the user mail box to deliver the product through an authentication and an authorization process provided by the Cloud to the mobile devices respectively of the user and the delivery man, in case the user is not at his home to receive the package. Once done, the mobile device of the delivery man can interact with an electronic board on the mailbox that will manage the box access. In the same context, the mailbox can be located at the post office. In this case, it is the delivery man that authorizes

the user to have access to the secure mail box. Figure 7.3 represents the DropLock system architecture.

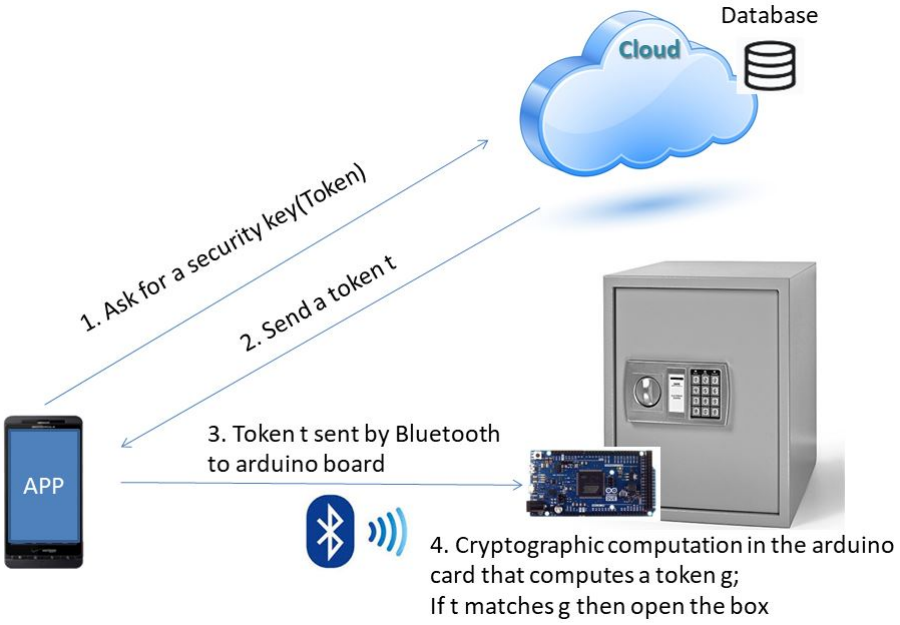


Figure 7.3: Architecture of DropLock system

The overall of this approach has described as the following: first, the user is aware of receiving a product and uses his mobile phone to authorize the delivery office to access to his mailbox, thanks to an android application that interacts with a cloud application. He obtains, by his smartphone, a virtual key from this cloud application. Then, the same Android mobile device makes a Bluetooth connection with the arduino board hosted within the box. As a result, a communication is made between the arduino board and this mobile device. Third, the mobile device presents the virtual key received from the cloud and gives this key to the arduino board. Then, the smartphone sends a notification to the cloud and thus, the cloud informs the delivery man and sends him the key to his smartphone.
When the delivery man needs to access to the secure mailbox to deliver the product, he uses his Android mobile device to connect to arduino board by providing the key. The arduino board compares the key presented by the delivery man, and the one given by the owner of the mailbox. If they are equal, the arduino board unlocks the mailbox.
The delivery man deposits the product, closes the door of the mailbox and informs the

arduino board. The latter locks the door and informs the smartphone of the delivery man. This smartphone sends a message of good delivery to the delivery office cloud, which informs the user of the mailbox. When the user of the mailbox goes home, he can retrieve his product.
This functioning assumes that in the cloud side, the owner of the secure box has previously registered the users/devices which are authorized to access to his mailbox. In fact, to send the virtual key, the cloud checks before if the delivery man/device is allowed to access to the mail box. The One-Time-Password based security solution has been privileged because it avoids replay attacks.
To be more specific, in [14], we have presented the system model to describe the general architecture of any system that relies on machine-to-machine Cloud computing model for smart cities that meets the following requirements:

- Data communication methods.
- Security methods to secure the communications and control the access to the Cloud.
- Resources registration methods that allow the registration of new users within the Cloud system.
- Execution on constrained environments since data processing is partly performed on sensors and mobile devices.

Based on these requirements, as can be seen from figure 7.4, we propose a distributed based machine to machine platform that integrates many services operating at different levels such as network management services, system management services, data management services, security and authentication services.

7.3.1 Network management services

These services offer the applications the suitable means to achieve communication between different smart entities. According to mentioned context, the framework should handle different ranges of communication protocols.

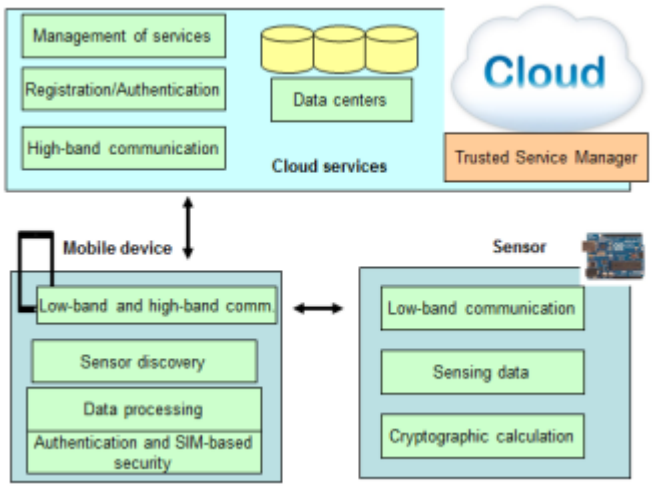


Figure 7.4: Service architecture for Sensor, Mobile and Cloud computing

a. Communication between smart devices: This peer to peer communication focuses on the IoT context. It may occur between the sensors using low-band communication protocols such as ZigBee or Bluetooth or between the mobile device and the sensors using Bluetooth or NFC/RFID technology.

b. Communication between the mobile device and the Cloud: The high-band communication is adopted to allow interaction between the mobile device and the Cloud thanks to wireless networks managed by telecom mobile operators.

c. Sensor discovery: The mobile device may integrate services that allow it to discover neighboring sensors using a peer-to-peer communication.

7.3.2 Data management services

Data can either be sensed by smart devices and processed by the Cloud or sent by the Cloud towards the actuators. Depending on the platform, data has different roles as in the following:

a. Data collection/aggregation: The basic function of a sensor is to sense data from its surrounding environment. Using the data collection/aggregation services, a mobile terminal can collect data from sensors, process and report them using the communication service.

b. Data externalization from the mobile device to the Cloud: Because of their constraints in terms of energy, computational power and memory, smart devices may need to outsource their data storage and computation on the Cloud computing.

c. Data Centers in the Cloud: Data centers are used to store big data collected from multiple sources. They need also to be managed efficiently regarding energy management concerns.

7.3.3 System management services

Because smart devices are constrained environments, they host minimum services with a low footprint. On the contrary, in the Cloud computing context, system management services offer the applications the necessary services that allow task scheduling, load balancing, offloading, migration, virtualization and energy optimization. The system manager relies on distributed infrastructures (IaaS) and platforms (PaaS) to meet Sensing as a Service (SaaS) requirements. In addition, SaaS enables the smart devices such as sensors or mobile devices which may be physically accessible from the Cloud or virtualized so that the Cloud can access to the virtual images of these devices, hence facilitating the management of these devices.

7.3.4 Security and authentication services

The platform should provide security mechanisms in order to ensure trust in the platform and improve data privacy. In fact, unlike the access to the Cloud that can rely on a trusted party (called Trusted Service Manager in figure 7.4), the other platforms such as mobile devices and sensors are generally not trustworthy. However, mobile devices can build trusted environment by relying on hardware trusted platforms as stated in [39], [111] such as SIM cards, Secure Elements (SE), TEE (Trusted Execution Environment). Some sophisticated sensing platforms like Arduino offer specific libraries that implement some cryptographic algorithms, even if they do not have crypto-processors. In our approach, the following steps must be considered:

a. The smart devices need to be authenticated before communicating with any other

platforms (sensor, mobile device, Cloud). For devices with limited resources, the authentication could be implemented as a protocol with lightweight cryptographic algorithms.

b. Data must be encrypted and sent in secure channels.

c. Privacy: Protecting the privacy of user data is the most important factor in Cloud computing. Thus, privacy management related to IoT, mobile devices and Cloud computing should be considered to make it possible for the convergence.

7.4 Summary

In this chapter, we highlight the case study of *MCC* by looking into the energy efficiency and demonstration in real context. Although the concern related to *MCC* is a wide area, we have presented areas that are relevant to the objective of this thesis. In addition, the simple security method (i.e token) in Droplock inspire us to develop a new security solution based software token. Hence, in the next chapter, the novel schema of remote attestation is proposed as the second contribution.

Chapter 8

Property based Token Attestation : A Novel Framework

8.1 Introduction

As cited in Chapter 3, the surge of the presence of personal mobile devices in multi-environment makes a significant attention to the mobile cloud computing. Along with this concern, security issues also appear as a barrier to prevent the propagation of this trend. The goal of this chapter is to focus on an important feature in many security protocols and applications, which is the device attestation in the Mobile Cloud Computing (MCC) as described in Chapter 4. The existing remote attestation mechanisms currently used in trusted computing environment such as Binary Attestation (BA) and Property based Attestation (PBA) are mentioned in Chapter 4. In addition, by taking advantage of the combination of technologies and trends, such as Trusted Platform Module (TPM), Cloud Computing, and Bring Your Own Device (BYOD), we introduce Property based Token Attestation (PTA) to secure the mobile user in the enterprise cloud environment. In order to accomplish a secure MCC environment, security threats need to be studied and acted accordingly, and therefore, we first present again the common threats and then explain a novel attestation schema for addressing these threats by providing security proofs. In addition, Scyther [112] is used to verify the correctness of the proposed protocol.

8.2 Scenario and Threats

8.2.1 Scenario

Bring Your Own Device (BYOD). This term has become a current trend as it combines the benefits of the enterprises and the employees. Besides its advantages, BYOD still remains a dark side probably putting a company's business system at risk. In the BYOD context, we consider our proposed solution to address the existing challenges: 1) The employee with her own device which can be compromised to access restricted business data; 2) The user tries to access data which is not available for her level; 3) The employee quits her job but keeps accessing business data. By overcoming these issues, our work interests a real-world use case. Considering in business context, the employee can use her own phone for personal purposes such as Facebook, Twitter etc.; she can also install a company application for carrying out her work responsibilities. For daily working, depending on her working role and level, she wants to access or receive sensitive business data such as business contract, strategy; these data can be stored locally or globally. In this case, her phone needs to provide its secure evidence before being granted a right to access or receive specific information.

8.2.2 Threats

Although BYOD is quickly becoming the new standard in workplace technology, there are a number of security risks associated with it. Without fully understanding and controlling, the term BOYD can be turned into Bring Your Own Danger/Disaster. In fact, threats on BYOD are quite similar to that of the mobile cloud computing. The only different threat is the user increases the risks by more exposing her devices because of multi-working environments and the enterprise being out of control in the device's using circumstances, such as selling, buying, or maintaining. In this approach, there are only three involved parties in our protocol. We assume that the cloud side, including the Trusted Third Party and the enterprise server, is secure. In addition, vulnerabilities in Cloud are neglected. Before discussing the goal of our proposed solution to address the following listed threats in this subsection, we begin by shortlisting the common threats relevant to

the enterprise mobile user.

1. **Vulnerability**. The mobile device is considered as the resource-constraint device with the limitation of data storage, computation, and energy. So it is not often installed with sophisticated anti-virus application. The mobile data also contains many privacy issues and insufficient management of data encryption techniques when being used both socially and in working environment. For example, with the existence of many data encryption methods, the enterprise faces with a difficult decision to find data encryption strategy that can adapt for all devices and different levels of users. This makes the mobile or tablet prone to attack.

2. **Leakage of data**. There is a risk that software can deliberately/accidently leak data; or a bad user can use a compromised application to obtain legitimate access. In BYOD context, the mobile user is responsible for devices/ software patch updates. It is to increase the risk of data leakage due to buggy/ malicious software or untrustworthy employees.

3. **Miscellaneous data**. The enterprise as well as personal data are often stored in the same local device storage. In some cases, a malware injected without any user awareness could find a way to harm the enterprise data.

4. **Careless using**. There are many attractive mobile applications for daily use. However, these apps are not always free or suitable for mobile platforms. As a result, the user tends to modify her mobile platform by rooting/ jailbreaking it. Root/Jailbreak is not really bad but it is not suitable in the BOYD context where the platform cannot be exposed. In addition, lost and theft are the other risk factors to breach the security of mobile devices.

8.3 Proposed solution

In previous sections, we present the current techniques and the use case for motivating the creation of software token based TPM. In this section, we present our proposed solution, namely *Property based Token Attestation (PTA)*, which takes the advantage of the existing

solutions to adapt the security of mobile device in the Mobile Cloud Computing. Generally, in our solution (See figure 8.1), the Employee (Client M) needs a valid Token provided by the Trusted Third Party (TTP) to access company's resources. To obtain this Token, she has to show her secure evidence of hardware/software platform to TTP, which provides cloud-based security. TTP has a knowledge of its provided services and clients such as Application ID, Property List etc. After evaluating client's evidence, TTP then issues the corresponding Token based on received valid evidence. The Token, in this case, works as *The Letter of Introduction*. Depending on the internal policy of the Enterprise (V), TTP can provide a proof of existence for a token at an instant in time. The token is only valid within T_{TTP} (limited time), otherwise it will be discarded automatically. With this Token, the Employee can access her specific data for a limited time. The Enterprise's role is to evaluate the received Token to enable the employee to access its resource by outsourcing its security service to the Trusted Third Party. Particularly, the Enterprise can delegate its security to a Cloud acting as a Security as a Service platform but it can be a private or a hybrid Cloud as well.

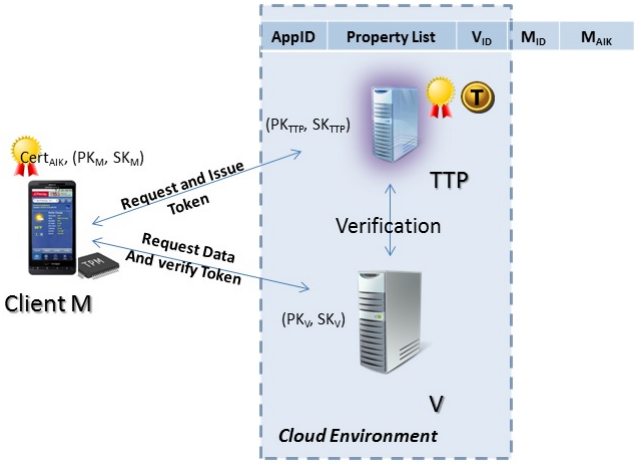


Figure 8.1: Property based Token Attestation

8.3.1 Definition of property and trust token

The property in Property based Attestation (PBA) mechanism is used to describe the behaviour of a platform or program without revealing its configuration. However,

the definition of property to be attested is a broader scope. Thus, everything related to platform or application can be considered as property [46]. In this issue, we attest abstract properties of program's classes to define the security requirements. For instance, the application has three classes: Class A has N properties α where $(\alpha_1, \ldots, \alpha_N) \in \alpha$. Similarly, Class B and C have β and γ respectively. The active TTP has a knowledge of this property collection, namely a property list $(PL) = (\alpha, \beta, \gamma)$. Meanwhile, $\Omega \in (\alpha \cup \beta \cup \gamma)$ is a subset of properties; SP = `{Low, Med, Hi}` is a set of security levels which is defined by the enterprise security policy. Depending on the value of SP, Ω will be generated by the application service. As a result, the random property list Pi is established at run time as $Pi = PL \backslash \Omega$. The integrity of Pi is ensured and verified by a reserved Platform Configuration Register (PCR) and the active TTP respectively. In addition, we use a recomputed Nonce of involved parties using Diffie-Hellman exponentiation function as security property as well. After receiving and verifying the evidences from the client, the TTP signs a trust credential to certify the trustworthy of the specific application. This trust credential is referred to a trust Token for the authentication of the client. The Token consists of the proof, such as *Pi, Timestamp, and recomputed Nonce etc.* to prove the client's trustworthiness to the enterprise.

8.3.2 Assumptions.

In this proposed model, we present the Property Token which contains random properties. The quantity of the property is used to set the policy for accessing or for security level. We assume each application has its own property list (PL). The third party who provides Software/Security as a Service to the Enterprise manages this list. In the client side, the mobile device can generate a list of random properties Pi which is compared to the third party's property list (PL) later. We also assume that all involved parties are equipped with the TPM, especially for the mobile client. This TPM generates a key pair to perform asymmetric cryptography. Before describing the protocol, the following pre-conditions need to be satisfied.

The Enterprise (V)

- V is the enterprise who outsources its security service to the Trusted Third Party

(TTP)

- V and TTP set a security policy for the mobile client (M) based on the Token issued by TTP.
- V grants permission for its employees to access resources if the employees satisfy its requirements.

The Trusted Third Party (TTP)

- TTP is the organization which provides Cloud based security (XaaS) i.e. Software/Security as a Service
- TTP is responsible for checking valid M, either hardware or software.
- TTP has knowledge of its provided services and clients such as *Application ID, Property List, and Certificate of Attestation Identity Key* of M (See Fig. 8.1).
- The Public-Key Cryptosystem is used for securing data transmission between TPP and V only to prevent the compromised M.

The Mobile client (M)

- M is an employee who asks to access V's data or carries out some tasks in extension model, for example offloading or migration.
- Depending on M's role in the enterprise, she can access a specific data only.
- M installs an application securely and logs in with the user ID and password successfully.

The Token (Tk)

- Tk can be considered as a trusted proof for the employee M to access a secure resource.
- Tk is issued to M by TTP if M's hardware/software platform evidences pass the attestation process.

- As discussed earlier, Tk consists of the level of security based on the random property list (Pi). It is assumed, for a specific application, that the software provider (TTP) has a full list of application properties (PL) which can be defined in class levels. The application also has a special service for picking and generating a random property list for the mobile client. For example, as illustrate the figure 8.2, the PL consists of twenty six properties (a-z) of application while a quantity of matched properties (Pi) refers to a level of security as higher number higher level. Let call p a property; M1 and M2 are clients with their legal Pi. Then their integrity is verified successfully. While M3 with its compromised Pi, M3 will be discarded. We have the following attestation conditions.

$$Pi \subset PL \mid Pi \neq 0$$
$$With\, p \in Pi \,\&\, p \in PL$$
$$If\ (\forall p \in (Pi \cup PL))Then\ Pass$$
$$If\ (\forall p \in Pi \,\&\, \exists p \notin PL)Then\ Fail$$

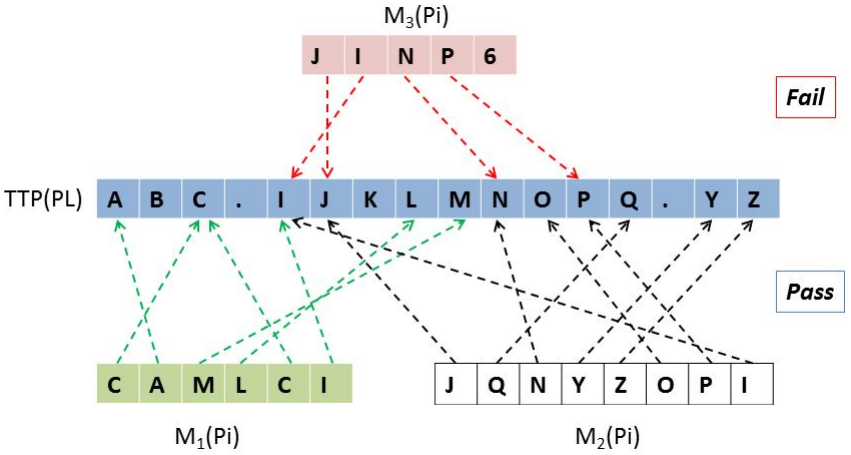


Figure 8.2: Property evidence

8.3.3 System Architecture

In this architecture, there are three involved parties in working out a solution. The TPM is equipped to both mobile device and remote server. The figure 8.3, which is a high level architecture, shows the communication among them. These parties include the Client

who begins the process with her mobile phone by asking indirectly for a property token (PT) from *the Trusted Property Party (TPP); the Enterprise (V)*, which is considered as a resource provider or a verifier. Note that the Certificate of AIK can be obtained easily by either Privacy- Certification Authentication (P-CA) or Direct Anonymous Attestation (DAA)[113], [114], [115]. Hence, in this context, we assume that TPM has its own Certificate of AIK.

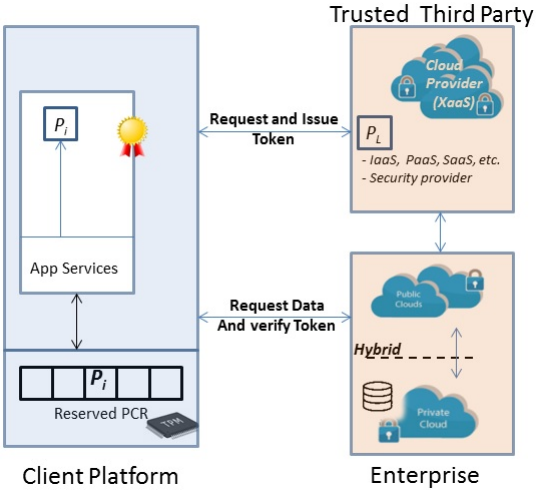


Figure 8.3: Property based Token Attestation model

8.3.4 Property based Token Attestation Protocol.

We start this part by explaining the structure of PTA protocol in high level. After recalling the necessary notations, we present the detail of message sequence charts of PTA protocol in 8.3.5.

8.3.4.1 General Overview of the PTA protocol.

Before discussing the structure of the proposed protocol, we recall the following notations that we use here.

- TTP: is the software provider with a key pair (S_{KTTP}, P_{KTTP}). TTP has a Property List (PL).

- M: is the mobile identified by M_{ID} that has an App installed from TTP with a key pair(S_{KAIK}, P_{KAIK}). This key pair is the result of the remote attestation protocol used to attest the trust of M.

- the Enterprise (V) that has data to be accessed by App of M with a key pair (S_{KV}, P_{KV}).

- V has a list of mobile identities authorized to access to V's data. Hence, M_{ID} must be registered within V.

- App can be any software or specific software of V available from TTP. App is installed with its own properties which represent App's behaviors.

- *Hash (Message)* computes the hash of Message.

- $(Hash(Message)S_{KX}, Message)P_{KX}$ demonstrates the hash is followed with the same data in clear text. The whole encrypted by the public key of X to guarantee the confidentiality of the message.

- $(Message)P_{KX}$ is the Public-Key Cryptosystem for secure data transmission among involved parties.

- X_{ID} denotes the identity of party X.

- T_X refers to the timestamp which is generated by party X to avoid replay attacks

- N_X is the nonce of party X.

- $N_X * N_Y$ is the shared nonce value which is computed by Diffie-Hellman exponentiation [112] between X and Y only.

- ML is the measurement list that contains the measurement value of all entities in the computing platform, for example the measurement of random desired properties in this context. In other words, ML contains the fingerprint list of the entities. ML is used for the integrity measurement with TPM.

Before using the protocol, there are two initial steps. One is related to remote attestation to attest that the mobile device is a trust environment based on TPM and that allows

getting the Certificate of AIK. The other described in the following is related to the way to obtain properties Pi for the app. The process to obtain the Certificate of AIK that is described in detail in Chapter 4 and in [115].

We assume that the user logs in with her Id and Password successfully to V's platform. After verifying online user information (this is the user authentication step) by V, the verified result, which may indicate user's access level, user role, or registered services, will return to the application. Based on this result, the trusted service/ trusted part of application within M will generate a random Pi. The Pi cannot either be affected by application/platform's update or reveal app/platform's configuration. It satisfies the feature of property based attestation.

To prevent a compromised Pi, TTP also has knowledge of user level by exchanging information with V. For example, if the attacker can modify the Pi before TPM's extend operation, TTP will terminate a process due to the invalid quantity or quality (matched properties) of compromised Pi. In case the user loses his Id and password, the attacker with their own devices can obtain the verified result. However, asking for the token to access data is denied thanks to the remote attestation of TPM.

8.3.5 Dataflow of PTA protocol

To depict the general overview of PTA protocol, the message sequence is presented by figure 8.4.

In PTA protocol chart, M has an app that wants to access to V's data according to BYOD concept. Because V delegates its mobile user authentication to TTP, V requests TTP the certificate of M proving its platform attestation. After some verification, TTP sends the certificate to V so that V can discuss with the legitimate M. M requests the token from TTP, by proving that V and M are authentic entities. Then, TTP sends the token to both M and V. Hence, V can check that the token received from M is the same with the one received from TTP, which allows V to grant access to M's app. In *flow 1*, M initiates a message with M_{ID}, nonce N_M and a timestamp T_M to start the communication. The generation of these parameters is achieved thanks to the TPM of M. Only receiver V

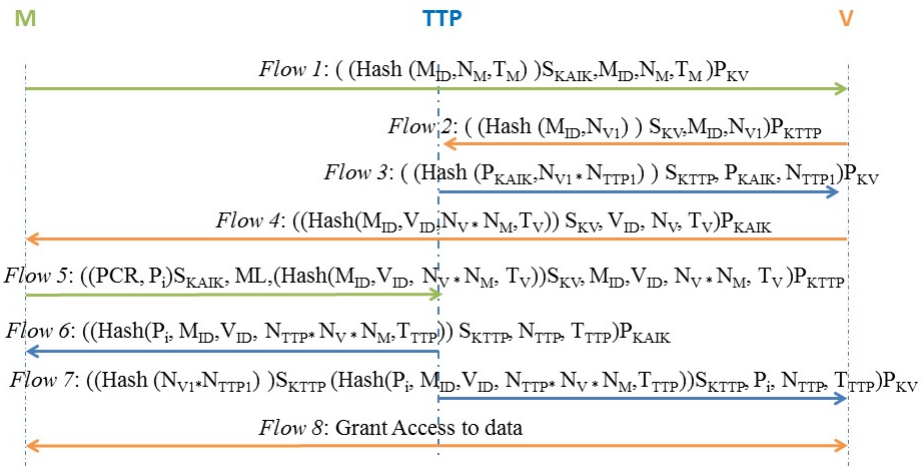


Figure 8.4: Overview message exchange of PTA protocol

can decrypt this message using his private key. However, due to a lack of knowledge of $CERT_{AIK}$, V cannot verify the hash value. V needs this certificate to extract the hash value to compare it with the calculated one and check the identity of M. It is why in *flow 2*, V asks TTP for $CERT_{AIK}$ by sending its own N_{V1} and M_{ID}. According to our assumption, TTP (acting as a Verifier) has obtained this certificate from P-CA or DAA during binary attestation. TTP decrypts the message using its secret key, and checks the integrity of (M_{ID}, N_{V1}). It then uses M_{ID} to find M's certificate $CERT_{AIK}$ that it will send in the next step. In response to V, TTP *flow 3* generates the nonce N_{TTP1} and sends it to V along with the hash value of $N_{V1} * N_{TTP1}$ and the public key AIK of TPM/M that it has. V then decrypts the message to get P_{KAIK}, N_{TTP1}. To avoid the compromised message, V re-computes and compares a new hash value with received hash value such as $Hash(PK_{AIK}, SavedN_{V1} * ReceivedN_{TTP1})$ and $Hash(P_{KAIK}, N_{V1} * N_{TTP1})$ respectively. Now V has the public key AIK of M/TPM. Having a proof of AIK public key, V generates a nonce N_V to work with M and computes N_V and received N_M as $N_V * N_M$. Then, V sends the message as presented in *flow 4* to M. Using its secret AIK key, M obtains the received message and then decrypts the hash value with the public key of V. To ensure the integrity of received message, M computes a shared secret nonce $N_V * N_M$ by using its own N_M and the received N_V. The result of $N_V * N_M$ is hashed together with V_{ID} and T_V to make a comparison with the value of $Hash(M_{ID}, V_{ID}, N_V * N_M, T_V)$. If they match, then M is sure about the identity of V.

In the next step, the mobile user M extends a random Pi to a reserved PCR by calling TPM_Extend() command. Note that the PCR is extended with the random Pi for each attestation. As presented in *flow 5*, the message in previous flow and the ML consisting of the fingerprint of Pi is sent together with an AIK signed value of the reserved PCR generated through the TPM_Quote() command. When TTP receives the message from the mobile user, it first uses the knowledge of AIK to verify and decrypt the first sub-message, $(PCR, Pi)S_{KAIK}$. TTP then compares this PCR value against new PCR value, which is recalculated from the fingerprint of Pi within ML by applying the PCR's extend operation. If they do not match, it implies that the integrity of application is violated. TTP should terminate the communication. In addition, Pi is always used to check if Pi values are coherent compared to PL that is held by TTP as discussed earlier. Finally, the second hash value is used to ensure that the interaction is done effectively with M.

After validating the correctness of M and V's evidence, TTP decides to generate and send a token either to M or V. TTP is sure that M is authentic thanks to the hash values to verify the authentic TPM and M in *flow 2* and *3*, and that the app has consistent properties Pi thanks to *flow 5*. The token is signed by TTP to be sure that he is the issuer of the token. Only TTP has the key to perform this operation. Thus, this trust credential is composed of: $(Hash(Pi, M_{ID}, V_{ID}, N_{TTP} * N_V * N_M, T_{TTP}))S_{KTTP}$ where N_{TTP} is a new nonce of TTP to create a shared secret nonce $N_{TTP} * N_V * N_M$ and the duration of the token is decided by T_{TTP}. The token is only valid within T_{TTP}, otherwise it will be discarded automatically. In *flow 6* and *7*, the token has been delivered to M and V. Having the essential data from previous flows, the receivers check the validity of the received token by re-computing the shared secret nonce $N_{TTP} * N_V * N_M$. For instance, M takes the value of $N_V * N_M$ obtained in *flow 4* and the received N_{TTP} to create the new shared nonce for the comparison.

Finally, the mobile user (*flow 8*) can obtain the right to access to the specific data within the valid duration T_{TTP}. Thanks to the knowledge of this trust token, Enterprise V can manage the access of their employees.

8.3.6 Security Properties Review.

In this part, we discuss briefly the following security properties associated with the messages exchanged for attestation purposes and show how our proposed protocol fulfills them.

- **Confidentiality**: each message m sent from A to B is confidential. The confidentiality is guaranteed thanks to the encryption of m with the B's public key allowing B to be the only entity that can decrypt m because the key used to decrypt the message is secret and held only by B. This property is applied for each flow as shown in the protocol. For example, in *flow 1*, when M sends a request to V, M encrypts with the public key of V to guarantee the confidentiality of the request containing sensitive data.

- **Authenticity of the sender**: to guarantee that sender A is the one who sends effectively data d, A signs d with its secret key. For example, in *flow 2* sent by V to TTP, the $d = (Hash(M_{ID}, N_{V1}))$ is signed by V using its secret key S_{KV} to ensure that only V can be the sender.

- **Data integrity**: each time data d is sent by A to B; d is sent with the hash value of d. This process guarantees data integrity because if clear data d is compromised, the computed hash value of d will be different from the one sent in the flow. For example, in *flow 2*, $d = (M_{ID}, N_{V1})$ is compared with $Hash(M_{ID}, N_{V1})$. Moreover, Hash Extend operation is used throughout the TPM for the integrity measurement. The value extended into PCR can reflect the entity's state. This operation is represented in *flow 5* where the sender extends Pi into specific PCR and the receiver has to re-calculate this value by using the same hash extend operation.

- **Shared secret**: the nonce is used to create a secret that is known to all involved parties. The party outputs data to be stored for later use with the shared secret. When this data is later input, the party verifies the shared secret to detect the validity of data. Take *flow 3* for a specific example, instead of sending a whole value of shared secret $N_{V1} * N_{TTP1}$, a partial value N_{TTP1} is sent from TTP to V. The receiver V

should re-calculate and verify this shared value with its saved N_{V1}.

8.4 Security Analysis

Since V and TTP are assumed to be trust environments, our proof is related only to the behavior of the mobile device M that may host malicious software and/or harmful data.

Proof 1: related to the reception of messages by M. Compromising messages *flow 4* and *flow 6* means that we can have access to the secret key S_{KAIK} to decrypt the message. This can be possible if the key S_{KAIK} is saved within the memory of a malicious mobile device M. But since M has been attested using its TPM (according to our assumption), this means that S_{KAIK} is safely saved in the EEPROM of the TPM. By assumption, TPM cannot undergo physical attacks, that is, TPM is a secure and trust environment. In addition, all the cryptographic operations are performed inside the TPM. In case M is not a legitimate mobile device, M cannot decrypt the messages of *flow 4* and *6* because the malicious M does not have the appropriate secret key S_{KAIK}. This key is saved in the TPM of M and changes each time the remote attestation protocol is used.

1. *Flow 4 from V to M* $((Hash(M_{ID}, V_{ID}, N_V * N_M, T_V))S_{KV}, V_{ID}, N_V, T_V)P_{KAIK}$
   Let the message

   $$D = ((Hash(M_{ID}, V_{ID}, N_V * N_M, T_V))S_{KV}, V_{ID}, N_V, T_V) \tag{8.1}$$

2. Suppose there is a malicious mobile M$'$ who sniffs the message D. The secret key S_{KAIK} of M is stored safely EEPROM of TPM. By our assumption TPM cannot undergo any physical attacks. So, M$'$ cannot get in any way the key S_{KAIK}. It is not possible to derive S_{KAIK} from P_{KAIK}. As in public key cryptography, a public key and a secret key are mathematically related using one-way function, thereby making a unique combination of keys. Now, M$'$ has an inappropriate secret key S'_{KAIK} where $S_{KAIK} \neq S'_{KAIK}$.

3. Now, using the secret key S'_{KAIK}, M$'$ tries to decrypt the message.

   $$d' = [D]S'_{KAIK} = (A', V'_{ID}, N'_V, T'_V) \tag{8.2}$$

$[A' = decrypted\,value\,of\ Hash(M_{ID}, V_{ID}, N_V * N_M, T_V))S_{KV}\ using\ S'_{KAIK}]$

Whereas, if message D is decrypted by S_{KAIK}, then we would get

$$d = [D]S_{KAIK} = (A, V_{ID}, N_V, T_V) \tag{8.3}$$

$[A = decrypted\,value\,of\ Hash(M_{ID}, V_{ID}, N_V * N_M, T_V))S_{KV}\ using\ S_{KAIK}]$

Here as, V is a trusted environment, M′ will not know V_{ID}, N_V, T_V, because D is encrypted by P_{KAIK}, not by the public key P'_{KAIK} of M′. So, $V'_{ID} \neq V_{ID}, N'_V \neq N_V, T'_V \neq T_V$.

To a legitimate M,

$$[A]P_{KV} = hv \tag{8.4}$$

Legitimate M knows M_{ID}, N_M. So, it calculates

$$Hash(M_{ID}, V_{ID}, N_V * N_M, T_V) = hv \tag{8.5}$$

From (8.4) and (8.5), M is sure about the identity of V.

But, to a malicious M′, $[A']P_{KV} = hv'$. Malicious M′ does not know N_M, but it may or may not know valid M_{ID}.

(a) If M′ does not know M_{ID}, then it generates a M'_{ID} and N'_M. Now,

$$Hash(M'_{ID}, V'_{ID}, N'_V * N'_M, T'_V) = hv1' \neq hv' \neq hv \tag{8.6}$$

The reasons are as follows:

- N'_M is a randomly generated number, so it is almost impossible to get $N'_M = N_M$
- It is very difficult to know an input x from h, where $Hash(x) = h$
- Also finding another input z is very difficult where $Hash(z) = h$

(b) Due to the same reasons as above, when M′ knows M_{ID},

$$Hash(M_{ID}, V'_{ID}, N'_V * N'_M, T'_V) = hv2' \neq hv' \neq hv \tag{8.7}$$

As it is not possible by any means for M′ to know *hv* and the inputs used in the Hash function, M′ cannot get access in any way V_{ID}, N_V, T_V, N_M. So, it does not matter whether M′ knows M_{ID} or not; the message used in *flow 4* will not be compromised unless it knows S_{KAIK}.
In case of message in *flow 6* from TTP to M also, M′ cannot know in any way $Pi, V_{ID}, N_{TTP}, N_V, N_M, T_{TTP}$, thanks again to the Hash function. So, the message used in *flow 6* will not be compromised unless it knows S_{AIK}. The same process follows for every secret key of M, changed in each session the remote attestation protocol is used.

Proof 2: man in the middle. By using TPM remote attestation, there are many contemporary works investigated on this TPM feature, in order to prevent man-in-middle (MIM) attack [116], [117], [118]. Similarly, we also take advantage of TPM's benefits to avoid the threat of MIM. Let's assume that, the attacker (MIM) can mimic V and uses V's certificate, $Cert_{AIK}(V)$ in this case, to communicate to either TTP or M. $Cert_{AIK}(V)$ cannot be linked to his own TPM because $Cert_{AIK}(V)$ can only be accessed by V's TPM. As presented in section 4.8, all involved parties must perform remote attestation process to make sure all the parties in a trusted state before establishing the communication. With $Cert_{AIK}(V)$, it is not enough for the attacker to communicate successfully to TTP/M. For example, in step 4 (Figure 4.9), when he sends $(Attacker's\ PCR\ value), SML, Cert_{AIK}(V)$ to the Challenger (TTP/M) to verify, the Challenger (TTP/M) will discard and declare the Attestator (Attacker) is untrusted one due to the non-matching received information. Referring another scenario, the Attacker intercepts from step 1 and can obtain his own $Cert_{AIK}(Attacker)$. He then sends this certificate to the Challenger as in step 4. This case is also impossible for him to prove that he is the owner of V because his information does not appear in the Challenger's database. According to TCG, to avoid the attacked capture and reuse encryption key, the encryption key created by the TPM can only be decrypted by the same TPM. In our proposed work, we assume that App has a key pair (SK_{AIK}, PK_{AIK}) which is generated and protected by the TPM. TTP, M and V have the genuine proof of each other (e.g. evidence of Cert(AIK)) through the remote attestation. To conclude, the proposed work can avoid the threat of regular MIM as the attacker uses

the victim's certificate or his own certificate to communicate with other parities.

Moreover, assume that when M wants to send messages like in *flow 1* and *5*, an attacker R intercepts the message on the network channel and tries to build the same message and send it to V. In this case, the information $M_{ID}, nonce, etc.$ will be generated by the attacker. When V receives this request, it sends the information to the trust entity TTP. In this case, we can have two situations:

- M_{ID} is not known because it is forged by the attacker; hence the attacker request will be rejected.
- M_{ID} is a valid ID, the interaction is pursued between TTP/R and V/R. According to the security level of M_{ID} of R and the properties Pi of R's app, the data access rights are different with those of the legitimate M.

In case the attacker R wants to decrypt the message m to modify m's data, it cannot because since the whole message is encrypted with a public key, only the receiver (V in *flow 1* or TTP in *flow 5*) of the message can decrypt the message and has access to its content.

1. Message (intended to V from M in *Flow 1*)

   $$((Hash(M_{ID}, N_M, T_M))S_{KAIK}, M_{ID}, N_M, T_M)P_{KV} \tag{8.8}$$

   At this stage, public key of M is only available to TTP. V and others do not know P_{KAIK}. Attacker R present in between M and V intercepts the message to make V think that R is the actual sender of the message and to make M think that R is the actual V. So, R now has the message: $((Hash(M_{ID}, N_M, T_M))S_{KAIK}, M_{ID}, N_M, T_M)P_{KV}$. R knows the public key of V but does not know the secret key S_{KV} of V. From *proof 1*, we know that a message encrypted by the public key of the receiver can only be decrypted by the actual receiver using appropriate secret key. Thus, R cannot decrypt the message and therefore it has no access to the contents of the message. Now, R tries to build the same message as received and generates M'_{ID} and N'_M and T'_M. Also, R does not know the secret key S_{KAIK} of M. So, R generates a public key

and private key pair as (P'_{KAIK}, S'_{KAIK}). Now, it tries the build the same message. Hence, the message is now: $((Hash(M'_{ID}, N'_M, T'_M))S'_{KAIK}, M'_{ID}, N'_M, T'_M)P_{KV}$. R sends this message to V.

2. V has no information to check whether it has come from M or not, because V does not know M_{ID}, N_M, T_M. As, V has not $CERT_{AIK}$, V cannot decrypt the hash value. So, V cannot check the identity of R or M. V:

$$\begin{aligned}&[((Hash(M'_{ID}, N'_M, T'_M))S'_{KAIK}, M'_{ID}, N'_M, T'_M)P_{KV}]S_{KV}\\ &= ((Hash(M'_{ID}, N'_M, T'_M))S'_{KAIK}, M'_{ID}, N'_M, T'_M)\end{aligned} \tag{8.9}$$

So, V knows M'_{ID}, N'_M, T'_M. Now, V generates N_{V1}.

3. To know the identity of the sender, V requests certificate $CERT_{AIK}$ to TTP, by sending the message from V to TTP: $((Hash(M'_{ID}, N_{V1}))S_{KV}, M'_{ID}, N_{V1})P_{KTTP}$.

4. TTP:

$$\begin{aligned}&[((Hash(M'_{ID}, N_{V1}))S_{KV}, M'_{ID}, N_{V1})P_{KTTP}]S_{KTTP}\\ &= ((Hash(M'_{ID}, N_{V1}))S_{KV}, M'_{ID}, N_{V1})\end{aligned} \tag{8.10}$$

So, TTP has now M'_{ID}, N_{V1}. TTP also checks the integrity of (M'_{ID}, N_{V1}). To adapt to two aboved situations we have:

Case 1: M'_{ID} is not known to TTP.

TTP: $CERT_{AIK}$ of M'_{ID} is not found. So, TTP rejects the request of V. So, in this *case1*, Man in the Middle attack cannot occur.

Case 2: M'_{ID} is a valid ID and known to TTP.

TTP : $CERT_{AIK}[M'_{ID}]$ found. So, the public key against M'_{ID} which is P'_{KAIK} is obtained.

TTP to V: $((Hash(P'_{KAIK}, N_{V1} * N_{TTP1}))S_{KTTP}, P'_{KAIK}, N_{TTP1})P_{KV}$

5. V:

$$\begin{aligned}&[((Hash(P'_{KAIK}, N_{V1} * N_{TTP1}))S_{KTTP}, P'_{KAIK}, N_{TTP1})P_{KV}]S_{KV}\\ &= ((Hash(P'_{KAIK}, N_{V1} * N_{TTP1}))S_{KTTP}, P'_{KAIK}, N_{TTP1})\end{aligned} \tag{8.11}$$

So, now V has $M'_{ID}, N'_M, T'_M, N_{V1}, P'_{KAIK}, N_{TTP1}$.

6. V to R:$((Hash(M'_{ID}, V_{ID}, N_V * N'_M, T_V))S_{KV}, V_{ID}, N_V, T_V)P'_{KAIK}$

7. R:

$$[((Hash(M'_{ID}, V_{ID}, N_V * N'_M, T_V))S_{KV}, V_{ID}, N_V, T_V)P'_{KAIK}]S'_{KAIK}$$
$$= ((Hash(M'_{ID}, V_{ID}, N_V * N'_M, T_V))S_{KV}, V_{ID}, N_V, T_V) \tag{8.12}$$

because we assume that there is only one correspondence between the public and the private keys, P'_{KAIK} has the correspondence only with private key S'_{KAIK}. We here ignore the fact that multiple public keys can be associated with one private key, because it is a very rare case. So, R now knows V_{ID}, N_V, T_V. But knowing only these values does not fulfil the purpose of R, because it does not yet know original M_{ID}, N_M, T_M. It even does not know P_{KAIK} of M. So, R cannot be able to compromise the message in *flow 1*.

8. Now R poses as V to the original M and therefore sends the following to M:
R to M: $((Hash(M'_{ID}, V_{ID}, N_V * N'_M, T_V))S_{KV}, V_{ID}, N_V, T_V)P'_{KAIK}$
When M tries to decrypt the message using its secret key S_{KAIK}, the combination of S_{KAIK} and P'_{KAIK} will not match, thereby producing a garbage value. But, as M does not know V_{ID}, N_V, T_V, M will not be able to identify whether the message has come from R, after decrypting the message with S_{KAIK}.
M:

$$[((Hash(M'_{ID}, V_{ID}, N_V * N'_M, T_V))S_{KV}, V_{ID}, N_V, T_V)P'_{KAIK}]S_{KAIK}$$
$$= (A', V'_{ID}, N'_V, T'_V),\ where\ A' = [(Hash(M'_{ID}, V_{ID}, N_V * N'_M, T_V))S_{KV}]S_{KAIK} \tag{8.13}$$

Now, M:$[A']P_{KV} = h'$. Again, M calculates the value of Hash function in reverse way with its own M_{ID}, N_M.
M:

$$Hash(M_{ID}, V'_{ID}, N'_V * N_M, T_V) = h'' \neq h' \tag{8.14}$$

Now, M is sure that the message did not come from the original V. So, M discards the message and does not establish connection with V and TTP in this session. Thus,

Man in the Middle attack also could not occur in *case 2*. Therefore, message in *flow 1* is not vulnerable to *Man-in-the-Middle attack*.

9. R: $((PCR, Pi))S_{KAIK}, Pi, ML, (Hash(M_{ID}, V_{ID}, N_V * N_M, T_V))S_{KV}, M_{ID}, V_{ID}, N_V * N_M, T_V)P_{KTTP}$. As, R does not have the secret key S_{KTTP} of TTP, R cannot be able to decrypt the message. Now, R tries to build the message generating M'_{ID} (which may be a valid ID), PCR', ML', Pi', N'_M. R may know V_{ID} (as shown previously). R has its own secret key S'_{KAIK} and may know the value of $(Hash(M'_{ID}, V_{ID}, N_V * N'_M, T_V))S_{KV}$ [from 7 shown previously].

   R: $((PCR', Pi')S'_{KAIK}, Pi', ML', (Hash(M'_{ID}, V_{ID}, N_V * N'_M, T_V))S_{KV}, M'_{ID}, V_{ID}, N_V * N'_M, T_V)P_{KTTP}$.

10. R to TTP: $((PCR', Pi')S'_{KAIK}, Pi', ML', (Hash(M'_{ID}, V_{ID}, N_V * N'_M, T_V))S_{KV}, M'_{ID}, V_{ID}, N_V * N'_M, T_V)P_{KTTP}$.

11. TTP:

$$[((PCR', Pi')S'_{KAIK}, Pi', ML', (Hash(M'_{ID}, V_{ID}, N_V * N'_M, T_V))S_{KV}, M'_{ID}, V_{ID}, N_V * N'_M, T_V)P_{KTTP}]S_{KTTP}$$
$$= ((PCR', Pi')S'_{KAIK}, Pi', ML', (Hash(M'_{ID}, V_{ID}, N_V * N'_M, T_V))S_{KV}, M'_{ID}, V_{ID}, N_V * N'_M, T_V) \quad (8.15)$$

   TTP now has $Pi', ML', M'_{ID}, V_{ID}, N_V * N'_M, T_V$.
   TTP: $[(PCR', Pi')S'_{KAIK}]P'_{KAIK}$

   [$CERT_{AIK}$ tells that M'_{ID} has public key P'_{KAIK} and secret key

$$S'_{KAIK}] = Hash(PCR', Pi') = x1. \quad (8.16)$$

   TTP: Corresponding to M'_{ID}, we assume that the corresponding ML that is stored in TPM is ML1 and from ML1, the obtained PCR value is PCR1. [If R knows a valid M'_{ID}, it is not possible for it to know the corresponding ML and PCR value, that are stored securely in TPM. Also, the property Pi' of R will not match with one in PL corresponding to M'_{ID} that is held by TTP. Let the original property of device

with M'_{ID} is *Pi1*]

TTP:

$$Hash(PCR1, Pi1) = x2 \neq x1. \tag{8.17}$$

Therefore, TTP is sure that the sender of the message is not the intended and original sender, thereby discarding the token request of R. As a result, message in *flow 5* is not vulnerable to *Man-in-the-Middle attack*.

Proof 3: replay attacks. Timestamps T_X in flow messages are used to avoid replay attacks. Consider, for example, *flow 1*. An attacker that intercepts this message m can try to send it again after m is received by V. In this case, V notices that the time has passed and that m has been already initiated at time T_M. Hence, using timestamps avoid replaying the same messages.

1. At time T_M, M generates a message and sends it to V:
   $((Hash(M_{ID}, N_M, T_M))S_{KAIK}, M_{ID}, N_M, T_M)P_{KV}$. Now, an attacker R gets this message.However, R does not have secret key S_{KV} of V, so it cannot decrypt the message.

2. Suppose, V receives the message from M at time T_{M1} due to permissible delay in network. V now decrypts the message.
   V:

$$\begin{aligned}&[((Hash(M_{ID}, N_M, T_M))S_{KAIK}, M_{ID}, N_M, T_M)P_{KV}]S_{KV}\\ &= ((Hash(M_{ID}, N_M, T_M))S_{KAIK}, M_{ID}, N_M, T_M)\end{aligned} \tag{8.18}$$

   So, V now knows N_M, T_M.

3. Now at a different time T_N, R sends the same message to V. That is, at time $T_N, R\,to\,V : ((Hash(M_{ID}, N_M, T_M))S_{KAIK}, M_{ID}, N_M, T_M)P_{KV}$

4. At time T_{N+1}, V receives the message and then it decrypts the message.

V:

$$[((Hash(M_{ID}, N_M, T_M))S_{KAIK}, M_{ID}, N_M, T_M)P_{KV}]S_{KV}$$
$$= ((Hash(M_{ID}, N_M, T_M))S_{KAIK}, M_ID, N_M, T_M) \tag{8.19}$$

At this time also, V gets the duplicate message with N_M, T_M. V checks that $T_M \neq T_{N+1-\delta}$, where δ is a permissible delay in network for propagation of a message from sender to receiver. Also, the nonce N_M is also same in both the messages, which is not possible because nonce is a randomly generated number and two nonces cannot be the same at two different time instants.

Then, V discards this message. So, *Replay Attack* is not possible in this protocol.

Proof 4: related to Pi. A compromised App′ can generate a false set of properties Pi′ sent to TTP in *flow 5*. Then we have two cases:

- The compromised App′ is on the compromised M′. It returns to the *Proof 2*.
- The compromised App′ is on the legitimate M. By checking Pi′ against PL (Property List), TTP terminates a token request due to the invalid quantity or quality (matched properties) of compromised Pi′. This is used to prevent a bad user who does not have a right to access a higher level of data.

Let us assume that a compromised application App′ has false set of properties Pi′.

Case 1: App′ is on compromised M′.

So, M′ has M'_{ID} (which may or may not be a valid ID). M′ may know V_{ID} (as shown previously in Proof 2). M′ has its own secret key S'_{KAIK} and may know the value of $(Hash(M'_{ID}, V_{ID},$
$N_V * N'_M, T_V))S_{KV}$ [from *Proof 2*].

R: $((PCR', Pi')S'_{KAIK}, Pi', ML', (Hash(M'_{ID}, V_{ID}, N_V * N'_M, T_V))S_{KV}, M'_{ID}, V_{ID},$
$N_V * N'_M, T_V)P_{KTTP}$.

Then, R sends this message to TTP. Now, it returns to *proof 2* (for *flow 5*). TTP discards the token request. So, in this case a bad user cannot get access to higher level of data.

App′ is on legitimate M.

So, the property Pi′ of App′ is only the invalid entity here in the message sent from M to TTP.

That is, M to TTP:

$(PCR, Pi')S_{KAIK}, Pi', ML, (Hash(M_{ID}, V_{ID}, N_V * N_M, T_V))S_{KV}, M_{ID}, V_{ID},$
$N_V * N_M, T_V)P_{KTTP}$

TTP:

$$\begin{aligned}&[(PCR, Pi')S_{KAIK}, Pi', ML, (Hash(M_{ID}, V_{ID}, N_V * N_M, T_V))S_{KV}, M_{ID}, V_{ID},\\ &\qquad N_V * N_M, T_V)P_{KTTP}]S_{KTTP}\\ &= ((PCR, Pi')S_{KAIK}, Pi', ML, (Hash(M_{ID}, V_{ID}, N_V * N_M, T_V))S_{KV}, M_{ID}, V_{ID},\\ &\qquad N_V * N_M, T_V)\end{aligned} \tag{8.20}$$

TTP now has $Pi', ML, M_{ID}, V_{ID}, N_V * N_M, T_V$.

TTP: by using Hash extend operation and $CERT_{AIK}$ tells that M has public key P_{KAIK} and secret key S_{KAIK}

$$[(PCR, Pi')S_{KAIK}]P_{KAIK} = Hash(PCR, Pi') = y1 \tag{8.21}$$

TPM has safely stored ML corresponding to M and the property Pi corresponding to valid app in M is also stored in TTP.

TTP:

$$Hash(PCR, Pi) = y2 \neq y1 \tag{8.22}$$

because though the PCR value was the same, $Pi \neq Pi'$. Therefore, TTP discards the token request of M.

Proof 5: related to the duration of the token. To explain the validity duration of the token, assuming that the mobile device M obtains the token from TTP and M has no battery; if the token is stored in the TPM RAM, then after the mobile battery is fully charged up, the mobile device needs to launch the protocol again to get a new token. In case the token is stored in the TPM EEPROM, we check the validity duration to see if it's not expired. If not, the token is used otherwise the protocol is launched to get a new token.

1. After *flow 6*, at time T_{TTP}, TTP sends the token to M:
   $((Hash(Pi, M_{ID}, V_{ID}, N_{TTP} * N_V * N_M, T_{TTP}))S_{KTTP}, N_{TTP}, T_{TTP})P_{KAIK}$.
   TTP stores the token $(Hash(Pi, M_{ID}, V_{ID}, N_{TTP} * N_V * N_M, T_{TTP}))S_{KTTP}$ in its TPM EEPROM along with the timestamp T_{TTP}. We assume that the validity of the token is $\sigma 1$.

2. M:

   $$[((Hash(Pi, M_{ID}, V_{ID}, N_{TTP} * N_V * N_M, T_{TTP}))S_{KTTP}, N_{TTP}, T_{TTP})P_{KAIK}]S_{KAIK}$$
   $$= ((Hash(Pi, M_{ID}, V_{ID}, N_{TTP} * NV * NM, T_{TTP}))S_{KTTP}, N_{TTP}, T_{TTP}) \quad (8.23)$$

   The token that M receives is $(Hash(Pi, M_{ID}, VID, N_{TTP} * N_V * N_M, T_{TTP}))S_{KTTP}$. This is stored in M's RAM. Now, M has no battery.

3. When M is fully charged, it cannot restore the token because RAM is volatile. At time T_X, M launches the protocol again to get a new token from TTP. Launching the protocol means repeating the steps from *flow 1 to flow 5* with different time stamps and different nonce. Now, the public key, private key pair of M has changed to (P_{KAIK1}, S_{KAIK1}) and this pair is stored in TPM EEPROM.

4. We assume that, at time T_Y, TTP received the token request message from M:

   $$[(PCR, Pi)S_{KAIK1}, Pi, ML, (Hash(M_{ID}, V_{ID}, N_{V2} * N_{M2}, T_{V2}))S_{KV}, M_{ID}, V_{ID}, N_{V2} * N_{M2}, T_{V2})P_{KTTP}]S_{KTTP}$$
   $$= ((PCR, Pi)S_{KAIK1}, Pi, ML, (Hash(M_{ID}, V_{ID}, N_{V2} * N_{M2}, T_{V2}))S_{KV}, M_{ID}, V_{ID}, N_{V2} * N_{M2}, T_{V2}) \quad (8.24)$$

   Where N_{V2}, N_{M2} are different Nonce and T_{V2} is one timestamp different from the previous session. TTP now has M_{ID} which is the same as the M_{ID} in the previous session.
   TTP:

   $$[(PCR, Pi)S_{KAIK1}]P_{KAIK1} = Hash(PCR, Pi) = y \quad (8.25)$$

   Hash(PCR, Pi) is the Hash extend operation of PCR in TPM and $CERT_{AIK}$ tells that M_{ID} has public key P_{KAIK1} and secret key S_{KAIK1} at another session. TPM

has safely stored ML corresponding to M and the property Pi corresponding to valid app in M is also stored in TTP.

TTP:

$$Hash(PCR, Pi) = y \tag{8.26}$$

From (8.25) and (8.26), TTP can check the validity and authenticity of M.

5. Now TTP can check in its EEPROM that there was a token generated at time T_{TTP} against M_{ID}.

   (a) If $(TTP : T_Y - T_{TTP} \leq \sigma 1)$, then [TTP to M]:
   $((Hash(Pi, M_{ID}, V_{ID}, N_{TTP} * N_V * N_M, T_{TTP}))S_{KTTP}, N_{TTP}, T_{TTP})P_{KAIK1}$.
   M:

   $$[((Hash(Pi, M_{ID}, V_{ID}, N_{TTP} * N_V * N_M, T_{TTP}))S_{KTTP}, N_{TTP}, T_{TTP})$$
   $$P_{KAIK1}]S_{KAIK1}$$
   $$= (Hash(Pi, M_{ID}, V_{ID}, N_{TTP} * N_V * N_M, T_{TTP}))S_{KTTP}, N_{TTP}, T_{TTP}) \tag{8.27}$$

   So, M now has the same token
   $(Hash(Pi, M_{ID}, V_{ID}, N_{TTP} * N_V * N_M, T_{TTP}))S_{KTTP}$ as previous.

   [The assumption here is that though N_V, N_M are nonces from previous session and T_{TTP} is previous time stamp, there is no possibility of replay attack, because TTP is a trusted environment.]

   (b) If $(TTP : T_Y - T_{TTP} > \sigma 1)$, then [TTP to M]:
   $((Hash(Pi, M_{ID}, V_{ID}, N_{TTP2} * N_{V2} * N_{M2}, T_{TTP2}))S_{KTTP}, N_{TTP2}, T_{TTP2})P_{KAIK1}$.
   M:

   $$[((Hash(Pi, M_{ID}, V_{ID}, N_{TTP2} * N_{V2} * N_{M2}, T_{TTP2}))S_{KTTP}, N_{TTP2}, T_{TTP2})$$
   $$P_{KAIK1}]SK_{AIK1}$$
   $$= (Hash(Pi, M_{ID}, V_{ID}, N_{TTP2} * N_{V2} * N_{M2}, T_{TTP2}))S_{KTTP}, N_{TTP2}, T_{TTP2}) \tag{8.28}$$

   The token that M gets is $Hash(Pi, M_{ID}, V_{ID}, N_{TTP2} * N_{V2} * N_{M2}, T_{TTP2}))S_{KTTP}$, which is different from the previous one.

8.5 Security Discussion

To address the common threats in section 8.2.2, we analyze the security of our proposed protocol while using Scyther tool for verification of security protocol. In this part, we discuss the following security concerns.

1. *Security based hardware.* To lessen the well-known vulnerability with regard to the resource- constraint, each TPM is equipped with a special set of registers (PCRs) and cryptography unit which is a couple of public/private key pair (RSA) at manufacturing time. This can provide security functions like secure storage, attestation of platform state and identity by performing cryptographic algorithms of encrypting, authenticating and attesting data. The mobile users can take advantage of benefits of TPM's functionality, especially the remote attestation, as a virtual anti-virus application (*Threat 1*). Furthermore, the design of TPM is to offer hardware protection for cryptographic keys. It means that the private part of key pair is stored in the Non-Volatile memory of TPM; and never leaves the TPM. As a result, the secret still remains safe even if the system is compromised like Rooting/Jailbreaking (*Threat 4*).

2. *Software Token.* In our work, security and privacy depend mainly on the proposed Token. This token defines the security policy for its owner by either the quantity of random property or matched properties (Pi). It prevents malicious application/user to use a valid token to access enterprise data illegally (*Threats 2* and *3*). To obtain a valid token, the client application has been examined to get its own appropriate confidence level. The user is only permitted to access enterprise data with limited privilege. In addition, depending on the policy of the enterprise, the finite time (T) of token is set for each access session. This turns the token into *One Time Ticket* to access specific data in a specific environment for preventing damage of malware (*Threat 3*).

3. *Attestation based Token Property.* Based on the binary attestation and property based attestation, we proposed the Property based Token Attestation to secure the enterprise cloud mobile device in BOYD context. Differently from [62], [63], we discuss how properties can be evaluated as presented in Chapter 8.3.1. The proposed

attestation mechanism makes a specific application and platform bind a token together to prevent another malicious application with valid token to gain secure data (*Threat 2*).

4. *Scyther.* Scyther has been developed by CAS Cremer [112]. It is a tool for the formal analysis of security protocols under the perfect cryptography assumption. Scyther uses the Dolev-Yao adversarial model [119]. In order to establish the security analysis with Scyther, protocols must be specified in its input language, namely Security Protocol Description Language (SDPL). It is easy for a fresh user to get used to working on Scyther due to the syntax of SDPL being familiar with the existing object oriented languages such as C++/Java. To prevent the attacker to control the network and all the communication, it is assumed that all cryptographic functions are perfect: the attacker learns nothing from an encrypted message unless he knows the decryption key. The tool can be not only used to find problems that arise from the way the protocol is [120] but also automatically find attacks on cryptographic protocols. Once verification is completed, the verification results are represented in OK or Fail status which demonstrates the protocol whether it is correct or false. In our experimental test, the output of the verification is OK. On behalf of the involved parties three roles M, V and TTP were defined within a scope of *P{M, V, TTP}* . We could split the whole protocol into three sub-protocols: *P1{M, V}, P2{V, TTP} and P3{M, TTP}* to simplify the codes and take less verification time. However, to prove a robust protocol, we decide to verify our work in one protocol, though the verification time is quite long.

5. *Repudiation.* Firstly, in our protocol, we have used an asymmetric key used as signing key offers non-repudiation. For example: in flow 4, $((Hash(M_{ID}, V_{ID}, N_V * N_M, T_V))S_{KV}, V_{ID}, N_V, T_V)PK_{AIK}$. It is used to create an asymmetric-key digital signature, the sender (V) applies its private key (SK_V) to a message ($M_{ID}, V_{ID}, N_V * N_M, T_V$). Then, the receiver applies the public key to the signature to recover and to verify the message. The detailed discussion is presented in part 8.3.5. Secondly, how do we trust the public key which is used to sign the message? This is due to the functionality of TPM. In order to prove it is an authentic TPM and platform,

each TPM has its own certificates which are signed by the certificate authority, such as TPM vendor and platform manufacturer. Once again, though remote attestation process, each involved party, which is equipped with TPM, has knowledge of others' TPM evidences. In addition to, the TPM 2.0 specification provides commands to create certificates [9]. In our work, we assume that the TPM is equipped to both client side and remote side in chapter 8.3.3. We also assume that each TPM has its own Certificate of Attestation Identity Key (AIK) to show its identity and to provide its valid evidences with other parties. Thus, we believe that our solution would avoid the repudiation.

6. *Careless using.* The term Careless refers to the context of human activity. The employees compromise their devices for the purpose. For example, they want to download pirated software or access illegally to the prohibited information. Due to the security concern, rootkits/bootkits are completely unsuitable in BOYD context. To overcome this situation, TPM has provided Measured Boot feature, which runs independently with Secure Boot, to record a series of measurements for critical startup related components, including firmware, OS boot component and drivers [121]. The measurement is stored by extending (hash operation) to a particular PCR and the measured value is stored in a Stored Measurement Log (SML). The PCR values and SML are used for later attestation. If either device or TPM has been compromised, the Trusted Third Party/Verifier can detect this device being compromised or not genuine TPM by verifying the PCR value and SML. Similarly, if a careless employee wants to obtain illegally secret information with his compromised application, TTP also can detect this violation by checking integrity of application property under Property based attestation process.

8.6 Conclusion

By introducing the token consisting of random property in this chapter, the proposed security model not only lessens the remaining demerit of remote attestation mechanisms but also adapts to the context-aware. Depending on the context, attestation mechanism has

its own advantages and disadvantages. The proposed model describes the novel attestation schema based on the existing attestation mechanisms. In order to prove the correctness, we have verified the proposed protocol under Scyther and analyzed it with the security proofs. Moreover, the present protocol enables the involved parties, i.e. the enterprise and the trusted third party, to transparently deploy and manage digital tokens for mobile devices not only to accomplish authentication but also grant secure access to enterprise networks.

However, this contribution is considered as the traditional mechanisms that rely on either authenticated identities or clients getting credentials authorized to make certain actions. Most of approaches mainly rely on the cryptography mechanisms [122], [73], [48],[123], such as Public Key Infrastructure. Indeed, the nature of mobile cloud computing cannot be based on static infrastructures, so these traditional mechanisms are insufficient in *MCC* context. Moreover, the traditional security mechanisms are used to resist to external attacks, however it is not that smart to prevent malicious or compromised entities from doing misbehavior. Thus, trust and reputation management need to be reckoned carefully to deal with the mentioned concerns in order to enrich the security and robustness of *MCC*. The next chapter will deal with trust and reputation management in *MCC* environment.

Chapter 9

Trust and Reputation for Mobile Cloud Computing: A Road Map

9.1 Introduction

As discussed in Chapter 3, the cloudlet is considered as a specific example of MCC model. The cloudlet is three-tiered (Mobile-Cloudlet-Cloud) architecture which is placed to serve the proximate users and operates in the same way as a cloud. However, their performance is better [35], [124]. One of the problems that may occur is the management of encryption key management in cloud environment. Several cloud customers may have their own encryption method and managing these keys is another issue to address in the context of encrypted data [11]. By supporting protection of cryptographic keys, random number generation, cryptographically binding data can accomplish certain system configuration, sealing data in the configuration of the application and authentication, through the trusted platform module (TPM)[1]. However, the trusted platform, does not indicate a satisfactory reliable cloudlet because cloudlet security is not only limited to the existence of TPM, the cloudlet can be malicious. In addition, good access to the cloudlet must avoid the uncertainty of network, which persists in the form of network bandwidth, latency and expectation time. Hence, more smart, intelligent and adaptive devices [21] and platforms need to be explored. The evaluation of the degree of trust is significant and it is substantially indicated towards human decisions and perspectives. We argue and introduce possible scope of fuzzy computation for measuring such human linguistics driven contexts of multiple cloudlets. In

[1]https://trustedcomputinggroup.org/tpm-main-specification/

addition to evaluate trusted platforms for cloudlets, it also becomes necessary to emphasize the trusted platforms' services. This is because many application domains with high social and business impact such as personal healthcare, home automation, mobile payment may use mobile devices which rely solely on trust environment. In a dynamic environment, trust has become the major concern of the service providers and the customers. Trust is also an important facilitator for successful business relationships and an important technology adoption determinant [125]. Trust can be supported by hardware, software or even as a service. For example, Trust as a Service (TaaS) should provide a single point for configuring and managing the security of cloud services from multiple providers [126]. The concept of trust can be used to measure an expectation or uncertainty that the entity in the network has other's future actions [127]. As trust is a critical factor, trust evaluation in service also plays an important role in the cloud computing.

9.2 Scenario and Proposed model

In cloud ecosystem, service providers publish and advertise their services as a quality of service (QoS) information, which may not be always true and credible. In fact, these services would provide similar functionalities to the customers, who may select abnormal service with poor quality, less security, or even damaging one. A customer deserves to know what a good service can serve and how to select a good service [128]. Therefore, the challenge still remains and needs to be addressed to help customers to distinguish good services from bad ones. Furthermore, the cloud providers are also responsible for supporting security as important features. This becomes important as the reliability and availability of cloud should be the prime objective. In terms of security and performance, the varieties of users' requirements lead the service provider to optimally balance them. For example, the security and privacy are important factors in enterprise context due to the stored sensitive data; while academic users may require good performance for processing voluminous data [11]. In order to enable customers to select their suitable services, the trust value should be estimated to identify reliable services. Reputation is a subjective assessment of a cloud service to represent public consensus towards a subject's expertise, attitude or reliability in a certain context [129]. Thus, trust based reputation is a mechanism to evaluate an

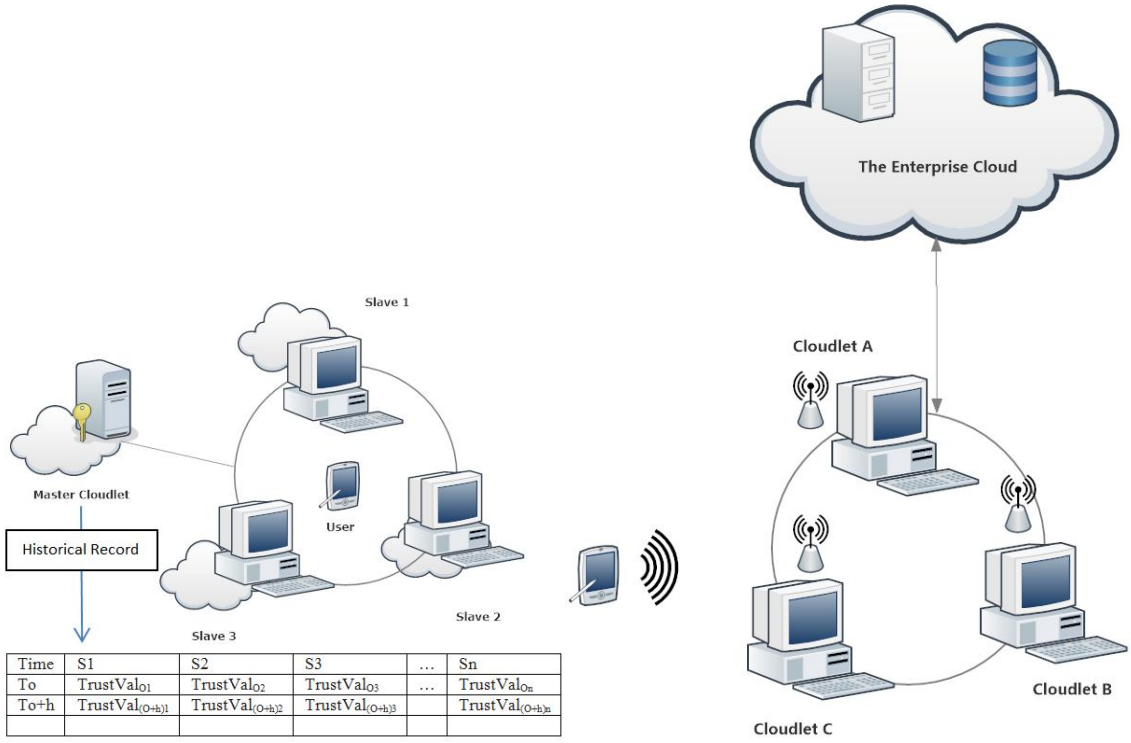
entity's trustworthiness in the cloud environment.

9.2.1 Scenario

As the interesting motivation of cloudlet presented in Chapter 3, considering the scenario in which a mobile user wants to use the cloudlet's services for data processing/downloading externally. His device discovers many available cloudlets nearby. However, these available cloudlets are suitable for his purposes or not; this is the concern of the mobile user in this situation. Thus, the objective of this contribution here is how the mobile user can choose the trusted cloudlet, which is assessed by performance and security parameters, among the cloud mesh to meet his specific requirements.

9.2.2 Proposed model: Trusted based Reputation

In this contribution, we consider a cloudlet mesh that includes cloudlet master to manage its cloudlet slaves as shown in figure 9.1a. The master is responsible for collecting/tracking the related data (e.g. rated score/user's feedback, security and services information) of its slaves to calculate the trust value based reputation. To be specific, this special cloudlet receives the reputation based feedback and also verifies the quality of service information of its cloudlet slaves. In this architecture, for desired agenda the cloudlet processes data, the mobile user must find the reliable cloudlet in the cloudlet mesh. Another key point is that this master cloudlet can be considered as the *Trusted Third Party* (TTP) in the public cloud environment where the trusted Cloudlet does not exist. As such, we can rely on the TTP to replace a role of a master cloudlet. For example (refer figure 9.1b), the employee wants to use a specific cloudlet for processing/downloading his data which can be obtained from the enterprise cloud (EC). In this case, the EC, worked as the master cloudlet, enables the trusted cloudlet access to the EC on behalf of the user by performing the same process as the master cloudlet does. Hence, the EC, which is considered as a trusted cloud, has to verify the trust value of corresponding cloudlets before granting the access right. In general, the proposed concept will be suitable for Mobile- Cloudlet model as well as for Mobile-Cloudlet-Cloud model. To mitigate the complexity, we have used Mobile-Cloudlet architecture to demonstrate for the proposed model. We also assume that



(a) Cloudlet mesh master-slave model (b) Cloudlet mesh and Trusted Third Party

Figure 9.1: Master-Slave and Trusted Third Party model of Cloudlet

all involved parties are equipped with the trusted platform module (TPM). In addition, the ability and quality of each cloudlet provider to support customer services is reflected in its trust and reputation. The service details are stored in a historical record of the master. The reputation of each entity constantly changes due to the genesis of new services or replacing one old service. The historical record stores the latest values and adds the newest values after the ongoing session is terminated. The master uses this record to monitor the change of trust values in order to deliver a suitable recommendation for the customer.

The session of the proposed model starts with a request of the customer, which is sent to the cloudlet mesh to ask for the desired service at time t. The cloudlet master processes the request by tracking the trust values of all available cloudlet slaves at time t. After recalculating the trust value, the befitting cloudlet providers are recommended to the

customer. Finally, a feedback for using service would be left by the customer to calculate the reputation based feedback. Figure 9.2 shows the trust based reputation model in general.

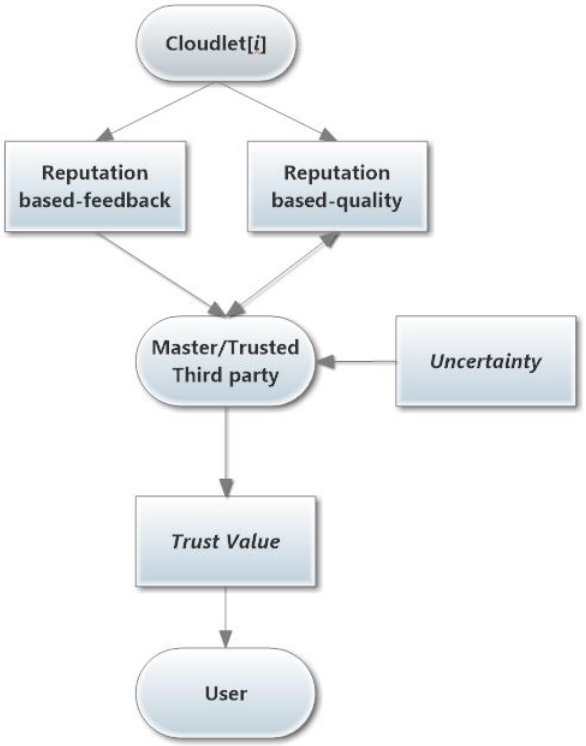


Figure 9.2: Proposed Trust based reputation model

However, the reputation based-feedback is vulnerable to whitewashing, incorrect reported feedback and collusion attacks [130]. In order to gain more accessing or advertisement tariffs, the service providers may exaggerate their QoS information or pay consumers to give them high scores for their services or to leave low scores to their competitors [131]. As a result, the customer cannot distinguish good from poor or worst service providers. The detail of this proposed model is explained in the next section.

9.2.3 Trust based Reputation model

By assuming the context when no prior knowledge about a specific cloudlet is available to make a decision, we propose a model to estimate the trust value of multi-cloudlet by combining the reputation based-feedback, the reputation based-quality and the uncertainty.

This is expressed as in equation 9.1, by using fuzzy inference system. The proposed model is enforced in the master to filter the false reputation of cloudlet slaves. Thus, whatever the cloudlet providers advertise, the master verifies them by following the role: *"a system with a high reputation not for sure to support good service and security. But for providing good service and security, the system must be in high reputation"*. Algorithm I describes the process of trust evaluation model.

$$Trust_{Value} = Reputation_{Feedback} + Reputation_{Quality\ of\ Service} + Uncertainty_{Value} \tag{9.1}$$

The QoS varies according to the application type [124]. In the cloudlet context, we consider two important factors of QoS at runtime, namely, availability and performance. For the security, three prime pillars of security are considered such as confidentiality, integrity and authentication. We also take advantage of TPM functionality such as machine authentication, data protection and remote attestation. Thus, in our solution, we assume that the appearance of TPM is one of the important parameters to calculate the reputation based quality. In the cloudlet mesh, we use n cloudlets, whose set is denoted by $S = \{1..., S_i, ...S_n\}$ and R_S refers to the reputation based-feedback of S in a given time t denoted by $R_{S1}, \ldots, R_{SS}$, where R_{Si} is composed by the correct feedback (cf) and the incorrect feedback (icf) as presented in 9.2

$$R_{Si} = \delta \times (cf) + (1 - \delta) \times (icf) \tag{9.2}$$

Where δ is a reputation weight factor and $0 \leq \delta \leq 1$. In table 9.1, we present a possible degree structure of all involved elements in the process. On the other hand, we use R_{QoS} to present the reputations based on the quality of cloudlet S at time t, denoted by $R_{QoS1} \ldots R_{QoSS}$. R_{QoS} is composed with 3 tuples (Availability (Av), Performance (Per), and Secure Parameters (SP)) as expressed by:

$$R_{QoS} = \omega 1 \times (Av) + \omega 2 \times (Per) + \omega 3 \times (SP) \tag{9.3}$$

where

- $\omega 1$, $\omega 2$, $\omega 3$ are the weight factors and $\omega 1 + \omega 2 + \omega 3 = 1$

- Availability $(Av) = \frac{A_{req}}{N_{req}}$; A_{req} refers to the number of accepted requests; N_{req} is the total number of submitted requests.

- Performance $(Per) = \frac{ts*bs}{max(bs)}$ is the performance of a cloudlet in term of time efficiency, which is calculated by bandwidth and responded time of each cloudlet; denoted by b_S and t_S respectively.

- Security Parameters includes the Confidentiality (C), Integrity (I), Authentication (A) and TPM's existence (T). We require each cloudlet has all security features, like $C \cap I \cap A \cap T$ to work as service provider. Thus, we consider C, I, A and T as a single parameter (SP).

Finally, the trust value ($Trust_{Value}$) of a particular cloudlet slave (S_i) at a given time t is calculated by R_{Si} and R_{QoS} in 9.2 and 9.3 respectively combined with the uncertainty R_U. Where α, β and γ are the weight factors and $\alpha + \beta + \gamma = 1$ as presented in equation 9.4

$$Trust_{Value} = \alpha \times (R_{QoS}) + \beta \times (R_S) + \gamma \times (R_U) \tag{9.4}$$

The result of the trust value is updated to the historical record and the cloudlet with the highest value is recommended to the customer.

Level	Scale	Status	Av & Per	R_{Si}	SP	$Trust_{Value}$
1	< 1	Very Poor (Vp)	✓	✓	✓	✓
2	$0.75 - 1.75$	Poor (Pr)	✓	✓	✓	✓
3	$1.5 - 3.5$	Acceptable (Ac)	✓	✓	✓	✓
4	$3.25 - 4.5$	Satisfactory (Sa)	✓	✓	✓	✓
5	> 4.5	Highly Satisfactory (Hs)	✓	✓	✓	✓

Table 9.1: Degree structure of certainty elements

In the next section, we present how fuzzy logic system (FLS) and bio-inspired intelligence is used to calculate the trust based reputation for cloudlet with reasonably high-ranking accuracy. Generally, FLS is a rule based system [91], based on a mathematical model, that allows solving difficult simulated problems with variety input/output parameters. Coupled with the linguistic variables, we present how to mimic human decision making based on the fuzzy control rules. FLS begins with the process of converting the input crisp sets into fuzzy sets. It includes the membership functions and linguistic values, namely fuzzification.

Algorithm 01: Trust and Reputation Evaluation Algorithm

Input: A set of Cloudlets $S_1, S_2, ..., S_n$ need to evaluate trust level;
A set of quality of service QoS_i for each S_i including Availability (Av), Performance (Per) and Security parameter (SP);
A set of reputation based user feedback $R_{Si} \in (R_{Si}...R_{Sk})$ for S_i;
Uncertainty value R_U for unknown states, $R_U \in [0, 1]$.
Ouput: Trust value of each S_i is recommended to the user.

```
for each S_i ∈ (S_1, S_2, ..., S_n) do
    if S_i is not registered as a cloudlet,
       S_i = False then remove(S_i)
    else
      Compute reputation based feedback (equation 9.2)
      Compute a new R_QoSi with (Av_i, Per_i, SP_i) (equation 9.3)
          if new_R_QoSi < old_R_QoSi then remove(S_i)
          endif
       Let R_U = random() and compute Trust_Value(S_i) (equation 9.4)
       Trust_Value(S_i) = update(Trust_Value(S_i))
    endif
endfor
i=1;
Highest is a value of the previous transaction, stored in the historical record
while (Trust_Value(S_i) in List(Trust_Value(S_i))
    if Trust_Value(S_i) > Highest then Highest = Trust_Value(S_i)
    endif
    i = i + 1
endwhile
Recommend the highest value to the user
Stop
```

The fuzzy inference applies to the rule-base. This is essentially the control strategy of the system and presented as a set of *IF-THEN* rules, to generate the fuzzy values.

9.2.4 Fuzzification

The fuzzfier converts the crisp input data including Av, Per, R_{Si} and SP into fuzzy sets which contain the membership functions (equation 9.5) and linguistic variables (Vp, Pr, Ac, Sa and Hs) as presented in table 9.1 and figure 9.3.

$$\mu(x) = \begin{cases} 1 - \frac{x-c}{r-c} & (c < x < r) \\ 1 & (c{=}x) \\ 1 - \frac{c-x}{c-l} & (l < x < c) \\ 0 & otherwise \end{cases} \tag{9.5}$$

where: x is a value in a range of all possible values considered as fuzzy system input; l, c and r refers to left, center and right value respectively.
For example on x axis in figure 9.3, if we have $(l = 1.5, c = 2.5, r = 3.5)$ then $\mu_{Ac}(x) = 0.5$

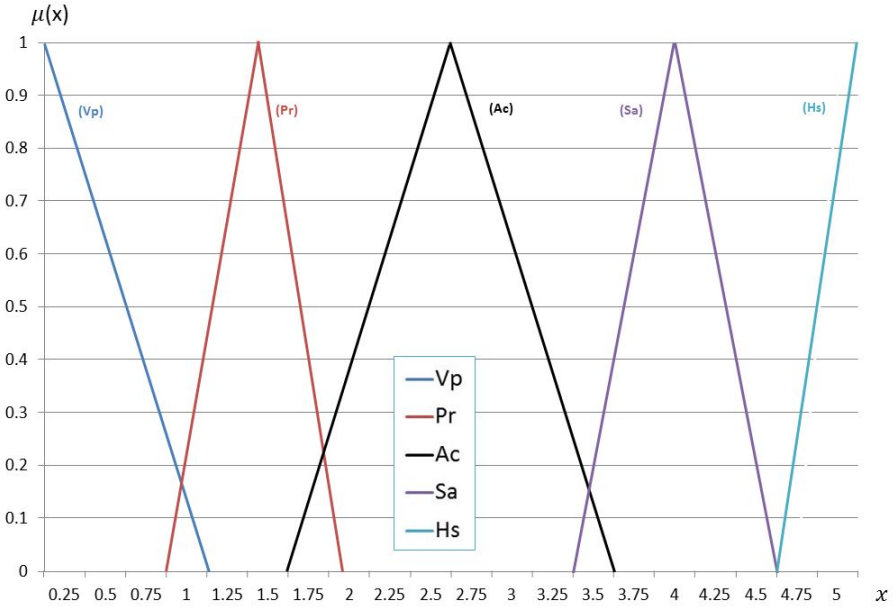


Figure 9.3: Membership input functions of linguistic variables

9.2.4.1 Inference system

In inference rules, we use AND ($\cap$) operator for dependent input parameters, such as Availability (Av), Performance (Per) and OR ($\cup$) operator for the independent parameters, such as security level (SP) and reputation based user feedbacks (R_{Si}) [132], [92] (refer equation 9.6).

$$\mu(x) = \mu_{(SP \cup R_{Si}) \cup (Av \cap Per)}(x) \tag{9.6}$$

If we assume $SP \cup R_S$ is A, Av is B and Per is C then we have

$$\begin{aligned} \mu(x) &= \mu_{A \cup (B \cap C)}(x) \\ &= \mu_{(A \cup B) \cap (A \cup C)}(x) \end{aligned} \tag{9.7}$$

Where again,

$$\mu_{A\cup B}(x) = max(\mu_A(x), \mu_B(x))$$
$$\mu_{A\cup C}(x) = max(\mu_A(x), \mu_C(x)) \tag{9.8}$$

On the other hand: If $A \cup B$ and $A \cup C$ represent X and Y respectively, then
We have,

$$\mu(x) = \mu_{X\cap Y}(x)$$
$$\mu(x) = min(\mu_X(x), \mu_Y(x)) \tag{9.9}$$

For relevant dataset, the reference and details are given in appendix A.

9.2.4.2 Insight to the Fuzzy Rule Configuration

In fuzzy logic, low and high is explicit if their upper value is 1 and lower is 0 and if only if the membership function stands with 0,1. However, it is not necessary that values will lie only in the range of 0 and 1. It can be with any ranges (minimum and maximum, Low and High, Small and Large). Therefore, these are straight linguistics not fuzzy linguistics like extreme, very, moderate etc. In table 9.2, x_d represents part of the whole membership values according to the membership value (refer figure 9.4) while $a, middle$ and b are the initial, middle and terminal final value of triangular membership functions of the linguistic value.

In addition, fuzzy models are used to extract interesting rules from quantitative or linguistic values in relational and transactional databases [133]. The user can subjectively increase the number of rules depending on his experiments. As a result, many uninteresting rules might be generated that leads to the increasing of computation cost, for example, the application needs more time for processing the algorithm due to the complexity and quantity of rules. In fuzzy logic, the aim of the association rule mining is to find all the rules that have the degree of support and confidence greater than the degree of support (min_sup) and confidence (min_conf) determined by the user (i.e. administrator or superuser of cloud/cloudlet). While the form of the association rule is presented as

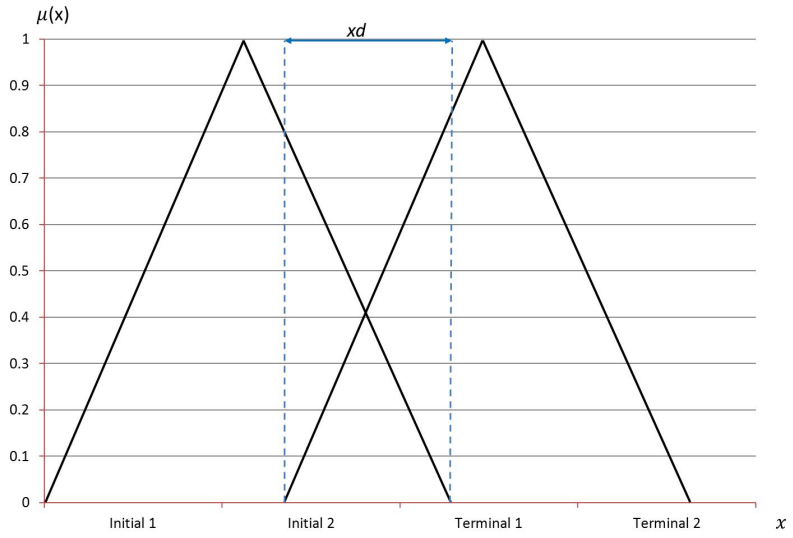


Figure 9.4: Part of membership values for different types of membership range

"If X is A Then Y is B",

the support of association rule refers to the degree to which relationship exists in the database, $support(X \rightarrow Y) = \frac{|n(X \cup Y)|}{|N|}$, and the confidence of association rule is the probability that if X then Y, $confidence(X \rightarrow Y) = \frac{|n(X \cup Y)|}{|n(X)|}$ [133; 134]. Where n(X) is the number of transactions and N is the total of the transaction in database. X and Y are item-sets which contain different attributes. A and B are the fuzzy sets associated with the corresponding attributes in X and Y. For example, considering the pairs of attribute and value as: (Performance, Very Poor), (Security, Poor) or (Availability, Acceptable). Depending on the internal policy of cloud/cloudlet, the particular attributes are considered to generate the rules. Here, we consider, for example, *Time, Property* and $HTTP_{Get}$ are the members of attribute set of a specific cloud/cloudlet. The association rule is presented as the following:

"IF [(Time is Low) and (Property(i) is Small) and ($HTTP_{Get}$(i) is Not Available)] THEN [O(i) is Normal] "

Where O(i) represents the status of the participating clients such as Trust, Not-trust or Safe, Unsafe. Hence, for each fuzzy association rule, the weight ω, which is assigned to

each fuzzy membership for each attribute, can be evaluated as the following equation:

$$\begin{aligned}\omega[i] = [Weight]_i = [Degree\,of\,membership\,for\,Time] \\ *[Degree\,of\,membership\,for\,Property] \\ *[Degree\,of\,membership\,for\,HTTP_{Get}]\end{aligned} \tag{9.10}$$

In addition, the association rules for intrusion detection and their classification can be expressed either by normal, sudden intrusion or attacks. They are inserted to the fuzzy knowledge base. Hence, to fire a particular fuzzy association rule, the following attributes could be augmented:

a. Confidence

b. Support

c. Selection weight of the association rule to find out the rule, that have significant input/output dependency

For example,

$$Confidence_{rule}(A \rightarrow B) = \frac{|\sum_{x_p \in Class(y_p)} \beta_{AB}(x_p)|}{|\sum_{x_p \in Class(y_p)} \beta_A(x_p)|} \tag{9.11}$$

$$Support_{rule}(A \rightarrow B) = \frac{|\sum_{x_p \in Class(y_p)} \beta_{AB}(x_p)|}{|N|} \tag{9.12}$$

Where,

♢ $|N|$ is the number of transactions

♢ $\beta_A(x_p)$ is the compatibility degree of transaction x_p with the antecedent part A

♢ $\beta_{AB}(x_p)$ is the compatibility degree of transaction x_p with the antecedent and consequent of the rule $A \rightarrow B$

♢ y_p represents p^{th} of p consequent.

Membership Functions	Initial value	Bi-selection value	Terminal value
Low	$a_{Low} = 0.0$	$middle_{Low} = (a_{Low} + b_{Low})/2$	$b_{Low} = Low$
High	$a_{High} = b_{Low} - x_d$	$middle_{High} = (a_{High} + b_{High})/2$	$b_{High} = 1.0$
Small	$a_{Small} = 0.0$	$middle_{Small} = (a_{Small} + b_{Small})/2$	$b_{Small} = Small$
Large	$a_{Large} = b_{Small} - x_d$	$middle_{Large} = (a_{Large} + b_{Large})/2$	$b_{Large} = 1.0$

Table 9.2: Configuration of membership values

9.2.5 Discussion

In fact, the proposed fuzzy logic has obtained a noticeable number of achievements in both theory and application to deal with vagueness and uncertainty [93], fuzzy set theory has still remained several drawbacks as listed in [135; 136; 137; 138]. In general, it is tedious to develop the fuzzy rules, which are known as the min-max rule for conjunctive (AND) and disjunctive (OR) reasoning. These rules are arbitrary, and it is difficult to figure out how many rules are adequate to address fully, a specific problem and to make the related robust membership functions. In addition, the fuzzy logic requires a considerable amount of data to formulate fuzzy membership functions. The decisions can also be interpreted in a number of different possibilities [136], [139]. This leads to a trade-off between performance & robustness. With the same problem and content, for instance, all membership functions take the values between 0 and 1. Depending on the personal experience, one can consider dividing the context into 5 partitions, whereas for another expert it might be about 10 or 15 partitions. However, the more partitions are divided the less performance they exhibit. Due to the fact that the rules and ranges of value cannot be defined automatically, the fuzzy logic is also considered as a static method. Here, to be more dynamic and adaptive, this research solicits another computationally intelligent approach (e.g. ant colony optimization (ACO)) to consolidate the measure of trust [140], [141].Hence, we believe that this contribution evaluates the trust level as well as reputation precisely and can overcome the drawback of fuzzy logic. The bi-focal (i.e fuzzy and ACO) intelligent approach to develop trusted platforms has been demonstrated with empirical validation. Next section presents how ACO can be adapted to the proposed model for evaluating the trust and reputation of multi-cloudlets.

9.3 Proposed model with ant colony optimization

9.3.1 Context

In the recent past, cloud storage service [2] have changed strategy to store and to share the data. Previously, data residing on one device was transferred to another by physically connecting the two or by transferring data using mass storage devices. At present, with readily available high speed internet access and cloud storage services like GoogleDrive (`https://cloud.google.com/products/cloud-storage/`), Live Drive (`http://www.livedrive.com/`) and Dropbox (`https://www.drop-box.com/business/cloud-storage-and-backup`) , users are more comfortable in keeping their data on the cloud for ready access to a plethora of devices. The trend of sharing content over social media has added an emerging feature. At present, people store their photos (Flickr, `https://www.flickr.com/`), videos (Vimeo, `https://www.vimeo.com/`) and other shareable documents Cloud storage services, and share photos/clips with their friends, using one click sharing options. The need to synchronize mobile phones, laptops and PCs is more or less eliminated. Cloud storage has also been suitable for business and enterprise users. Cloud enabled services such as web cloud hosting, cloud databases, and cloud-based data backup storage are very popular among normal users, enterprises, and big-data companies. Therefore, considering the different trends of cloud and cloudlet architecture, trust evaluation has become more pertinent, contextually. Mathematical foundation of trust is discussed in next subsection.

9.3.2 Background of the proposed computational intelligent agent based system

While considering the term and unit of trust is abstract, this part of the thesis is to concentrate on the measurement of trust or belief about cloudlets. Before measuring specific trust value, the following assumptions are considered:

- Trust is a relative measure and it also follows a time scale, where either time units to be enhanced or to be diminished.

[2] http://www.infoworld.com/article/2871290/cloud-computing/understanding-cloudstorage-models.html

- Trust values to be measured are composite in nature. It signifies that multiple sources of trust values can be referred under specific set of constraints and finally a single value should be aggregated.

- It is evident that the trust value is measured with appropriate time scale, however, the dynamic value of trust may demonstrate, same static value under subsequent time unit(s) and can undergo a minor change to next time unit (refer table 9.3), in some cases the values did not change. For example, t5 and t4 are sample of instances with random distribution, it is not fixed as t4 and t5. It demonstrated the degree of randomness of trust under very small time instances.

- The composite scale of trust and time is based on the mathematical induction principle and thus a maximum and minimum value is determined. All the intermediate empirical values should be included in the scale.

- The trust values are relative as well as inductive: it signifies that the multiple logical imperatives are evaluated to satisfy the either maximum or minimum referred trust value.

- The inherent uncertainty of trust or reputation of cloudlet could persist with time scale or it can be any of the intermediate incident values. It must be noted that multiple data acquisition sources are referred to evaluate the trust value and by default it is approximate value.

Considering these broader perspectives of trust and reputation, there will be certain typical mathematical components to quantify the foundation of term trust, which apparently is an abstract value.

The thesis defines the following notations and expressions to obtain the approximate relative values of trust for given cloudlet under different set of constraints. However, to establish the proof of concept and to satisfy multiple and random data sources for trust and reputation, a suitable entropy domain should be defined. By definition, entropy refers to disorder or uncertainty, and the definition of entropy pivots on the measure of unpredictability of the state, or equivalently, of its average information content. During

the investigation of entropy of trust, we have followed this principle.
Considering a cloudlet ecosystem which contains a number m of cloudlet meshes $K_i,\ i \in (1, m)$. For each cloudlet mesh K_i consists of a number n of cloudlets. The mathematical notations are:

- Discrete interval scale is chosen $(\gamma 1, \gamma 2, \gamma 3, \gamma 4, \gamma 5)$ to represent the value of trust level and also testimonials of the expressed concepts of security of cloudlets (refers table 9.3).
- For given m numbers of cloudlet meshes the trust values also differ as $T_1 \ldots T_m$.
- From $T_1 \ldots T_m$ the composition of trust is $T_1(AR_1, E_1, ME_1)$ and $T_m(AR_m, E_m, ME_m)$, where AR represents Anticipated Reputation (The computed mean value of cloud drop due to various reasons. While expected value is real time value, anticipated value is desired predictive value. These values should be close, otherwise system tend to be unreliable), E defines Entropy, ME denotes Meta Entropy. With logical additive operation, it is expressed as $T = T_1 \oplus T_m$ (logical operations are mentioned previously).
- Practically, the cloudmesh is expressed as interconnected graph of different cloudelts. For example, when cloudlet M agrees to communicate with cloudlet N in specific cloudmesh (M and N can be resided in the same or different cloudlet meshes), M will first fetch the trust of the reputation of N and the cloudlet mesh master proceed this value for further interaction. The degree of trust is measured as:

  $$T_{MN} = \omega_i \times aReput_{MN} + \omega_j \times tReput_{MN} \tag{9.13}$$

  Where, ω_i and ω_j represents associated weight of importance of reputation, $aReput_{MN}$ is the anticipated reputation value from cloudlet M to N and $tReput_{MN}$ is the complete value of trust in actual from cloudlet M to N.

9.3.2.1 Mathematical basis of trust/reputation measures

Regarding the trust or reputation as the relative index of the entropy, the entropy is investigated with different time interval scales different than γ, which can be ψ, that

means both are changing dynamically. This is because the interval scale differs from context and location as well as on the variable of time elapse. Conventional standard deviation is applied, and as the composite structure of reputation or trust persists. Hence the reputation is always anticipated and predictable value. The level is defined within the maximum and minimum level of reputation. Finally, the relationship is expressed on the basis of standard deviation.

For example: if the time interval for trust is measured from parameter ψ or γ then the multiplication of measured scale and dynamic importance of trust (related to linguistic value) ω yields the expected reputation (close to anticipated reputation):

$$R = \frac{1}{n}\sum_{i=1}^{n}(\psi_i \omega_i) \tag{9.14}$$

where, n is the total number instances of trust to be measured at time t, i is the initial counter from which the value will navigate. However, anticipated reputation cannot be

Level of Trust	Status	Scale	Elapsed Time
1	Near Confidence	$\gamma 1 = 0$	t_5
2	Low Confidence	$\gamma 2 = 0.25$	t_4
3	Threshold	$\gamma 3 = 0.5$	t_3
4	High	$\gamma 4 = 0.75$	t_4
5	Peak Trust	$\gamma 5 = 1$	t_5

Table 9.3: Trust evaluation under uncertainty

calculated as it is only the approximation of reputation, and anticipated reputation is falling in between the trust value of the scale.

For the analytical experiments in the thesis, each artificial ant selects cloudlets of one cloudlet mesh (base cloudlet mesh) successively and pseudo-randomly according to the transition probability to form a complete path (this could be trusted solution path). The transition probability is determined by the pheromone on all cloudlets and the heuristic information. Commonly, the more pheromone deposited on a cloudlet, the higher the probability that the cloudlet is selected by an ant and hence it is more trusted for that specific time instance. However, ACO is a type of random algorithm, which means that it does not necessarily find the best solution; it can only provide a set of optimal solutions for the measurement of trust.

Following the mentioned entropy to represent the embedded uncertainty on the trust value, it is important to evaluate the value of entropy and meta-entropy accordingly and thus standard variance formula modified with the subtraction from anticipated or expected reputation is given, this also will eliminate the noises of data value from multiple sources:

$$E_n = \sqrt{\frac{1}{n}\sum_{i=1}^{n}(\psi_i\omega_i - R)^2} \tag{9.15}$$

$$ME = \sqrt{\frac{1}{n}\sum_{i=1}^{n}(\psi_i\omega_i - R)^2 - E_n^2} \tag{9.16}$$

Thus, the values shown in table 9.3 as per the time scale can be varied with any value from near Confidence to Peak Trust accordingly. Hence numerically, value of E and ME is evaluated with random number generation from base cloudlet/mesh to successive cloudlet/mesh connection. Thus, when the reputation evaluation method should be referred, it could be the combination of anticipated trust, entropy, and meta-entropy (R, E and ME) with logical induction.

9.3.2.2 Analogy with ant colony

The proposed method is analytical and there is a possibility of loss function in trust or reputation with a dynamic time interval scale. At this outset, a loss function is obvious for cloudlet, because all cloudlet components are subjected to loss of trust measures with number of changing constraints. One way to measure the trust is to measure the minimal loss function of dynamic values of cloudlet. If the cloudlet is sequentially positioned from base cloudlet to the successive cloudlets, then degree of trust should be changing and final value of any particular time interval instant is the summation of trust value from base cloudlet to successive cloudlet divided by the total number of cloudlets. However, the loss function of trust is a dynamic function characterizing the loss of its value at time t. The loss rate function is a non-decreasing power function of time. The analogous behavior of trust is presented through ants' pheromone deposition and evaporation. The following expression 9.17 shows that Δ_j is the small amount of trust with regards to the summation

of trust value for changing time interval Ω from initial time interval scale γ.

$$\Delta_j = \frac{1}{n}\sum_{i=1}^{n}\Omega_i \tag{9.17}$$

The loss function can also be expressed as exponential and proportional loss function, which will be a type of convex function and obviously it will be non-decreasing power function of time.

Type of loss function. Considering an entropy structure (for uncertainty) in trust, the basic idea is to visualize a loss function, which can finally be measured with respect to elapsed time. Inspiring the equation from the seminal work [142]

$$L(A,B) = tr(A^{-1}B) - log(det(A^{-1}B)) - m \tag{9.18}$$

where A and B are $m \times m$ matrices, $tr(A^{-1}B)$, $det(A^{-1}B)$ is a trace and a determinant of $(A^{-1}B)$ respectively and, to ensure that L(A,B) is non-negative, we assume that A and B are symmetric positive definite. Therefore, statistically:

- Given a covariance matrix and
- a class of possible candidate covariance structures,

to find for each structure a covariance matrix that minimizes the entropy loss function. However, this loss function improves the present work to establish the correlational measures with pheromone evaporation and deposition (i.e. pheromone update) of ant colony.
To sum up, from the equation 9.17, it is observed that reputation and trust require more adaptive and dynamic model. Thus, in reputation of loss function and reputation itself, may contain uncertain values. This phenomenon motivates another suitable intelligent method though ant colony optimization approach.

9.3.3 Modeling trust with ant colony system: mathematical perspectives

Categorically, ant colony system is modeled to distribute the load of cloudlets according to the reputation. It also expresses a reverse way to identify the most trusted cloudlet that

will have multiple numbers of service request with initial and final time interval. Definitely, the loss function of trust is minimum in that case.
In this thesis, ant colony optimization is used as a trust or reputation search strategy (not as an alert algorithm directly) while distributing the service load. Here also, to incorporate ant colony as an effective trust or reputation search, certain prior initialization of model and parameter setting is required.

The following node scenario is represented an undirected graph G(V, E), in which V is the set of all slave cloudlets in the specific cloudlet mesh of interest among many (whose reputation has to be measured), E is the network set which connects to each slave cloudlet. After dividing the cloudlet meshes with effective groups, and then ants can be assigned to each group randomly. Then each ant will search for trusted entropy from random assignments. There might be several exploration of the trusted path in the form of different search algorithms.

The following parameters should be considered in the metric:

- Expected execution time,
- $E_T(f)$ refers to the path of f at the end of the computing resources to cope with this job takes time T, which is the time to execute the search to find the optimal trust.
- Network latency: $N_D(f)$ refers to the maximum network delay D generated by path f, for the base cloudlet/mesh
- Network bandwidth: $B_W(f)$ refers to the maximum bandwidth provided by the path f, with respect to concurrent cloudlet/mesh.

To investigate a trusted and optimal search path, it becomes mandatory to adjust the pheromone matrix with respect to loss function (evaporation mechanism). In this context, from the ant colony perspectives, group of different colonies can be used for different cloudlet in the same location. The behavior of ants in one colony will be influenced by the solution information received from other colonies, where pheromone is added to the colony

edges that belong to the best solutions of the group of colonies. The pheromone trails on the edges of the best solutions are updated adaptively in response to determined weights, and an extra amount of pheromone has been deposited on the edges of these solutions accordingly. Here, solution denotes the trusted optimal path.

$$Rt_{avg} = \frac{1}{|S|} \sum_{h \in S} Rt^{h}_{opt} \tag{9.19}$$

While accomplishing the final road-map of the proposed steps, the model assumes recursive solutions:

- Overall average reputation and trust Rt_{avg},
- $h \in S$, where S is the optimal search solution and h is the dynamic edge value of trust or reputation for each edge of cloudlet connection.
- Rt^{h}_{opt} is the value of optimal trust with the dynamic edge value h.
- τ_0, the initial amount of pheromone.
- The evaporation rate, ρ, is a parameter in the range [0, 1] that regulates the reduction of pheromone on the edges.
- Pheromone trail $s_{ij}(ij)$, where the 1st part is the suffix of s and other ij is the factor of s it means pheromone trail s from node i to j for suffix and next ij is the factor that is the actual numerical value of pheromone which is approximated deported or evaporated ; for factor ij part the edge will be visible if it is high, that means other ants will follow it, if not then value is less no one will follow it.

The composite mathematical expression is given in equations 9.20 and 9.21. In equation 9.21, τ_0 is finally replaced by the optimum trust, this is in denominator to ensure that optimal value is always a reduced value with power of time.

$$\tau_{ij}(t+1) = (1-\rho)\tau_{ij}(t) + \rho\tau_0 \tag{9.20}$$

$$\tau_{ij}(t+1) = (1-\rho)\tau_{ij}(t) + \rho(\frac{1}{Rt_{opt}}) \tag{9.21}$$

where, τ_{ij} is the edge value of pheromone for graph from cloudlet/mesh i to cloudlet/mesh j and Rt_{opt} is the value of optimal trust. This expression has the influence of increasing the quantity of pheromone(represents reputation and trust) on edges associated with solutions according to the quality of trust and reputation and the current convergence state of the cloudlet information exchange.

Finally, the value of affinity (refer equation 9.22) of trust for M number of ant colonies and N numbers of cloudlets of specific cloudlet mesh can be calculated:

$$affinity(M,N) = 1 - \frac{\sqrt{\sum_{x \in P(M,N)} \frac{\sum_{i=1}^{P(x,M)} s(x,i)}{P(x,N)}}}{|PI(M,N)|} \tag{9.22}$$

Where, the complete participation for all cloudlets in different cloudlet meshes can indicate *PI(M,N), P(x, M)* can indicate individual information exchange towards trust gain. P(M, N) is the total information exchange for M number of ant colonies and N numbers of cloudlets; s(x,i) is the optimal search function from the initial position of dynamic edge h and x is the part of edge which will be searched through search function .

This affinity is a probabilistic numerical value, which indicates trust gain from a base cloudlet/mesh to a destination token. The choice of affinity value is an inductive approach to demonstrate the gain of trust value for a specific location based cloudlet. If more locations and cloudlets are connected, the affinity value also becomes recursive.

Analytical Implementation Following the proposed high-level description of ant colony system and data acquisition model for cloudlet, the analytical simulations are performed. High level description is given here (refer Algorithm 02).Initially, the proposed model uses Nepho[3], which is a command-line tool that orchestrates the creation of complete working application stacks on virtual infrastructure. Initially targeting Amazon Web Services as well as Vagrant, Nepho abstracts data center creation, instance configuration, and application deployment into portable "cloudlets" that can be shared between developers and teams. Considering the deposition and evaporation of pheromone, the work considers false positive and true positive rate of pheromone depending on the magnitude of trust value enhancement. Analytically, which depicts the successful deposition means trusted path and thus false positive will be high. As mentioned in an earlier section that trust is

[3]https://pypi.python.org/pypi/nepho

Algorithm 02: Pheromone Measure for Trust Algorithm

Input:
select $cloudlet_1 \ldots cloudlet_n$
select Colony and map of colony to cloudlet. Applying the ant heuristic
to generate an initial feasible solution for all colonies set an
initial value of pheromone on every edge of colonies and cloudlet
set service iteration number to 1 for each ant colony.
While termination condition is not met **do**
for each cloudlet parallel **do**
for $k = 1$ to m ants **do**
Construct solution and update pheromone (equations 9.20 & 9.21
endfor
if exchange of cloudlet information is equal to Δ, **then** evaluate equation 9.17
improve the solution by local search Rt_{avg}
endif
increment for next service iteration
endfor
endwhile
Display the optimal value of affinity for Trust/Reputation
Stop

an abstract index, therefore the measured value will be analyzed statistically.

The trust evaluation in the previous section using fuzzy components are based on linguistics decisions of human users, whereas ant colony can inspect the performance and quality aspects of messages in cloudlet transmission. Hence, the scope of proposed algorithm and solutions, amalgamating more than one intelligent method could infer about most suitable single or multiple cloudlet operations.

Generally, the proposed architecture can be elaborated as:

- Step 1: Distribute the ants randomly in the solution space. Each ant in the solution space corresponds to the possible combination of the different multiple cloudlet parameters.

- Step 2: Evaluate the value of the loss function

- Step 3: From the Fuzzy rule base: The rule base has been formed by utilizing the past experiences. (table 9.1 and 9.2)

- Step 4: Calculation of the resultant location utilizing the fuzzy rule base, evaluate to

the next best location that the ant can occupy.

- Step 5: Update the ant pheromone for cloudlet nodes

- Step 6: Estimate the movement of the ants

- Step 7: Check if the termination criterion is reached or not. If reached, then stop the algorithm or otherwise move to step 2.

With respect to the above scheme, the value of the loss function can be formulated while combining execution time, network latency and network bandwidth. Both the computationally intelligent techniques have been used with respect to different contexts.

9.4 Conclusion

The trust evaluation is significant in any cloudlet exchange process. However, stereotype intrusion detection and associated measures cannot ensure trust and validation of multiple intelligent sensors and devices under the cloudlet. We propose an elaborated and specific hybridization of computationally intelligent measures of trust both from the perspectives of user and quality of service. In the initial section, the proposal fuzzy computation is adopted to tackle the uncertainty in human inferences while trusting a specific cloudlet. Subsequently, the performance of network also improves the trust validation and therefore the parameters are being measured through ant colony optimization algorithm. In the next chapter, the current results are compared and it is evident that effective hybrid scheme of both fuzzy and ant colony can conceive a comprehensive intelligent measure of trust in the cloudlet paradigm.

Chapter 10

Performance Analysis

The goal of this chapter is to compare between the proposed model and the contemporary models. Regarding to the contributions, the comparison is twofold: firstly, the performance of our proposed security approach, namely Property based Token Attestation, is discussed with various well-known contemporary approaches to figure out the difference in aspect of security and energy consumption. Second, by the same fashion, the performance of relevant trust and reputation models, in terms of accuracy, path length and energy consumption, is compared with the present model to prove the robustness of the contribution.

10.1 Property based Token Attestation

In mobile computing, performance and security are two important factors that need to be addressed to figure out how much protection a security mechanism (e.g. cryptographic algorithm) can offer and how much energy consumption will be lowered by using a corresponded security mechanism. Thus, depending on the user's requirement, the trade-off between security and performance needs to be considered. The author in [143] proposed a trade-off function that presents a relationship between performance and security metrics:

$$maxU = \omega P + (1 - \omega)S,\ 0 < \omega < 1 \tag{10.1}$$

where ω is the weight factor to present the preferences of the user on security and performance. In this part, we analyze the performance based security by comparing our proposed solution with the others [144], [145], [146], [67] and [147], which also have presented security protocols in mobile computing environment. We also use a notion N*T where T refers to

types of operations (e.g. hash, encryption/decryption operation) and N mentions about how many times this operation can be used in the protocol. While other solutions have used ECC, we use RSA for key encryption, digital signature and DH exponential function. To improve the performance, most of the heavy tasks (e.g. signature generation and complex shared secrets) have been performed remotely by TTP. For example, the heaviest shared secret $N_{TTP} * N_V * N_M$ has been calculated by TTP in flows 6 and 7 (refer Chapter 8). The difference of ECC and RSA in term of performance is represented clearly in [148]. Loosely speaking, they are differences in execution time and energy cost regarding to key generation, signature generation and signature verification. To simplify, we denote T_H as an hash operation, T_{eDH} and T_{eECC} as exponentiation functions of Diffie-Hellman (DH), Elliptic Curve D-H (ECDH) algorithms respectively; T_{ECC} and T_{RSA} refer to the whole operations related to the corresponding algorithms. In this work, we only consider three parameters such as hash, modular exponentiation and asymmetric operation for discussion. The computation cost in the proposed work calculated also the content of similarity to the others. For example, in flow 1, M has 1 Hash operation and 1 RSA operation while V has 1 RSA operation. Totally, we have $1 * T_H, 2 * T_{RSA}$ in flow 1.

On the other hand, depending on the methodology, approach and context, the other solutions may use additional operations such as XOR, or symmetric encryption/decryption operations. For example, besides asymmetric/hash operations, the authors in [146] have used additional 7 XOR operations, and 6 Symmetric Encrypt/Decrypt operations or 7 XOR operations in [145]. Thus, the table 10.1 presents the computation cost based performance related to hash and asymmetric algorithms of different approaches.

However, table 10.1 only presents the overall cost of all the involved parties, including

Reddy et al. [67]	Memon et al. [146]	Mun et al.[145]	Le et al.[147]	Zhao et al.[144]	Proposed Approach
$22T_H + 6T_{ECC}$	$16T_H + 5T_{eECC} + 4T_{ECC}$	$14T_H + 2T_{ECC}$	$18T_H + 4T_{ECC}$	$13T_H + 4T_{ECC}$	$8T_H + 14T_{RSA} + 5T_{eDH}$

Table 10.1: Computation cost

mobile client (poor resource) and remote servers (rich resources), of those solutions. Thus, it is not evident to demonstrate the performance based security of mobile devices, which

are considered as resource-constraint devices. In this step, we analyze the energy cost (in mJ) and execution time (in ms) of the client side by extracting data from [148]. Potlapally et al. measure a wide analysis on the expenses of launching symmetric and asymmetric cryptographic algorithms on a mobile device (HP IPAQ3670), which is equipped with Intel SA-1110 StrongARM processor clocked at 206 MHz, 64 MB SDRAM, 64 kB Cache, 950 mAh Li-On Battery and OS Linux. In our work, we extract only data related to RSA (1024), ECC (160) and Diffie-Hellman (DH), Elliptic Curve D-H (ECDH) key-exchange algorithms (refer table 10.2). For hash operation, we take data from [149], that perform a similar work with Potlapally but measure SHA-256 instead of SHA-1. Thus, we have

Algorithm	Key size (bits)	Key generation (mJ)	Sign (mJ)	Verify (mJ)	Key exchange (mJ)
RSA	1024	2017.13	546.5	15.97	N/A
ECC	160	226.65	134.2	196.23	N/A
DH	1024	875.96	N/A	N/A	1046.5
ECDH	160	276.70	N/A	N/A	163.5

Table 10.2: Energy cost of asymetric algorithms

728mJ for hashing 1MB data in 1000ms. To reduce the complexity, we calculate an average value of each algorithm to figure out T_{RSA}, T_{ECC}, T_{eDH} *and* T_{eECC}. For example,

$$T_{RSA} = \frac{Key\ generation + Sign + Verify}{3} \approx 860(mJ) \tag{10.2}$$

Similarly, we have $T_{ECC} \approx 186(mJ), T_{eDH} \approx 961(mJ)$ *and* $T_{eECC} \approx 220(mJ)$. Consequently, table 10.3 presents the performance consumption based security for the mobile device of mentioned approaches in table 10.1.

Reddy et al. [67]	Memon et al. [146]	Mun et al.[145]	Le et al.[147]	Zhao et al.[144]	Proposed Approach
$10T_H + 3T_{ECC}$	$10T_H$	$5T_H + 1T_{ECC}$	$7T_H + 2T_{ECC}$	$7T_H + 1T_{eECC}$	$3T_H + 4T_{RSA} + 1T_{eDH}$
7838(mJ)	7280(mJ)	3826(mJ)	5468(mJ)	5316(mJ)	6585(mJ)

Table 10.3: Performance consumption based security (mJ)

10.1.1 Conclusion

As this is an empirical evaluation, these values would be varied depending on the different situations such as input data or device configuration. In this contribution, we use these values as a standard scale to prove that despite of using heavy security algorithms, our proposed solution has acceptable performance consumption. As mentioned earlier, the recent challenges of mobile computing are security and energy consumption. By proposing the security protocol and considering its performance, our approach not only addresses the current challenge of mobile computing but also is more realistic than the other existing solutions.

10.2 Trust and Reputation based Intelligence

Before discussing the comparison in detail, we present briefly the well-known models related to trust and reputation management in this section.

Bio-inspired trust and reputation model (BTRM-WSN) [150] is an approach based on ant colony system, where ants construct paths satisfying certain conditions in a graph. To provide trust and reputation in wireless sensor networks (WSNs), the objective of this model is to help a client, who requests a particular service over the network, for finding the most trustworthy route leading to a service provider node. In BTRM-WSN, each node maintains a trail of pheromone $\tau \in [0, 1]$ for each of its neighbors. Because of determining the probability of ants for choosing a certain route, this pheromone is known as the trust that a particular node delivers to the others. Considering, for instance, ant k moves from node i to node j, the pheromone value is updated as the following equation:

$$\tau_{ij} = (1 - \varphi) \cdot \tau_{ij} + \varphi \cdot \Omega \tag{10.3}$$

where τ_{ij} is the pheromone value of the path between node i and j; $\Omega = (1 + (1 - \varphi) \cdot (1 - \tau_{ij}) \cdot \eta_{ij})$ is the convergence value of τ_{ij}; φ and η_{ij} is is a parameter to control the amount of pheromone left by ants and the heuristic value [150], [151], respectively. When ant k reaches a service provider node j, the trustworthy of node j is determined by the average pheromone value (avg_{τ}) from client to node j. With this value, ant k decides to

stop and return the current solution (Sk) with a probability equal to avg_τ to the client (case 1), or keeps tracking another better one based on the reputation (case 2). To make a decision, avg_τ is compared with a defined transition threshold ($TraTH \in [0, 1]$). If (avg_τ) is greater than $TraTH$ then case 1 is performed. Otherwise, ant k decides node j not meets the reputation requirements and then case 2 is considered. Once BTRM-WSN selects the trustworthy path leading to the most reputable node, here the client must perform the flowing complex tasks:

- Examine and assess the path quality of the received solution (denoted by $Q(S_k)$) which is brought back by each ant k.
- Select a path with the highest score among the received path quality of returned ants and store this score as *Current_Best* (CB).
- Then compare CB with the best value *Global_Best* (GB), which is obtained at the previous transaction. If CB is greater than GB, then CB is stored as GB. The pheromone value of this path is updated.
- After selecting the GB solution, the transaction between the client and the specific trustworthy node is established.
- After receiving the requested service, if the client satisfies with the received service, he will give a satisfaction value (Sat) is greater than another defined threshold $PunTH \in [0, 1]$. The pheromone value is updated with a reward.
- Otherwise, the client will give Sat as less than the threshold and the pheromone value is updated with a punishment.

Linguistic Fuzzy Trust Model (LFTM) [152] is known as an improvement of BTRM-WSN by applying the fuzzy theory for the enhancement. The following step is represented how LFTM applies its linguistic fuzzy sets and fuzzy logic to the existing BTRM-WSN model:

- BTRM-WSN selects the node to have a transaction with.

- This node has a goodness linguistic scale, include "Very High, High, Medium, Low and Very Low".

- Depending on the required service attributes, the selected node can provides a service with higher, lower or equal to the expectation.

- The expected service and the received one are compared with certain subjective weights for the attributes of the services.

- The client satisfactory is assessed as presented in BTRM-WSN.

- Together with client's goodness, the punishment level is determined by the client satisfaction with the received service.

PeerTrust [153] is another effective approach for determining and assessing the trustworthiness of peers in Peer to Peer network of e-business communities. The principle of PeerTrust is that the reputation-based trust is established in terms of the quality of services provided by peers, rather than the quality of the feedback arranged by peers. Generally, the trust value $T(u)$ of peer u is represented in the following equation:

$$T(u) = \alpha * \sum_{i=1}^{I(u)} S(u,i) * Cr(p(u,i)) * TF(u,i) + \beta * CF(u) \tag{10.4}$$

Where, I(u) is the total number of transactions performed by peer u with all other peers; p(u,i) refers to the other participating peer in peer u's i^{th} transaction. S(u,i) denotes the normalized amount of satisfaction between 0 and 1 that can be computed based on the feedbacks. Cr(p(u,i)) is the credibility of the feedback submitted by p(u,i) and TF(u, i) denotes the adaptive transaction context factor for peer u's i^{th} transaction, and CF(u) denotes the adaptive community context factor for peer u. The normalized weight factors for the collective evaluation and the community context factor denoted by α and β, respectively. However, for trust computation, the authors have used a binary feedback system where a peer rates the other peer either 0 or 1 according to whether the transaction is satisfactory. This drawback makes this model less flexible to deal with the uncertainty context.

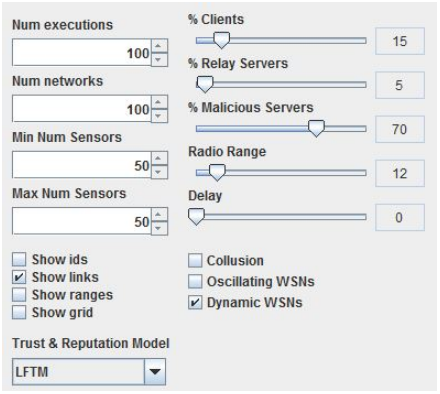
In brief, these mentioned models are the well-known approaches to estimate trust and

reputation in WSN context. Next, we discuss the difference between our approach and these models. In similar fashion, we also use fuzzy logic as well as ACO to evaluate trust and reputation in cloudlet context. However, BTRM-WSN relied mainly on the feedback of clients (e.g. satisfactory value) to decide the trustworthy of the entity by modifying the pheromone value. This can lead to the unfair rating problem. This model also asks the end-user to do a complicated computation for generating the feedback value. After selecting the trusted node by BTRM-WSN, fuzzy theory is applied to improve the performance as presented in LFTM. This improvement helps the client understand precisely the quality of received service based on linguistic scale. Thus the client can leave the accuracy feedback based on his goodness. In contrast, the feedback of users is not sufficient to calculate trust and reputation in our proposed model. Similar to the principle of PeerTrust, we also consider the quality of service as an additional element to estimate trust and reputation of cloudlets. By combining user's feedback and quality of service, the proposed model can prevent the unfair-rating problem [87] as well as collusion attack. It also overcomes the past experience problem when new cloudlets are added on the cloudlet mesh. Once the new cloudlet is added on the mesh, the cloudlet master can rely on the reputation based quality of service of this cloudlet to estimate its trust level. In addition, fuzzy theory in this approach is applied to deal with the uncertainty factor and calculate the trust degree of the certain cloudlet based on user's feedback and QoS. As mentioned earlier, to be more dynamic and adaptive, ACO is considered to consolidate the measure of trust. For example, instead of relying on satisfactory values [150], we calculate and update the pheromone value based on the overall average reputation as presented in equations 9.19 and 9.21. Last but not least, by enforcing the proposed mechanism to the cloudlet master, this approach does not require the end-user to perform unexpected tasks after receiving his suitable service.

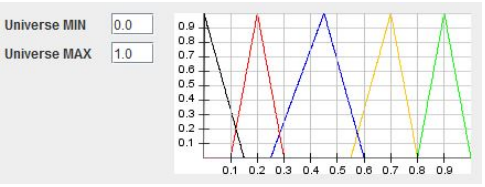
10.2.1 Comparison of Performance

We also use TRMSim-WSN [154], which is a Java-based trust reputation models simulator, to make a comparison among discussed models. The main purpose of TRMSim-WSN is to provide an easy method to test a trust and reputation model over WSNs and also enable users to compare their own model against others. In this part, we consider

comparing the proposed approach to LFTM, which includes BRTM-WSN, and PeerTrust in dynamic WSNs environment. We keep the network configuration similar to these models as presented in figure 10.1a [150], [152].



(a) Network configuration

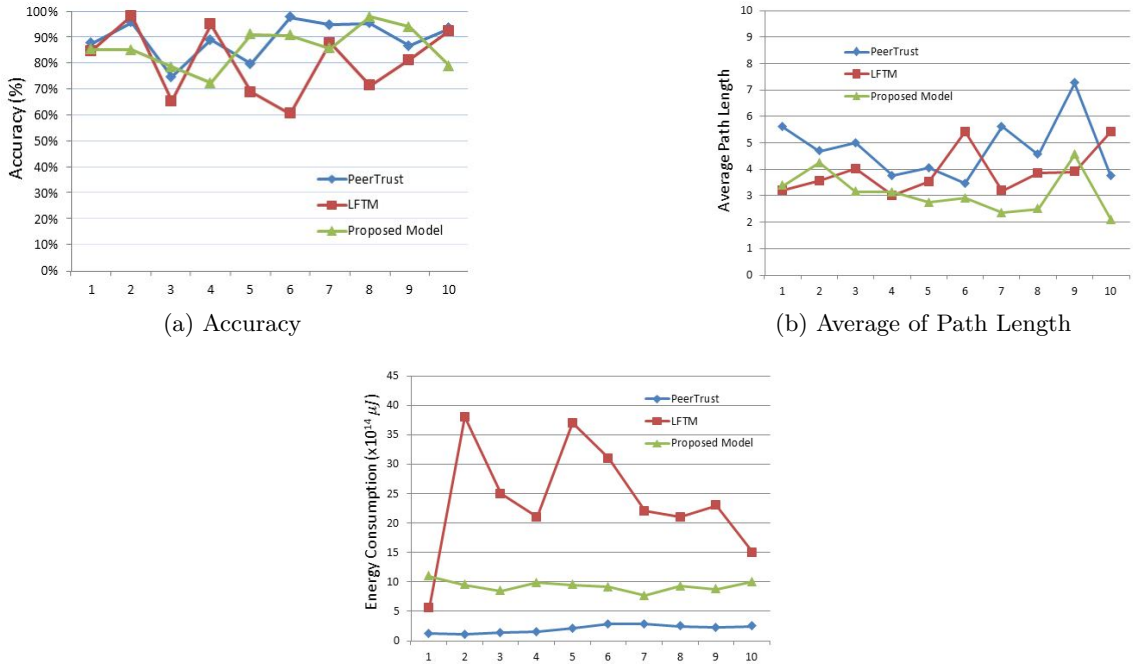


(b) Membership function settings

Figure 10.1: TRMSim-WSN configuration settings

We also adjust the parameters to match with the proposed approach settings. In our proposed model, for example, we do not rely mainly on the user's feedback to modify/update the pheromone value and the client is supposed to receive the suitable requested service from the cloudlet master. Hence, we set the threshold of punishment and transition is equal to maximum and minimum, respectively. Regarding to fuzzy logic, we also change the model of membership function to triangular (refers figure 10.1b), instead of trapezoidal of LFTM, to match with the figure 9.3 (refer Chapter 9). The data of the following graphs are collected by the first 10 values when the simulator is started.

Accuracy. The term accuracy refers to the percentage of the number of time for successful selecting trustworthy nodes. In figure 10.2a, it shows that PeerTrust model has the most stable accuracy. Its accuracy oscillates in the smallest range ($\Delta_{PeerTrust} = \max_{value} - \min_{value} = 97.83\% - 74.76\% = 23.07\%$). Although LFTM has the highest accuracy point (98.57%) but its $\Delta_{LFTM} = 37.75\%$ is higher than others. Similar to PeerTrust, our proposed model also has a stable accuracy and obtains an acceptable accuracy with a maximum and minimum values respectively equal to 98.11% and 72.42% for $\Delta_{proposedmodel} = 25.69\%$.



(a) Accuracy

(b) Average of Path Length

(c) Energy Consumption

Figure 10.2: Performance of LFTM, PeerTrust and the Proposed Approach

Path Length. According to [150; 151; 155] the path length is considered as the average hops leading to the trustworthy nodes which are selected by the users in the network. It assumes that the smaller number, the better performance. As shown in figure 10.2b, the proposed model has the best performance when it has the lowest path length point at 2.09 in comparison with the worst performance of PeerTrust which has 7.26 point. The results of LFTM and our proposed model in this comparison prove that the bio-inspired technique is an effective solution for evaluating the trust and reputation in a computing network.

Energy Consumption. The consumption of energy is the overall energy consumed by all involved entities, such as the clients and remote nodes. The figure 10.2c shows the fact that there is the trade-off between security and performance; and the more security the models support the more energy they consume. For example, LFTM is considered as the most effective model but it consumes the highest energy. Moreover, the graph also demonstrates that our proposed model is suitable for the mobile user in mobile cloud computing environment where the mobile device is known as a resource-constraint device.

10.2.2 Conclusion

In short, this subsection shows the overview of the difference between our proposed approach and the most known models. Although each model has its own pros and cons, they are considered as the effective models to assess the trust and reputation. According to the comparison data, it is worth noting that our proposed approach has obtained the acceptable results in terms of accuracy, shortest path and energy consumption.

10.3 Summary

In this chapter, in order to prove the advantage of our proposed solution, we have analyzed the performance of our approaches against others. To finalize this dissertation, we draw a conclusion in the next chapter.

Chapter 11

Conclusion

The work in this Phd dissertation focuses on building trusted environment for mobile devices in the context of *MCC*. As an essential factor of human life, current improvement of mobile technology makes daily life more convenient and effective regardless of time and place. In fact, people have to deal with many secrets or sensitive data daily, for example: bank account, personal records or even his firm's strategies. Due to the convenience of mobile devices and well-promises of the manufacturers, people tend to use their mobile devices for their important works as well as other normal purposes. Thus, we strongly believe that trusted environment is vital for mobile devices along with the raise of challenges. In this thesis, we presented how to secure mobile devices from the conventional type of security to the intelligence. We also investigated the real case study to address the current challenges of mobile computing which is known as resource-constraint system.

Firstly, coming along with the performance, energy efficiency of different *MCC* architectures are presented by examining various approaches, namely, OSGi, Cloudlet Overlay and Container based solution. We consider that energy efficiency must be addressed not only at the cloud side but also at the mobile side. The former is used to solve the existing limitation of the latter by using remote resource provider rather than processing or storing data locally. After that, we came up with the Droplock system, which is known as the demonstration of MCC model, by introducing the service architecture towards the application of mobile devices, sensors and cloud computing.

Secondly, the primitive security function of Droplock system has inspired us to propose the novel security model, Property based Token Attestation. In this work, we proposed the

token with random properties to authenticate mutually the mobile user and the involved parties. We defined in detail the token content and how it works as well as the random properties list that is generated by the application service automatically, regardless the update or back up of the system. The integrity of properties list is also ensured and verified by the TPM functionalities and the active trusted third party. Hence, our approach not only lessens the remaining demerit of remote attestation mechanisms but also adapts to the context-aware.

Last but not least, by considering everything must be smart, the conventional security solution has not satisfied us much to enhance the security level in the context of *MCC*. As a result, we introduced the approach to estimate the trust and reputation of *MCC* model that is relied on the computational intelligence (i.e. fuzzy logic and ant colony optimization). We believe that the usage of computational intelligence in security perspective is not only restricted with intrusion, rather this contribution demonstrates certain trust decision making paradigms.

At first the energy efficiency of different *MCC* architectures has been investigated, in the end we used this experience to analyze the performance between our approaches and related recent models in order to prove the robustness of our contributions.

Finally, regarding future works, the research on trusted environment design for *MCC* has made great progress however lightweight cryptographic methods such as elliptic curve cryptography need to be investigated for enhancing the performance of PTA. To this end, our approaches have become more adaptable to the context-aware.

For the context-aware, security models can be well extended with the help of computational intelligence. As cloud computing consists of interconnected computing resources and their virtual representation could be standardized with different protocols. Hence, for heterogeneous environment, intelligence must support in context specific security applications. This means there could be possibility to incorporate hybrid intelligent algorithms for mobile cloud. In recent research [155], it has been observed that understanding different uncertain situations across mobile cloud platform hybrid solution using fuzzy computing map has been used. Similarly, different versions of smart fuzzy architectures are available for secure coding and threats investigation models are used [156].

In this Phd thesis, this scope of experimental validation is restricted because of the mention diversify hardware and middleware components. However, an attempt has been made to fetch the live data in the form of context and application for different users has also accomplished. The data-set used in this work is principally synthetic data-set, which has been referred from different mobile cloud users scenarios. The middleware of different mobile devices are treated as different contexts, imprecise and dynamic, therefore the road map of the future work must envisage different adaptive protocols for security mobile cloud paradigm. In this research the modeling and validation have been emphasized. The inclusion of ant colony optimization also indicates the significant lightweight application development to identify the intrusion and related behavior in mobile cloud. The defined extension for computational intelligence with respect to mobile cloud environment should definitely encourage different independent as well as hybrid algorithms to keep the usage more secure. Several components of intelligence can help to design and adjust the security solutions, if and only if the detection of security concern can be done. The prime goal of this thesis is to detect the possibility of attacks in security, identity management scenario and thereby suggesting suitable methods to protect the information of users. Hence, the perspective of computing intelligence and decision support system should be the ideal extension of the proposed measure in this thesis.

Publications

International Journal

1. T. Le Vinh, H. Cagnon, S. Bouzefrane, S. Banerjee. "Property based Token Attestation in Mobile Computing", Concurrency and Computation: Practice and Experience. (To appear)

2. T. Le Vinh, S. Banerjee, S. Bouzefrane, "Estimating Trust value of Multiple Cloudlet Under Network Uncertainty using Computational Intelligence" Emerging Trends, Issues and Challenges in Internet of Things, Big Data and Cloud Computing, Future Generation Computer Systems (FGCS). (Submitted)

International Conferences

1. T. Le Vinh, S. Bouzefrane, S. Banerjee. "Convergence in trusted computing and virtualized systems: A new dimension towards trusted intelligent system", 5th IFIP International Conference on Performance Evaluation and Modeling in Wired and Wireless Networks, November 2016, pp.1-6, Paris, France, (DOI: 10.1109/PEMWN.2016.7842896)

2. T. Le Vinh, R. Pallavali, F. Houacine, S. Bouzefrane. "Energy efficiency in Mobile Cloud Computing Architectures", The IEEE 4th International Conference on Future Internet of Things and Cloud, August 2016, pp.Pages: 326 - 331, Austria, (DOI: 10.1109/W-FiCloud.2016.72)

3. T. Le Vinh, S. Bouzefrane, J. Farinone, A. Attar, B. Kennedy. "Middleware to Integrate Mobile Devices, Sensors and Cloud Computing", The 6th International

Conference on Ambient Systems, Networks and Technologies (ANT-2015)), June 2015, Vol. 25, pp.234–243, London.

4. T. Le Vinh, S. Bouzefrane. "Trusted Platforms to secure Mobile Cloud Computing", The 16th IEEE International Conference on High Performance Computing and Communications , August 2014, pp.1096-1103.

Appendix A

The cloudlet is considered as resource-rich, well-connected and powerful computer installed in the public infrastructure with the connectivity to Cloud server. It can be implemented thanks to Wi-Fi hotspot servers. Thus, a cloudlet also inherits the features of the Cloud in term of Quality of Service, security. In this work, we extract the data from the dataset of Expert Cloud[1], which is a new class of cloud systems to search trustworthy entities in order to benefit from their knowledge and skills [132]. To simplify, we only take A_{req} (number of accepted jobs), N_{req} (number of submitted jobs), SP (Security Parameter), the correct negative/positive opinions which entity takes from the others ($InputdegreeC^{-}$ /$InputdegreeC^{+}$) (refer table 11.1). Another key fact is that the cloudlet is notably useful for the proximate mobile device to distribute its workload while ensuring low delay and high bandwidth. For this reason, we use another dataset from [1] which measures the throughput when downloading a 1MB file over a Wi-Fi hotspot. For example, we take a row 1 to adapt to the mentioned formulas 9.2 & 9.3. We have:

$$\begin{aligned} R_{Si} &= \delta(cf) + (1-\delta)(icf) \\ &= \delta(28) + (1-\delta)(3) \end{aligned}$$

$$\begin{aligned} R_{QoS} &= \omega 1(Av) + \omega 2(Per) + \omega 3(SP) \\ &= \omega 1(\frac{A_{req}}{N_{req}}) + \omega 2(838.62) + \omega 3(0.0781) \\ &= \omega 1(\frac{12}{18}) + \omega 2(838.62) + \omega 3(0.0781) \end{aligned}$$

[1] http://www.dataset12.expertcloud.ir/

Resource No	A_{req}	N_{req}	SP	Input degreeC^-	Input degreeC^+	Throughput(Per)
01	12	18	0.0781	28	3	838.62
02	15	21	0.648	94	83	833.85
03	13	24	0.1825	80	2	840.95
04	14	19	0.621	33	37	868.95
05	13	23	0.756	74	17	1032.12
06	14	22	0.246	57	50	822.78
07	18	21	0.0891	1	50	836.79
08	14	20	0181	42	87	814.54
09	14	25	0.537	2	57	838.53
10	15	23	0.835	79	17	860.47
11	14	20	0.475	99	91	809.86
12	18	22	0.646	69	67	866.52
13	15	25	0.055	78	50	810.15
14	15	19	0.552	60	83	860.14
15	17	19	0.421	28	29	832.17
16	14	19	0.151	75	88	813.17
17	14	20	0.995	70	53	811.21
18	17	23	0.861	44	9	857.94
19	14	23	0.842	42	48	843.59
20	14	19	0.847	15	32	825.95

A_{req}: number of accepted job
N_{req}: number of submitted job
SP: Security level of each resource (i.e cloudlet)
$InputdegreeC^-$: correct negative opinions which entity (i.e cloudlet) taken from other entities (i.e cloudlets, users)
$InputdegreeC^+$: correct positive opinions which entity (i.e cloudlet) taken from other entities (i.e cloudlets, users)
Throughput(Per): the performance of a cloudlet

Table 11.1: Dataset [1], [2]

Bibliography

[1] Jongwon Yoon, Sayandeep Sen, and Joshua Hare. CRAWDAD dataset wisc/wiscape (v.2012-08-03). *CRAWDAD wireless network data archive*, August 2012. doi: 10.15783/C71C7D.

[2] Matin Chiregi and Nima Jafari Navimipour. A New Method for Trust and Reputation Evaluation in the Cloud Environments Using the Recommendations of Opinion Leaders' Entities and Removing the Effect of Troll Entities. *Comput. Hum. Behav.*, 60(C):280–292, July 2016. ISSN 0747-5632. doi: 10.1016/j.chb.2016.02.029.

[3] Samia Bouzefrane, Julien Cordry, Hervé Meunier, and Pierre Paradinas. Evaluation of Java Card Performance. In *Smart Card Research and Advanced Applications*, Lecture Notes in Computer Science, pages 228–240. Springer, Berlin, Heidelberg, September 2008. ISBN 978-3-540-85892-8 978-3-540-85893-5. doi: 10.1007/978-3-540-85893-5_17.

[4] Damien Sauveron. Multiapplication smart card: Towards an open smart card? *Information Security Technical Report*, 14(2):70–78, May 2009. ISSN 1363-4127. doi: 10.1016/j.istr.2009.06.007.

[5] HUIJUN WU. *MOBILE CLOUD COMPUTING.* MORGAN KAUFMANN PUBLISHER, S.l., 2017. ISBN 0-12-809641-1 978-0-12-809641-3. OCLC: 974027147.

[6] Jiehan Zhou, Teemu Leppanen, Erkki Harjula, Mika Ylianttila, Timo Ojala, Chen Yu, and Hai Jin. CloudThings: A common architecture for integrating the Internet of Things with Cloud Computing. *Proceedings of the 2013 IEEE 17th International Conference on Computer Supported Cooperative Work in Design (CSCWD).*

[7] U. A. Kashif, Z. A. Memon, A. R. Balouch, and J. A. Chandio. Distributed trust protocol for IaaS Cloud Computing. In *2015 12th International Bhurban Conference on Applied Sciences and Technology (IBCAST)*, pages 275–279, January 2015. doi: 10.1109/IBCAST.2015.7058516.

[8] Nuno Santos, Himanshu Raj, Stefan Saroiu, and Alec Wolman. Using ARM trustzone to build a trusted language runtime for mobile applications. pages 67–80. ACM Press, 2014. ISBN 978-1-4503-2305-5. doi: 10.1145/2541940.2541949.

[9] Will Arthur and David Challener. *A Practical Guide to TPM 2.0: Using the Trusted Platform Module in the New Age of Security.* Apress, 2015.

[10] Chen Chen, Himanshu Raj, Stefan Saroiu, and Alec Wolman. cTPM: A cloud TPM for cross-device trusted applications. In *Proceedings of the 11th USENIX Conference on Networked Systems Design and Implementation*, pages 187–201. USENIX Association, 2014.

[11] Farzad Sabahi. Cloud Computing Reliability, Availability and Serviceability (RAS): Issues and Challenges. *International Journal on Advances in ICT for Emerging Regions (ICTer)*, 4(2), 2012.

[12] Alan M. Dunn Owen S. Hofmann and Brent Waters Emmett Witchel. Cloaking Malware with the Trusted Platform Module.

[13] Thinh Le Vinh, Reddy Pallavali, Fatiha Houacine, and Samia Bouzefrane. Energy Efficiency in Mobile Cloud Computing Architectures. pages 326–331. IEEE, August 2016. ISBN 978-1-5090-3946-3. doi: 10.1109/W-FiCloud.2016.72.

[14] Thinh Le Vinh, Samia Bouzefrane, Jean-Marc Farinone, Amir Attar, and Brian P. Kennedy. Middleware to Integrate Mobile Devices, Sensors and Cloud Computing. *Procedia Computer Science*, 52:234–243, 2015. ISSN 18770509. doi: 10.1016/j.procs.2015.05.061.

[15] Abdul Nasir Khan, M. L. Mat Kiah, Samee U. Khan, and Sajjad A. Madani. Towards secure mobile cloud computing: A survey. *Future Generation Computer Systems*, 29 (5):1278–1299, July 2013. ISSN 0167-739X. doi: 10.1016/j.future.2012.08.003.

[16] Gerald J. Popek and Robert P. Goldberg. Formal Requirements for Virtualizable Third Generation Architectures. *Commun. ACM*, 17(7):412–421, July 1974. ISSN 0001-0782. doi: 10.1145/361011.361073.

[17] Nivedita Manohar. A Survey of Virtualization Techniques in Cloud Computing. In *Proceedings of International Conference on VLSI, Communication, Advanced Devices, Signals & Systems and Networking (VCASAN-2013)*, Lecture Notes in Electrical Engineering, pages 461–470. Springer, India, 2013. ISBN 978-81-322-1523-3 978-81-322-1524-0. doi: 10.1007/978-81-322-1524-0_54.

[18] Fernando Rodríguez-Haro, Felix Freitag, Leandro Navarro, Efraín Hernánchez-sánchez, Nicandro Farías-Mendoza, Juan Antonio Guerrero-Ibáñez, and Apolinar González-Potes. A summary of virtualization techniques. *Procedia Technology*, 3: 267–272, 2012. ISSN 22120173. doi: 10.1016/j.protcy.2012.03.029.

[19] Jui-Hao Chiang. *Optimization Techniques for Memory Virtualization-Based Resource Management*. PhD thesis, State University of New York at Stony Brook, Stony Brook, NY, USA, 2012.

[20] Matías Zabaljáuregui. Grid Operating Systems/Middlewares and New Virtualization Techniques. Technical report.

[21] T. LE VINH, S. Bouzefrane, and S. Banerjee. Convergence in trusted computing and virtualized systems: A new dimension towards trusted intelligent system. In *2016 International Conference on Performance Evaluation and Modeling in Wired and Wireless Networks (PEMWN)*, pages 1–6, November 2016. doi: 10.1109/PEMWN.2016.7842896.

[22] Wes Felter. An Updated Performance Comparison of Virtual Machines and Linux Containers- IBM Research Report. `https://domino.research.ibm.com/library/cyberdig.nsf/papers/0929052195DD819C85257D2300681E7B/$File/rc25482.pdf`.

[23] T. Sridhar. Cloud Computing - A Primer - The Internet Protocol Journal, Volume 12, No.3. *Cisco*, 12.

[24] Rolf H. Weber and Romana Weber. *Internet of Things*, volume 12. Springer, 2010.

[25] Charith Perera, Arkady Zaslavsky, Peter Christen, and Dimitrios Georgakopoulos. Context Aware Computing for The Internet of Things: A Survey. *IEEE Communications Surveys & Tutorials*, 16(1):414–454, 2014. ISSN 1553-877X. doi: 10.1109/SURV.2013.042313.00197.

[26] Robert Schmohl and Uwe Baumgarten. Heterogeneity in Mobile Computing Environmens. In *ICWN*, pages 461–467, 2008.

[27] Stuart Taylor, Andy Young, Neeraj Kumar, and James Macaulay. Mobile Consumers Reach for the Clouds. *Cisco Internet Business Solutions Group (IBSG)*, 2011.

[28] Niroshinie Fernando, Seng W. Loke, and Wenny Rahayu. Mobile cloud computing: A survey. *Future Generation Computer Systems*, 29(1):84–106, January 2013. ISSN 0167739X. doi: 10.1016/j.future.2012.05.023.

[29] Xiaopeng Fan, Jiannong Cao, and Haixia Mao. A survey of mobile cloud computing. *zTE Communications*, 9(1):4–8, 2011.

[30] Hoang T. Dinh, Chonho Lee, Dusit Niyato, and Ping Wang. A survey of mobile cloud computing: Architecture, applications, and approaches: A survey of mobile cloud computing. *Wireless Communications and Mobile Computing*, 13(18):1587–1611, December 2013. ISSN 15308669. doi: 10.1002/wcm.1203.

[31] Z. Zhang and S. Li. A Survey of Computational Offloading in Mobile Cloud Computing. In *2016 4th IEEE International Conference on Mobile Cloud Computing, Services, and Engineering (MobileCloud)*, pages 81–82, March 2016. doi: 10.1109/MobileCloud.2016.15.

[32] Gonzalo Huerta-Canepa and Dongman Lee. A virtual cloud computing provider for mobile devices. In *Proceedings of the 1st ACM Workshop on Mobile Cloud Computing & Services: Social Networks and Beyond*, page 6. ACM, 2010.

[33] Cristian Barca, Claudiu Barca, Cristian Cucu, Mariuca-Roxana Gavriloaia, Radu Vizireanu, Octavian Fratu, and Simona Halunga. A virtual cloud computing provider

for mobile devices. In *Electronics, Computers and Artificial Intelligence (ECAI), 2016 8th International Conference On*, pages 1–4. IEEE, 2016.

[34] M. Satyanarayanan, P. Bahl, R. Caceres, and N. Davies. The Case for VM-Based Cloudlets in Mobile Computing. *IEEE Pervasive Computing*, 8(4):14–23, October 2009. ISSN 1536-1268. doi: 10.1109/MPRV.2009.82.

[35] Soumya Simanta, Kiryong Ha, Grace Lewis, Ed Morris, and Mahadev Satyanarayanan. A reference architecture for mobile code offload in hostile environments. In *International Conference on Mobile Computing, Applications, and Services*, pages 274–293. Springer, 2012.

[36] O. Kotevska, A. Lbath, and S. Bouzefrane. Toward a real-time framework in cloudlet-based architecture. *Tsinghua Science and Technology*, 21(1):80–88, February 2016. ISSN 1007-0214. doi: 10.1109/TST.2016.7399285.

[37] G. Lewis, S. Echeverría, S. Simanta, B. Bradshaw, and J. Root. Tactical Cloudlets: Moving Cloud Computing to the Edge. In *2014 IEEE Military Communications Conference*, pages 1440–1446, October 2014. doi: 10.1109/MILCOM.2014.238.

[38] Kiryong Ha, Padmanabhan Pillai, Wolfgang Richter, Yoshihisa Abe, and Mahadev Satyanarayanan. Just-in-time Provisioning for Cyber Foraging. In *Proceeding of the 11th Annual International Conference on Mobile Systems, Applications, and Services*, MobiSys '13, pages 153–166, New York, NY, USA, 2013. ACM. ISBN 978-1-4503-1672-9. doi: 10.1145/2462456.2464451.

[39] Mohamed Amine Bouazzouni, Emmanuel Conchon, and Fabrice Peyrard. Trusted mobile computing: An overview of existing solutions. *Future Generation Computer Systems*, June 2016. ISSN 0167-739X. doi: 10.1016/j.future.2016.05.033.

[40] Siani Pearson. *Trusted Computing Platforms: TCPA Technology in Context*. Prentice Hall PTR, Upper Saddle River, NJ, USA, 2002. ISBN 978-0-13-009220-5.

[41] Ross Anderson. *Security Engineering: A Guide to Building Dependable Distributed Systems*. Wiley, New York, 2001. ISBN 978-0-471-38922-4.

[42] Timothy M. Jurgensen. *Smart Cards: The Developer's Toolkit*. Prentice Hall, Upper Saddle River, N.J, 2002. ISBN 0-13-093730-4.

[43] N. Asokan, Jan-Erik Ekberg, Kari Kostiainen, Anand Rajan, Carlos Rozas, Ahmad-Reza Sadeghi, Steffen Schulz, and Christian Wachsmann. Mobile Trusted Computing. *Proceedings of the IEEE*, 102(8):1189–1206, August 2014. ISSN 0018-9219, 1558-2256. doi: 10.1109/JPROC.2014.2332007.

[44] Scott A. Rotondo. Trusted Computing Platform Alliance. In Henk C. A. van Tilborg and Sushil Jajodia, editors, *Encyclopedia of Cryptography and Security*, pages 1332–1332. Springer US, 2011. ISBN 978-1-4419-5905-8 978-1-4419-5906-5. doi: 10.1007/978-1-4419-5906-5_499.

[45] George Coker, Joshua Guttman, Peter Loscocco, Justin Sheehy, and Brian Sniffen. Attestation: Evidence and trust. In *International Conference on Information and Communications Security*, pages 1–18. Springer, 2008.

[46] Aarthi Nagarajan, Vijay Varadharajan, Michael Hitchens, and Eimear Gallery. Property Based Attestation and Trusted Computing: Analysis and Challenges. pages 278–285. IEEE, 2009. ISBN 978-1-4244-5087-9. doi: 10.1109/NSS.2009.83.

[47] Rodrigo Vieira Steiner and Emil Lupu. Attestation in Wireless Sensor Networks: A Survey. *ACM Computing Surveys*, 49(3):1–31, September 2016. ISSN 03600300. doi: 10.1145/2988546.

[48] Nazanin Borhan and Ramlan Mahmod. Platform Property Certificate for Property-based Attestation Model. *International Journal of Computer Applications*, 65(13), 2013.

[49] 2014 Simalliance. Secure-Element-Deployment-Host-Card-Emulation-v1.0.pdf. `http://simalliance.org/wp-content/uploads/2015/03/Secure-Element-Deployment-Host-Card-Emulation-v1.0.pdf`.

[50] Junaid Shuja, Abdullah Gani, Kashif Bilal, Atta Ur Rehman Khan, Sajjad A. Madani, Samee U. Khan, and Albert Y. Zomaya. A survey of mobile device virtualization: Taxonomy and state of the art. *ACM Computing Surveys (CSUR)*, 49(1):1, 2016.

[51] Michael Pearce, Sherali Zeadally, and Ray Hunt. Virtualization: Issues, Security Threats, and Solutions. *ACM Comput. Surv.*, 45(2):17:1–17:39, March 2013. ISSN 0360-0300. doi: 10.1145/2431211.2431216.

[52] Ronald Perez, Reiner Sailer, Leendert van Doorn, and Berger, Stefan. vTPM: Virtualizing the trusted platform module. In *Proc. 15th Conf. on USENIX Security Symposium*, pages 305–320, 2006.

[53] Paul England and Jork Loeser. Para-virtualized TPM sharing. In *Trusted Computing-Challenges and Applications*, pages 119–132. Springer, 2008.

[54] F. John Krautheim, Dhananjay S. Phatak, and Alan T. Sherman. Private Virtual Infrastructure: A Model for Trustworthy Utility Cloud Computing. Technical report, DTIC Document, 2010.

[55] Mario Strasser and Heiko Stamer. A software-based trusted platform module emulator. In *Trusted Computing-Challenges and Applications*, pages 33–47. Springer, 2008.

[56] Ahmad-Reza Sadeghi, Christian Stüble, and Marcel Winandy. Property-based TPM virtualization. In *Information Security*, pages 1–16. Springer, 2008.

[57] Boris Danev, Ramya Jayaram Masti, Ghassan O. Karame, and Srdjan Capkun. Enabling secure VM-vTPM migration in private clouds. In *Proceedings of the 27th Annual Computer Security Applications Conference*, pages 187–196. ACM, 2011.

[58] Brian McGillion, Tanel Dettenborn, Thomas Nyman, and N. Asokan. Open-TEE–An Open Virtual Trusted Execution Environment. In *Trustcom/BigDataSE/ISPA, 2015 IEEE*, volume 1, pages 400–407. IEEE, 2015.

[59] Hailun Tan, Wen Hu, and Sanjay Jha. A TPM-enabled remote attestation protocol (TRAP) in wireless sensor networks. In *Proceedings of the 6th ACM Workshop on Performance Monitoring and Measurement of Heterogeneous Wireless and Wired Networks*, pages 9–16. ACM, 2011.

[60] Jethro G. Beekman, John L. Manferdelli, and David Wagner. Attestation Trans-

parency: Building secure Internet services for legacy clients. pages 687–698. ACM Press, 2016. ISBN 978-1-4503-4233-9. doi: 10.1145/2897845.2897895.

[61] Donglai Fu and Xinguang Peng. TPM-based remote attestation for Wireless Sensor Networks. *Tsinghua Science and Technology*, 21(3):312–321, 2016.

[62] Liqun Chen, Rainer Landfermann, Hans Löhr, Markus Rohe, Ahmad-Reza Sadeghi, and Christian Stüble. A Protocol for Property-based Attestation. In *Proceedings of the First ACM Workshop on Scalable Trusted Computing*, STC '06, pages 7–16, New York, NY, USA, 2006. ACM. ISBN 978-1-59593-548-9. doi: 10.1145/1179474.1179479.

[63] Ahmad-Reza Sadeghi and Christian Stüble. Property-based Attestation for Computing Platforms: Caring About Properties, Not Mechanisms. In *Proceedings of the 2004 Workshop on New Security Paradigms*, NSPW '04, pages 67–77, New York, NY, USA, 2004. ACM. ISBN 978-1-59593-076-7. doi: 10.1145/1065907.1066038.

[64] Tobias Rauter, Andrea Höller, Nermin Kajtazovic, and Christian Kreiner. Privilege-Based Remote Attestation: Towards Integrity Assurance for Lightweight Clients. pages 3–9. ACM Press, 2015. ISBN 978-1-4503-3449-5. doi: 10.1145/2732209.2732211.

[65] Eimear Gallery, Aarthi Nagarajan, and Vijay Varadharajan. A Property-Dependent Agent Transfer Protocol. In Liqun Chen, Chris J. Mitchell, and Andrew Martin, editors, *Trusted Computing*, volume 5471, pages 240–263. Springer Berlin Heidelberg, Berlin, Heidelberg, 2009. ISBN 978-3-642-00586-2 978-3-642-00587-9. doi: 10.1007/978-3-642-00587-9_15.

[66] SiYuan Xin, Yong Zhao, and Yu Li. Property-Based Remote Attestation Oriented to Cloud Computing. pages 1028–1032. IEEE, December 2011. ISBN 978-1-4577-2008-6 978-0-7695-4584-4. doi: 10.1109/CIS.2011.229.

[67] Alavalapati Goutham Reddy, Ashok Kumar Das, Eun-Jun Yoon, and Kee-Young Yoo. A Secure Anonymous Authentication Protocol for Mobile Services on Elliptic Curve Cryptography. *IEEE Access*, 4:4394–4407, 2016. ISSN 2169-3536. doi: 10.1109/ACCESS.2016.2596292.

[68] Abir Awad, Sara Kadry, Brian Lee, and Shuaijun Zhang. Property Based Attestation for a Secure Cloud Monitoring System. In *Proceedings of the 2014 IEEE/ACM 7th International Conference on Utility and Cloud Computing*, pages 934–940. IEEE Computer Society, 2014.

[69] Zhenpeng Liu, Xu Wang, Yifan Liu, Ding Guo, and Xianchao Zhu. Client Oriented Remote Attestation Model in Cloud Environment. *International Journal of Security and Its Applications*, 9(10):395–404, October 2015. ISSN 17389976. doi: 10.14257/ijsia.2015.9.10.36.

[70] Vijay Varadharajan and Udaya Tupakula. Counteracting security attacks in virtual machines in the cloud using property based attestation. *Journal of Network and Computer Applications*, 40:31–45, April 2014. ISSN 1084-8045. doi: 10.1016/j.jnca.2013.08.002.

[71] Mohammad Nauman, Sohail Khan, Xinwen Zhang, and Jean-Pierre Seifert. Beyond kernel-level integrity measurement: Enabling remote attestation for the android platform. In *International Conference on Trust and Trustworthy Computing*, pages 1–15. Springer, 2010.

[72] Aurélien Francillon, Quan Nguyen, Kasper B. Rasmussen, and Gene Tsudik. A Minimalist Approach to Remote Attestation. In *Proceedings of the Conference on Design, Automation & Test in Europe*, DATE '14, pages 244:1–244:6, 3001 Leuven, Belgium, Belgium, 2014. European Design and Automation Association. ISBN 978-3-9815370-2-4.

[73] Kari Kostiainen, N. Asokan, and Jan-Erik Ekberg. Practical Property-Based Attestation on Mobile Devices. In David Hutchison, Takeo Kanade, Josef Kittler, Jon M. Kleinberg, Friedemann Mattern, John C. Mitchell, Moni Naor, Oscar Nierstrasz, C. Pandu Rangan, Bernhard Steffen, Madhu Sudan, Demetri Terzopoulos, Doug Tygar, Moshe Y. Vardi, Gerhard Weikum, Jonathan M. McCune, Boris Balacheff, Adrian Perrig, Ahmad-Reza Sadeghi, Angela Sasse, and Yolanta Beres, editors, *Trust and Trustworthy Computing*, volume 6740, pages 78–92. Springer Berlin Heidelberg, Berlin, Heidelberg, 2011. ISBN 978-3-642-21598-8 978-3-642-21599-5.

[74] Michael Roland. Debugging and Rapid Prototyping of NFC Secure Element Applications. In *Mobile Computing, Applications, and Services*, Lecture Notes of the Institute for Computer Sciences, Social Informatics and Telecommunications Engineering, pages 298–313. Springer, Cham, November 2013. ISBN 978-3-319-05451-3 978-3-319-05452-0. doi: 10.1007/978-3-319-05452-0_28.

[75] Pascal Urien and Selwyn Piramuthu. Securing NFC Mobile Services with Cloud of Secure Elements (CoSE). In *Mobile Computing, Applications, and Services*, pages 322–331. Springer, Cham, November 2013. doi: 10.1007/978-3-319-05452-0_30.

[76] Qi Zhang, Lu Cheng, and Raouf Boutaba. Cloud computing: State-of-the-art and research challenges. *Journal of Internet Services and Applications*, 1(1):7–18, May 2010. ISSN 1867-4828, 1869-0238. doi: 10.1007/s13174-010-0007-6.

[77] Nuno Santos, Krishna P. Gummadi, and Rodrigo Rodrigues. Towards trusted cloud computing. In *Proceedings of the 2009 Conference on Hot Topics in Cloud Computing*, pages 3–3. San Diego, California, 2009.

[78] F. John Krautheim. Private Virtual Infrastructure for Cloud Computing. *HotCloud*, 9:2009–1, 2009.

[79] Kailash Patidar, Ravindra Gupta, Gajendra Singh, Megha Jain, and Priyanka Shrivastava. Integrating the trusted computing platform into the security of cloud computing system. *International Journal of Advanced Research in Computer Science and Software Engineering, ISSN*, 2277, 2012.

[80] Sue-Chen Hsueh, Jing-Yan Lin, and Ming-Yen Lin. Secure cloud storage for convenient data archive of smart phones. pages 156–161. IEEE, June 2011. ISBN 978-1-61284-843-3. doi: 10.1109/ISCE.2011.5973804.

[81] Peng Zou, Chaokun Wang, Zhang Liu, and Dalei Bao. Phosphor: A Cloud Based DRM Scheme with Sim Card. pages 459–463. IEEE, April 2010. ISBN 978-1-4244-6599-6. doi: 10.1109/APWeb.2010.43.

[82] Samia Bouzefrane, Amira F. Benkara Mostefa, Fatiha Houacine, and Herve Cagnon.

Cloudlets Authentication in NFC-Based Mobile Computing. pages 267–272. IEEE, April 2014. ISBN 978-1-4799-4425-5. doi: 10.1109/MobileCloud.2014.46.

[83] Dijiang Huang, Xinwen Zhang, Myong Kang, and Jim Luo. MobiCloud: Building Secure Cloud Framework for Mobile Computing and Communication. pages 27–34. IEEE, June 2010. ISBN 978-1-4244-7327-4. doi: 10.1109/SOSE.2010.20.

[84] Byung-Gon Chun, Sunghwan Ihm, Petros Maniatis, and Mayur Naik. CloneCloud: Boosting Mobile Device Applications Through Cloud Clone Execution. *arXiv:1009.3088 [cs]*, September 2010.

[85] Freerk A. Lootsma. Basic Concepts of Fuzzy Logic. In *Fuzzy Logic for Planning and Decision Making*, number 8 in Applied Optimization, pages 11–37. Springer US, 1997. ISBN 978-1-4419-4779-6 978-1-4757-2618-3. doi: 10.1007/978-1-4757-2618-3_2.

[86] M. N. Cirstea, editor. *Neural and Fuzzy Logic Control of Drives and Power Systems*. Newnes, Oxford ; Burlington, MA, 2002. ISBN 978-0-7506-5558-3. OCLC: ocm59350584.

[87] Siyuan Liu, Han Yu, Chunyan Miao, and Alex C. Kot. A fuzzy logic based reputation model against unfair ratings. In *Proceedings of the 2013 International Conference on Autonomous Agents and Multi-Agent Systems*, pages 821–828. International Foundation for Autonomous Agents and Multiagent Systems, 2013.

[88] Xu Wu Xu Wu. A Fuzzy Reputation-based Trust Management Scheme for Cloud Computing. *International Journal of Digital Content Technology and its Applications*, 6(17):437–445, September 2012. ISSN 1975-9339, 2233-9310. doi: 10.4156/jdcta.vol6.issue17.48.

[89] Shanshan Song, Kai Hwang, Runfang Zhou, and Y.-K. Kwok. Trusted P2P transactions with fuzzy reputation aggregation. *IEEE Internet computing*, 9(6):24–34, 2005.

[90] Saeed Javanmardi, Mohammad Shojafar, Shahdad Shariatmadari, and Sima S. Ahrabi. FR Trust: A Fuzzy Reputation-based Model for Trust Management in Semantic P2P

Grids. *Int. J. Grid Util. Comput.*, 6(1):57–66, December 2015. ISSN 1741-847X. doi: 10.1504/IJGUC.2015.066397.

[91] Li-Xin Wang. *A Course in Fuzzy Systems and Control*. Prentice-Hall, Inc., Upper Saddle River, NJ, USA, 1997. ISBN 978-0-13-540882-7.

[92] John Mitchell, Syed Rizvi, and Jungwoo Ryoo. A Fuzzy-Logic Approach for Evaluating a Cloud Service Provider. In *Software Security and Assurance (ICSSA), International Conference On*, pages 19–24. IEEE, 2015.

[93] Fengming Liu, Xiaoqian Zhu, Yuxi Hu, Lehua Ren, and Henric Johnson. A Cloud Theory-Based Trust Computing Model in Social Networks. *Entropy*, 19(1):11, December 2016. ISSN 1099-4300. doi: 10.3390/e19010011.

[94] Kawser Wazed Nafi, Tonny Shekha kar, Amjad Hossain, and M. M. A. Hashem. An Advanced Certain Trust Model Using Fuzzy Logic and Probabilistic Logic theory. *arXiv:1303.0459 [cs]*, March 2013.

[95] Sebastian Ries, Jussi Kangasharju, and Max Mühlhäuser. Modeling Trust for Users and Agents in Ubiquitous Computing. In Torsten Braun, Georg Carle, and Burkhard Stiller, editors, *Kommunikation in Verteilten Systemen (KiVS)*, pages 51–62. Springer Berlin Heidelberg, Berlin, Heidelberg, 2007. ISBN 978-3-540-69961-3. doi: 10.1007/978-3-540-69962-0_5.

[96] Sebastian Ries, Sheikh Mahbub Habib, Max Mühlhäuser, and Vijay Varadharajan. Certainlogic: A logic for modeling trust and uncertainty. In *International Conference on Trust and Trustworthy Computing*, pages 254–261. Springer, 2011.

[97] C. Qu and R. Buyya. A Cloud Trust Evaluation System Using Hierarchical Fuzzy Inference System for Service Selection. In *2014 IEEE 28th International Conference on Advanced Information Networking and Applications*, pages 850–857, May 2014. doi: 10.1109/AINA.2014.104.

[98] Marco Dorigo and Thomas Stützle. *Ant Colony Optimization*. MIT Press, Cambridge, Mass, 2004. ISBN 978-0-262-04219-2.

[99] Thomas Stützle and Holger H. Hoos. MAX–MIN ant system. *Future generation computer systems*, 16(8):889–914, 2000.

[100] Oscar Castillo, Héctor Neyoy, José Soria, Patricia Melin, and Fevrier Valdez. A new approach for dynamic fuzzy logic parameter tuning in Ant Colony Optimization and its application in fuzzy control of a mobile robot. *Applied Soft Computing*, 28: 150–159, March 2015. ISSN 15684946. doi: 10.1016/j.asoc.2014.12.002.

[101] Teng Gao, Jin-Yan Song, Ji-Yan Zou, Jin-Hua Ding, De-Quan Wang, and Ren-Cheng Jin. An overview of performance trade-off mechanisms in routing protocol for green wireless sensor networks. *Wireless Netw*, 22(1):135–157, January 2016. ISSN 1022-0038, 1572-8196. doi: 10.1007/s11276-015-0960-x.

[102] Cherry Amir, Amr Badr, and Ibrahim Farag. A fuzzy logic controller for ant algorithms. *Computing and Information Systems*, 11(2):26, 2007.

[103] Saied Asghari and Kianoush Azadi. A reliable path between target users and clients in social networks using an inverted ant colony optimization algorithm. *Karbala International Journal of Modern Science*, 3(3):143–152, July 2017. ISSN 2405-609X. doi: 10.1016/j.kijoms.2017.05.004.

[104] José Capela Dias, Penousal Machado, Daniel Castro Silva, and Pedro Henriques Abreu. An Inverted Ant Colony Optimization approach to traffic. *Engineering Applications of Artificial Intelligence*, 36:122–133, November 2014. ISSN 0952-1976. doi: 10.1016/j.engappai.2014.07.005.

[105] L. Shi, Y. Wang, and X. Liu. An ACO-Based Trust Inference Algorithm. In *2014 Ninth International Conference on Broadband and Wireless Computing, Communication and Applications*, pages 216–220, November 2014. doi: 10.1109/BWCCA.2014.70.

[106] Jennifer Ann Golbeck. *Computing and Applying Trust in Web-Based Social Networks*. PhD thesis, 2005.

[107] S. Biswas, P. Dey, and S. Neogy. Trusted checkpointing based on ant colony optimization in MANET. In *2012 Third International Conference on Emerg-*

ing Applications of Information Technology, pages 433–438, November 2012. doi: 10.1109/EAIT.2012.6408002.

[108] Srinivas Sethi and Siba K. Udgata. Fuzzy-based Trusted Ant Routing (FTAR) Protocol in Mobile Ad Hoc Networks. In *Proceedings of the 5th International Conference on Multi-Disciplinary Trends in Artificial Intelligence*, MIWAI'11, pages 112–123, Berlin, Heidelberg, 2011. Springer-Verlag. ISBN 978-3-642-25724-7. doi: 10.1007/978-3-642-25725-4_10.

[109] Fatiha Houacine, Samia Bouzefrane, and Aghiles Adjaz. Service Architecture for Multi-environment Mobile Cloud Services. *Int. J. High Perform. Comput. Netw.*, 9 (4):342–355, January 2016. ISSN 1740-0562. doi: 10.1504/IJHPCN.2016.077830.

[110] Karthik Kumar, Jibang Liu, Yung-Hsiang Lu, and Bharat Bhargava. A Survey of Computation Offloading for Mobile Systems. *Mobile Networks and Applications*, 18(1): 129–140, February 2013. ISSN 1383-469X, 1572-8153. doi: 10.1007/s11036-012-0368-0.

[111] Samia Bouzefrane and Le Vinh Thinh. Trusted Platforms to Secure Mobile Cloud Computing. pages 1068–1075. IEEE, August 2014. ISBN 978-1-4799-6123-8. doi: 10.1109/HPCC.2014.180.

[112] Cas Cremers and Sjouke Mauw. *Operational Semantics and Verification of Security Protocols*. Information Security and Cryptography. Springer Berlin Heidelberg, Berlin, Heidelberg, 2012. ISBN 978-3-540-78635-1 978-3-540-78636-8.

[113] Ernie Brickell, Jan Camenisch, and Liqun Chen. Direct anonymous attestation. In *Proceedings of the 11th ACM Conference on Computer and Communications Security*, pages 132–145. ACM, 2004.

[114] Ben Smyth, Mark Ryan, and Liqun Chen. Direct Anonymous Attestation (DAA): Ensuring privacy with corrupt administrators. In *European Workshop on Security in Ad-Hoc and Sensor Networks*, pages 218–231. Springer, 2007.

[115] George Coker, Joshua Guttman, Peter Loscocco, Justin Sheehy, and Brian Sniffen. Attestation: Evidence and trust. In *International Conference on Information and Communications Security*, pages 1–18. Springer, 2008.

[116] S. Cheng, L. Bing, X. Yang, Y. Yixian, Z. Li, and Y. Han. A Security-Enhanced Remote Platform Integrity Attestation Scheme. In *2009 5th International Conference on Wireless Communications, Networking and Mobile Computing*, pages 1–4, September 2009. doi: 10.1109/WICOM.2009.5302068.

[117] Gergely Alpár and Jaap-Henk Hoepman. Avoiding man-in-the-middle attacks when verifying public terminals. In *IFIP PrimeLife International Summer School on Privacy and Identity Management for Life*, pages 261–273. Springer, 2011.

[118] Aurelien Francillon, Quan Nguyen, Kasper Bonne Rasmussen, and Gene Tsudik. Systematic Treatment of Remote Attestation. *IACR Cryptology ePrint Archive*, 2012: 713, 2012.

[119] D. Dolev and A. C. Yao. On the Security of Public Key Protocols. In *Proceedings of the 22Nd Annual Symposium on Foundations of Computer Science*, SFCS '81, pages 350–357, Washington, DC, USA, 1981. IEEE Computer Society. doi: 10.1109/SFCS.1981.32.

[120] Cas JF Cremers. The Scyther Tool: Verification, falsification, and analysis of security protocols. In *International Conference on Computer Aided Verification*, pages 414–418. Springer, 2008.

[121] TPM 2.0 Trusted Computing Group. TPM 2.0 Mobile Reference Architecture Specification.

[122] M. Sujithra, G. Padmavathi, and Sathya Narayanan. Mobile Device Data Security: A Cryptographic Approach by Outsourcing Mobile Data to Cloud. *Procedia Computer Science*, 47:480–485, 2015. ISSN 18770509. doi: 10.1016/j.procs.2015.03.232.

[123] H. Othman, H. Hashim, and J. l Ab Manan. A conceptual framework providing Direct Anonymous Attestation (DAA) protocol in trusted location-based services (LBS). In *2010 International Conference for Internet Technology and Secured Transactions*, pages 1–7, November 2010.

[124] Lo'ai A. Tawalbeh, Fadi Ababneh, Yaser Jararweh, and Fahd AlDosari. Trust delegation-based secure mobile cloud computing framework. *International Journal*

of Information and Computer Security, 9(1/2):36, 2017. ISSN 1744-1765, 1744-1773. doi: 10.1504/IJICS.2017.10003598.

[125] Jens Lansing and Ali Sunyaev. Trust in Cloud Computing: Conceptual Typology and Trust-Building Antecedents. *SIGMIS Database*, 47(2):58–96, June 2016. ISSN 0095-0033. doi: 10.1145/2963175.2963179.

[126] Jingwei Huang and David M. Nicol. Trust mechanisms for cloud computing. *Journal of Cloud Computing: Advances, Systems and Applications*, 2(1):9, April 2013. ISSN 2192-113X. doi: 10.1186/2192-113X-2-9.

[127] Wanita Sherchan, Surya Nepal, and Cecile Paris. A survey of trust in social networks. *ACM Computing Surveys*, 45(4):1–33, August 2013. ISSN 03600300. doi: 10.1145/2501654.2501661.

[128] Yao Wang and Julita Vassileva. A Review on Trust and Reputation for Web Service Selection. In *Proceedings of the 27th International Conference on Distributed Computing Systems Workshops*, ICDCSW '07, pages 25–, Washington, DC, USA, 2007. IEEE Computer Society. ISBN 978-0-7695-2838-0. doi: 10.1109/ICDCSW.2007.16.

[129] Yanjun Zuo. Reputation-based service migration for moving target defense. pages 0239–0245. IEEE, May 2016. ISBN 978-1-4673-9985-2. doi: 10.1109/EIT.2016.7535247.

[130] Talal H. Noor, Quan Z. Sheng, and Abdullah Alfazi. Reputation Attacks Detection for Effective Trust Assessment among Cloud Services. pages 469–476. IEEE, July 2013. ISBN 978-0-7695-5022-0. doi: 10.1109/TrustCom.2013.59.

[131] Qingtao Wu, Xulong Zhang, Mingchuan Zhang, Ying Lou, Ruijuan Zheng, and Wangyang Wei. Reputation Revision Method for Selecting Cloud Services Based on Prior Knowledge and a Market Mechanism. *The Scientific World Journal*, 2014:1–9, 2014. ISSN 2356-6140, 1537-744X. doi: 10.1155/2014/617087.

[132] Manel Mrabet, Yosra ben Saied, and Leila Azzouz Saidane. A new trust evaluation approach for cloud computing environments. pages 1–6. IEEE, November 2016. ISBN 978-1-5090-2670-8. doi: 10.1109/PEMWN.2016.7842907.

[133] M. Sulaiman Khan, Maybin Muyeba, Christos Tjortjis, and Frans Coenen. An effective fuzzy healthy association rule mining algorithm (FHARM). *databases*, 4(5): 14, 2007.

[134] Nguyen Tuan Anh and Tran Thai Son. Improve efficiency of fuzzy association rule using hedge algebra approach. *Journal of Computer Science and Cybernetics*, 30(4), February 2015. ISSN 1813-9663, 1813-9663. doi: 10.15625/1813-9663/30/4/4020.

[135] N. M. De Reus. Assessment of benefits and drawbacks of using fuzzy logic, especially in fire control systems. Technical report, FYSISCH EN ELEKTRONISCH LAB TNO THE HAGUE (NETHERLANDS), 1994.

[136] K. R. Sasikala and M. Petrou. Generalised fuzzy aggregation in estimating the risk of desertification of a burned forest. *Fuzzy Sets and Systems*, 118(1):121–137, February 2001. ISSN 0165-0114. doi: 10.1016/S0165-0114(99)00064-0.

[137] Michael AS Guth. Some uses and limitations of fuzzy logic in artificial intelligence reasoning for reactor control. *Nuclear Engineering and Design*, 113(1):99–109, 1989.

[138] M. Braae and D. A. Rutherford. Theoretical and linguistic aspects of the fuzzy logic controller. *Automatica*, 15(5):553–577, September 1979. ISSN 0005-1098. doi: 10.1016/0005-1098(79)90005-0.

[139] MuhammadShahzad Shamim, SyedAther Enam, Uvais Qidwai, and SaniyaSiraj Godil. Fuzzy logic: A "simple" solution for complexities in neurosciences? *Surgical Neurology International*, 2(1):24, 2011. ISSN 2152-7806. doi: 10.4103/2152-7806.77177.

[140] Shashvat Sanadhya and Shailendra Singh. Trust Calculation with Ant Colony Optimization in Online Social Networks. *Procedia Computer Science*, 54:186–195, 2015. ISSN 18770509. doi: 10.1016/j.procs.2015.06.021.

[141] Phannakan Tengkiattrakul, Saranya Maneeroj, and Atsuhiro Takasu. Applying ant-colony concepts to trust-based recommender systems. pages 34–41. ACM Press, 2016. ISBN 978-1-4503-4807-2. doi: 10.1145/3011141.3011161.

[142] Lijing Lin, Nicholas J. Higham, and Jianxin Pan. Covariance structure regularization via entropy loss function. *Computational Statistics & Data Analysis*, 72(Supplement C):315–327, April 2014. ISSN 0167-9473. doi: 10.1016/j.csda.2013.10.004.

[143] Charlie Kaufman, Radia Perlman, and Michael Speciner. *Network Security: Private Communication in a Public World*. Prentice Hall series in computer networking and distributed systems. PTR Prentice Hall, Englewood Cliffs, New Jersey, 1995. ISBN 0-13-061466-1.

[144] Dawei Zhao, Haipeng Peng, Lixiang Li, and Yixian Yang. A Secure and Effective Anonymous Authentication Scheme for Roaming Service in Global Mobility Networks. *Wireless Personal Communications*, 78(1):247–269, September 2014. ISSN 0929-6212, 1572-834X. doi: 10.1007/s11277-014-1750-y.

[145] Hyeran Mun, Kyusuk Han, Yan Sun Lee, Chan Yeob Yeun, and Hyo Hyun Choi. Enhanced secure anonymous authentication scheme for roaming service in global mobility networks. *Mathematical and Computer Modelling*, 55(1-2):214–222, January 2012. ISSN 08957177. doi: 10.1016/j.mcm.2011.04.036.

[146] Imran Memon, Ibrar Hussain, Rizwan Akhtar, and Gencai Chen. Enhanced Privacy and Authentication: An Efficient and Secure Anonymous Communication for Location Based Service Using Asymmetric Cryptography Scheme. *Wireless Personal Communications*, 84(2):1487–1508, September 2015. ISSN 0929-6212, 1572-834X. doi: 10.1007/s11277-015-2699-1.

[147] Hai Duong Le. A Novel Untraceable Authentication Scheme for Mobile Roaming in GLOMONET. In *TechRepublic*, volume 17, pages 395–404, July 2015.

[148] Nachiketh R. Potlapally, Srivaths Ravi, Anand Raghunathan, and Niraj K. Jha. Analyzing the energy consumption of security protocols. In *Proceedings of the 2003 International Symposium on Low Power Electronics and Design*, pages 30–35. ACM, 2003.

[149] Helena Rifà-Pous and Jordi Herrera-Joancomartí. Computational and Energy Costs

of Cryptographic Algorithms on Handheld Devices. *Future Internet*, 3(4):31–48, February 2011. ISSN 1999-5903. doi: 10.3390/fi3010031.

[150] Félix Gómez Mármol and Gregorio Martínez Pérez. Providing trust in wireless sensor networks using a bio-inspired technique. *Telecommunication Systems*, 46(2):163–180, February 2011. ISSN 1018-4864, 1572-9451. doi: 10.1007/s11235-010-9281-7.

[151] Hosein Marzi and Mengdu Li. An Enhanced Bio-inspired Trust and Reputation Model for Wireless Sensor Network. *Procedia Computer Science*, 19:1159–1166, 2013. ISSN 18770509. doi: 10.1016/j.procs.2013.06.165.

[152] Zainab Hassan Fakhri. Performance Analysis of Dynamic Wireless Sensor Networks using Linguistic Fuzzy. *International Journal of Computer Applications*, 87(2), 2014.

[153] Li Xiong and Ling Liu. Peertrust: Supporting reputation-based trust for peer-to-peer electronic communities. *IEEE transactions on Knowledge and Data Engineering*, 16 (7):843–857, 2004.

[154] Félix Gómez Mármol and Gregorio Martínez Pérez. TRMSim-WSN, trust and reputation models simulator for wireless sensor networks. In *Communications, 2009. ICC'09. IEEE International Conference On*, pages 1–5. IEEE, 2009.

[155] Hui Xia, Xiu-qing Lu, and Zhen-kuan Pan. A Novel Ant Colony Optimization Algorithm for QoS-Based Multicast Trusted Routing in Wireless Ad Hoc Networks. *International Journal of Security and Its Applications*, 9(8):111–126, August 2017. ISSN 17389976. doi: 10.14257/ijsia.2015.9.8.09.

[156] Jungho Kang and Jong Hyuk Park. A secure-coding and vulnerability check system based on smart-fuzzing and exploit. *Neurocomputing*, 256(Supplement C):23–34, September 2017. ISSN 0925-2312. doi: 10.1016/j.neucom.2015.11.139.

le cnam

Thinh LE VINH

Security and Trust in Mobile Cloud Computing

le cnam

Abstract:
In this thesis, we investigate the way to secure mobile devices from the primitive security level (Trusted Platforms) to the sophisticated one (bio-inspired intelligence). More precisely, after addressing the challenges of mobile cloud computing (MCC), we have studied the real-case of mobile cloud computing, in terms of energy efficiency and performance, as well as proposed a demonstration of particular *MCC* model, called Droplock system. Moreover, taking advantages of trusted platform module functionality, we introduced a novel schema of remote attestation to secure mobile devices in the context of Mobile-Cloud based solution. To enhance the security level, we used fuzzy logic combining with ant colony system to assess the trust and reputation for securing another mobile cloud computing model based on the cloudlet notion.

Keywords:
trusted computing, mobile cloud computing, attestation, energy efficiency, fuzzy logic, ant colony system.

Résumé :
Dans cette thèse, nous étudions la sécurité des plates-formes mobiles en passant d'un niveau de sécurité primitif qui s'appuie sur les plates-formes de confiance, à un niveau plus sophistiqué qui se base sur de l'intelligence bio-inspirée. Plus précisément, après avoir abordé les défis du cloud computing mobile (MCC), nous avons développé une étude de cas appelée Droplock pour le cloud mobile et nous avons étudié son efficacité énergétique et ses performances pour illustrer le modèle MCC. En outre, en s'appuyant sur les plates-formes de confiance (comme les TPM), nous avons introduit un nouveau schéma d'attestation à distance pour sécuriser les plates-formes mobiles dans le contexte du cloud mobile. Pour améliorer le niveau de sécurité et être adaptatif au contexte, nous avons utilisé de la logique floue combinant un système de colonies de fourmis pour évaluer la confiance et la réputation du cloud mobile basé sur la notion de cloudlets.

Mots clés :
informatique de confiance, cloud mobile, attestation, efficacité énergétique, logique floue, algorithme de fourmis.