HAL
open science

Property based Token Attestation in Mobile Computing

Thinh Le Vinh, Hervé Cagnon, Samia Bouzefrane, Soumya Banerjee

To cite this version:

Thinh Le Vinh, Hervé Cagnon, Samia Bouzefrane, Soumya Banerjee. Property based Token Attestation in Mobile Computing. Concurrency and Computation: Practice and Experience, 2020, 10.1002/cpe . hal-02920654

HAL Id: hal-02920654

https://hal.science/hal-02920654v1

Submitted on 24 Aug 2020

HAL is a multi-disciplinary open access archive for the deposit and dissemination of scientific research documents, whether they are published or not. The documents may come from teaching and research institutions in France or abroad, or from public or private research centers.

L'archive ouverte pluridisciplinaire **HAL**, est destinée au dépôt et à la diffusion de documents scientifiques de niveau recherche, publiés ou non, émanant des établissements d'enseignement et de recherche français ou étrangers, des laboratoires publics ou privés.

Property based Token Attestation in Mobile Computing

Thinh Le Vinh[1], Hervé Cagnon[1], Samia Bouzefrane[1*], Soumya Banerjee[2]

Conservatoire National des Arts et Métiers Paris, France[1], Birla Institute of Technology Mesra, India[2]

SUMMARY

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. This paper focuses on an important feature in many security protocols and application, which is the device attestation in the Mobile Cloud Computing (MCC). The existing remote attestation mechanisms are currently used in trusted computing environment such as Binary Attestation and Property based Attestation. In this paper, 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 represent the common threats and then explain a novel attestation schema for addressing these threats by providing security proofs. In addition, Scyther is in use to verify the correctness of our protocol.

KEY WORDS: Security; TPM; Mobile Computing; Attestation; BYOD; MCC.

1. INTRODUCTION

As the proliferating quickly of the pace of mobile devices, we are aware that each year a wide variety of devices in different factors, improved capabilities, and more intelligence are published to the market. According to Cisco [1], global mobile devices and connections grow to around 8.3 billion in 2016. It predicts that this trend will grow to 11.6 billion by 2020. Mobile devices have become an essential part of human life for serving daily activities. Since cloud computing has been recognized as the next generation of computing infrastructure, mobile computing takes full advantage of the availability of Cloud computing facilities to reduce its limitation such as user scalability, data storage, data processing ability in heterogamous resources environment, and energy saving [2].

In addition, with the existing of personal mobile devices in workplace, most organizations have realized the business benefits like cost savings, job satisfaction and increased employee productivity

*Correspondence to: Samia Bouzefrane; E-mail: samia.bouzefrane@lecnam.net; 292 Rue Saint-Martin, 75003 Paris

as personally owned devices can serve for either corporate purposes or personal needs. This concept is called *Bring Your Own Device*. By turning businesses mobile, it is considered as a significant trend that transcends both industry and geographical boundaries [3, 4]. However, in BYOD concept, it also poses a number of security risks which threaten company's sensitive business data. Furthermore, with the constrained resource, mobile devices lack of sophisticated methods to secure themselves against the risks associated with security and privacy. In the end, information is vital and all possible measures need to be evaluated and implemented for security purpose.

By supporting hardware to provide security primitives independent from other system components, the concept of trusted computing is not new and is described in many researches. 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 the remote system will be detected if it is compromising. The properties of 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.

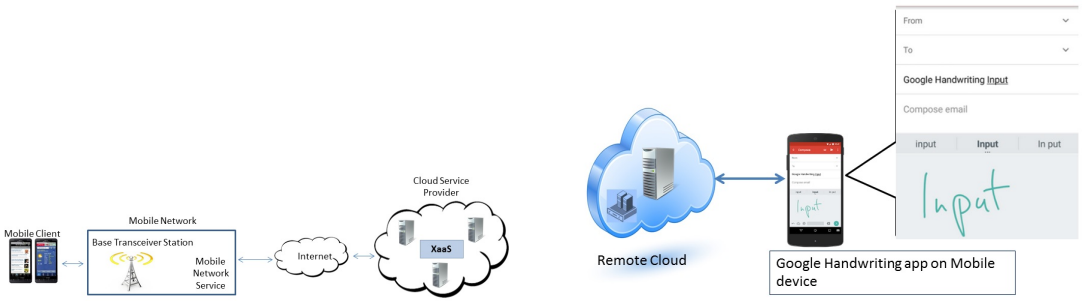
The Trusted Platform Module (TPM), as developed by the Trusted Computing Group (TCG), has been specifically designed to be a building block for trusted computing. It is a significant use in industry and government, for example: Bitlocker for driver encryption in Microsoft and security solution for laptops in the United States Department of Defense [5]. 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.

The main contributions of this paper are two-fold: First, we present a *Property based Token Attestation (PTA)* as well as the Token of random properties that relied on the literature of property-based attestation. Second, our proposed protocol is verified by using Scyther which is a cryptographic protocol verification tool. The rest of the paper is organised as follows: we begin in Section 2 by representing the background related to our work including the existing remote attestation mechanisms. To achieve a secure MCC, the possible scenario and its threats are pointed out in Section 3. We then figure out how to address these threats by introducing a proposed protocol in Section 4. After analysing the security proofs in Section 5, we discuss the main concerns of security with the proposed protocol including the Scyther in Section 6. Section 7 presents the existing solutions with their pros and cons while explaining how different between our solution and theirs. Finally, we draw the conclusion and refer to the next step.

2. BACKGROUND

2.1. Mobile Cloud Computing

In the recent years, the term of Cloud has become the technology trend and ubiquitous in the computing ecosystem. To address the challenges in the computing world such as user scalability, data storage, data processing ability in heterogeneous resources environment, and energy saving, 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 [2]. As a result, the shared



(a) MCC Architecture (b) Client-Server Mode

Figure 1. Mobile Cloud Computing architecture and Client-Server Model

Cloud infrastructure services work as the utility: the users with their devices 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.

Mobile Cloud Computing (MCC) is viewed as the combination of Cloud Computing and mobile networks to bring benefits for mobile users, while augmenting the resource-constraint mobile devices with varied Cloud-based resources in terms of data storage, computation and energy. Although MCC's architectures can be classified into many categories, the figure 1a [6] represents the general architecture of MCC.

To propose a global security solution for MCC in this paper, the traditional Client- Server model of MCC is used to adapt for a proposed solution. Theoretically, 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. In Figure 1b, 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. Based on this model, our work is to answer the question on how to build a secure and trust environment for MCC. In common computing environment, security issues remain a challenge, such as integrity, anonymity, confidentiality, authenticity, and non-repudiation. To address these issues, there are many solutions to protect our system, namely anti-virus, firewalls and cryptography. However, these solutions require a platform that they can trust to work. This platform refers to the trusted platform. According to Balacheff et al. [7], the 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. In our context, the Trusted Platform Module (TPM) is implemented to enable this foundation of trust to provide a set of security features.

2.2. *Trusted Platform Module*

The Trusted Platform Module (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) [8] refers to this secure hardware chip. TPM is the core component of Trusted Computing Group (TCG) which is a consortium

of companies: Compaq, HP, IBM, Intel, Microsoft, AMD, etc. [9, 10]. 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 [11], 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.

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 is

PCRnew value= Digest of (PCR old value|| data to extend) [10]

In remote attestation as discussed later, 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.

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 and Property Based Attestation [12]. The latter is Hardware-based, Software-based, and Hybrid technique [13]. 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. Generally speaking (See Fig. 3), when platform A (Attester) requests access to the protected resources on the platform V (Verifier), before granting a right to access, V requires a device attestation to confirm the trusted state of A. Platform A then provides some measurement data which represent the state of the device as an evidence of its integrity. To approve the resource request, the platform V verifies received evidence to determine whether the platform is in an expected state.

The TCG attestation mechanism is sometime called Binary Attestation. It enables the Attester to provide its evidence for the Verifier. V then evaluates the knowledge of the received AIK called Certificate of AIK, which can be provided by a trusted third party (Privacy Certification Authority), to verify the signature on the A's PCR value. After recalculating/comparing the received value of PCR, the platform V determines the platform A' status and the reporting content's authenticity. Besides the advantages of Binary Attestation (BA), it remains some drawback as described in [11, 12, 14]: 1) Privacy violation: it discloses the complete information about the hardware and

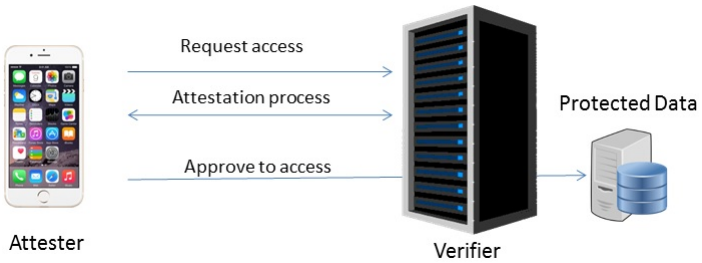


Figure 2. Remote Attestation

software configuration of all platforms; 2) Flexibility limit: data is sealed to a certain platform configuration. As a consequence, this data is inaccessible after a change of system configuration such as update, patch, and back up; 3) Scalability: it does not necessarily imply that the platform complies with desired properties.

In the other hand, another approach of attestation is founded on the Binary Attestation, called Property-based Attestation (PBA). 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. [12] 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. However, each mechanism has its own pros and cons. By taking the advantage of availability of either BA or PBA facilities, our work not only lowers the shortcomings of existing mechanisms but also proposes a new schema of attestation based on the token which contains particular authenticated information for obtaining a specific data.

3. SCENARIO AND THREATS

3.1. Scenario

Bring Your Own Device (BYOD); this term has become a significant trend not only now but also in the future 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 his 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 be proved its secure evidence before being granted a right to access or receive specific information.

3.2. 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 difference only is that the user increases the risks by more exposing her devices because of multi-working environments and the enterprise is out of control in the device's using circumstances, such as selling, buying, or maintaining. For this moment, 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. Within this scope of article, vulnerabilities in Cloud are neglected. Before discussing the goal of our proposed solution to address the following listed threats in Section 6, 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. For private using, the mobile also contains many privacy issues and insufficient data encryption processing. 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 using. 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 stolen are the other risk factors to breach the security of mobile devices.

4. 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 Fig. 3), the Employee needs a valid Token provided by the trusted Third Party to access company's resources. To obtain this Token, she has to show her secure evidence of hardware/software platform to the Third Party, which provides cloud-based security. After evaluating user's evidence, the Third Party then issues the corresponding Token based on received valid evidence. The Token, in this case, works as *The Letter of Introduction*. 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 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.

4.1. Definition of property and trust token

The property in 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 [12]. 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 properties 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 property; 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 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.

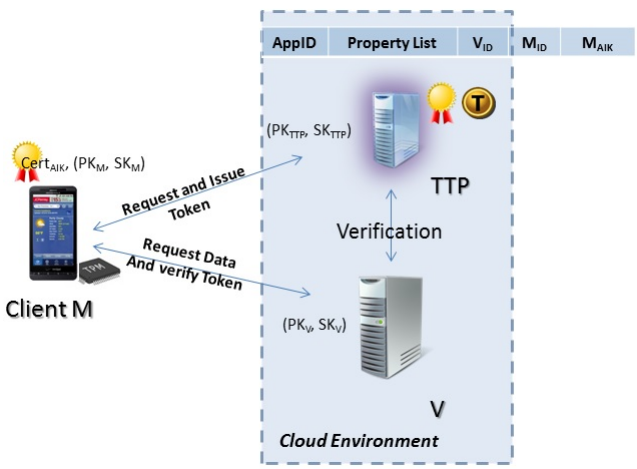


Figure 3. Property based Token Attestation

4.2. Assumptions

In this work, 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 the TPM is equipped for all involved parties, 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 for the Trusted Third Party (TTP)
- V and TTP set a security policy for the employee 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. 3).
- 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 demonstrated in the figure 4, 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.

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

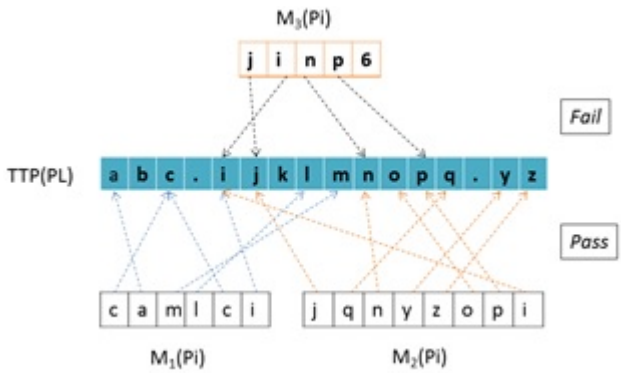


Figure 4. Property evidence

4.3. 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 5, 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) [15, 16, 17]. Hence, in this context, we assume that TPM has its own Certificate of AIK.

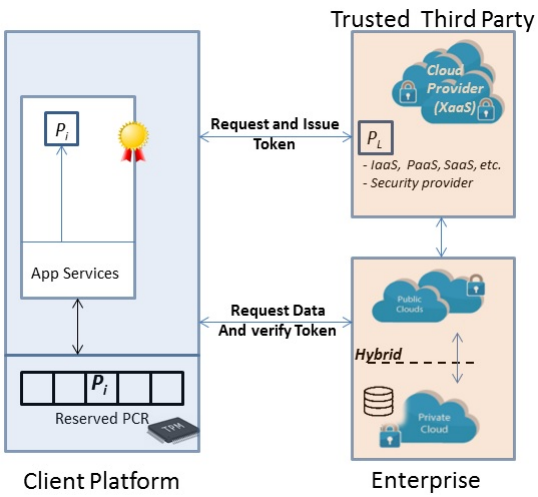


Figure 5. Property based Token Attestation model

4.4. Property based Token Attestation Protocol

We start the section 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 4.4.2.

4.4.1. General Overview of the PTA protocol. Before discussing the structure of the proposed protocol, we recall the following notations that have been used in this article.

- 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 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.
- Application 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.
- $(Hash(Message), Message)P_{KX}$ 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.
- 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 [18] between X and Y only.
- ML is the measurement list that contains the measurement value of all the entity 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.
- $(Message)P_{KX}$ is the Public-Key Cryptosystem for secure data transmission among involved parties.

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. Within the scope of the paper, we do not discuss the process to obtain the Certificate of AIK that is described in detail [17].

We assume that the user logs in with his 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.

4.4.2. Dataflow of PTA protocol. To depict the general overview of PTA protocol, the message sequence is represented as the following chart.

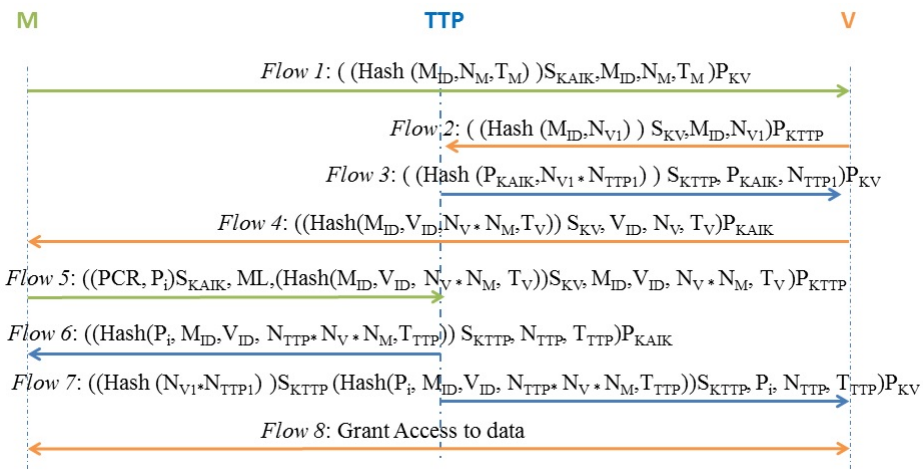


Figure 6. The overview message exchange of PTA protocol

In which, 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 either to M or to 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 can decrypt this message using his private key. However, due to a lack of knowledge of $CERT_{AIK}$, V cannot decrypt 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 PCA 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 he 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 he 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 (authentic TPM and M) received in *flow 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.

4.4.3. 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 can be adapted to 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}.

5. 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=$((Hash(M_{ID}, V_{ID}, N_V * N_M, T_V))S_{KV}, V_{ID}, N_V, T_V)$
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.
   $d' = [D]S'_{KAIK} = (A', V'_{ID}, N'_V, T'_V)$
   $[A' = decrypted\ value\ of\ Hash(M_{ID}, V_{ID}, N_V * N_M, T_V))S_{KV}\ using\ S'_{KAIK}]$
   Whereas, if message D is decrypted by S_{KAIK}, then we would get
   $d = [D]S_{KAIK} = (A, V_{ID}, N_V, T_V)$
   $[A = decrypted\ value\ of\ Hash(M_{ID}, V_{ID}, N_V * N_M, T_V))S_{KV}\ using\ S_{KAIK}]$
   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, $[A]P_{KV} = hv \ldots .(i)$
   Legitimate M knows M_{ID}, N_M.
   So, it calculates $Hash(M_{ID}, V_{ID}, N_V * N_M, T_V) = hv \ldots .(ii)$
   From (i) and (ii), 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, $Hash(M'_{ID}, V'_{ID}, N'_V * N'_M, T'_V) = hv1' \neq hv' \neq hv$. 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 MID, $Hash(M_{ID}, V'_{ID}, N'_V * N'_M, T'_V) = hv2' \neq hv' \neq hv$.

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_{KTTP}.
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_{KTTP}. 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. 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*)
   $((Hash(M_{ID}, N_M, T_M))S_{KAIK}, M_{ID}, N_M, T_M)P_{KV}$
   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:$[((Hash(M'_{ID}, N'_M, T'_M))S'_{KAIK}, M'_{ID}, N'_M, T'_M)P_{KV}]S_{KV}$
   $= ((Hash(M'_{ID}, N'_M, T'_M))S'_{KAIK}, M'_{ID}, N'_M, T'_M)$
   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:$[((Hash(M'_{ID}, N_{V1}))S_{KV}, M'_{ID}, N_{V1})P_{KTTP}]S_{KTTP}$
   $= ((Hash(M'_{ID}, N_{V1}))S_{KV}, M'_{ID}, N_{V1})$ 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: $CERT_{AIK}[M'_{ID}]$ 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. $V : [((Hash(P'_{KAIK}, N_{V1} * N_{TTP1}))S_{KTTP}, P'_{KAIK}, N_{TTP1})P_{KV}]S_{KV}$
   $= ((Hash(P'_{KAIK}, N_{V1} * N_{TTP1}))S_{KTTP}, P'_{KAIK}, N_{TTP1})$
   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: $[((Hash(M'_{ID}, V_{ID}, N_V * N'_M, T_V))S_{KV}, V_{ID}, N_V, T_V)P'_{KAIK}]S'_{KAIK}$
   $= ((Hash(M'_{ID}, V_{ID}, N_V * N'_M, T_V))S_{KV}, V_{ID}, N_V, T_V)$, 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: $[((Hash(M'_{ID}, V_{ID}, N_V * N'_M, T_V))S_{KV}, V_{ID}, N_V, T_V)P'_{KAIK}]S_{KAIK}$
   $= (A', V'_{ID}, N'_V, T'_V),\ where\ A' = [(Hash(M'_{ID}, V_{ID}, N_V * N'_M, T_V))S_{KV}]S_{KAIK}$. 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: $Hash(M_{ID}, V'_{ID}, N'_V * N_M, T_V) = h'' \neq h'$. 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:$[((PCR', Pi')S'_{KAIK}, Pi', ML', (Hash(M'_{ID}, V_{ID}, N_V * N'_M, T_V))S_{KV}, M'_{ID}, V_{ID}, N_V * N'_M, T_V)P_{KTTP}]S_{KTTP}$
    $= ((PCR', Pi')S'_{KAIK}, Pi', ML', (Hash(M'_{ID}, V_{ID}, N_V * N'_M, T_V))S_{KV}, M'_{ID}, V_{ID}, N_V * N'_M, T_V)$.
    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}]
    $= Hash(PCR', Pi') = x1$.
    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: $Hash(PCR1, Pi1) = x2 \neq x1$.
    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: $[((Hash(M_{ID}, N_M, T_M))S_{KAIK}, M_{ID}, N_M, T_M)P_{KV}]S_{KV}$
   $= ((Hash(M_{ID}, N_M, T_M))S_{KAIK}, M_{ID}, N_M, T_M)$. 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: $[((Hash(M_{ID}, N_M, T_M))S_{KAIK}, M_{ID}, N_M, T_M)P_{KV}]S_{KV}$
   $= ((Hash(M_{ID}, N_M, T_M))S_{KAIK}, M_ID, N_M, T_M)$
   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: $[(PCR, Pi')S_{KAIK}, Pi', ML, (Hash(M_{ID}, V_{ID}, N_V * N_M, T_V))S_{KV}, M_{ID}, V_{ID},$
$N_V * N_M, T_V)P_{KTTP}]S_{KTTP}$
$= ((PCR, Pi')S_{KAIK}, Pi', ML, (Hash(M_{ID}, V_{ID}, N_V * N_M, T_V))S_{KV}, M_{ID}, V_{ID},$
$N_V * N_M, T_V)$
TTP now has $Pi', ML, M_{ID}, V_{ID}, N_V * N_M, T_V$.
TTP: $[(PCR, Pi')S_{KAIK}]P_{KAIK} = Hash(PCR, Pi') = y1$. [Hash extend operation]
[$CERT_{AIK}$ tells that M has public key P_{KAIK} and secret key S_{KAIK}]
TPM has safely stored ML corresponding to M and the property Pi corresponding to valid app in M is also stored in TTP.
TTP: $Hash(PCR, Pi) = y2 \neq y1$, 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 send 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: $[((Hash(Pi, M_{ID}, V_{ID}, N_{TTP} * N_V * N_M, T_{TTP}))S_{KTTP}, N_{TTP}, T_{TTP})P_{KAIK}]S_{KAIK}$
   $= ((Hash(Pi, M_{ID}, V_{ID}, N_{TTP} * NV * NM, T_{TTP}))S_{KTTP}, N_{TTP}, T_{TTP})$
   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 Step 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 EPROM.
4. We assume that, at time T_Y, TTP received the token request message from M:
   $[(PCR, Pi)S_{KAIK1}, Pi, ML, (Hash(M_{ID}, V_{ID}, N_{V2} * N_{M2}, T_{V2}))S_{KV}, M_{ID}, V_{ID},$
   $N_{V2} * N_{M2}, T_{V2})P_{KTTP}]S_{KTTP}$
   $= ((PCR, Pi)S_{KAIK1}, Pi, ML, (Hash(M_{ID}, V_{ID}, N_{V2} * N_{M2}, T_{V2}))S_{KV}, M_{ID}, V_{ID},$
   $N_{V2} * N_{M2}, T_{V2})$.
   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: $[(PCR, Pi)S_{KAIK1}]P_{KAIK1} = Hash(PCR, Pi) = y \ldots .(iii)$ [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. . . .(iv)
From (iii) and (iv), 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: $[((Hash(Pi, M_{ID}, V_{ID}, N_{TTP} * N_V * N_M, T_{TTP}))S_{KTTP}, N_{TTP}, T_{TTP})$
   $P_{KAIK1}]S_{KAIK1}$
   $= (Hash(Pi, M_{ID}, V_{ID}, N_{TTP} * N_V * N_M, T_{TTP}))S_{KTTP}, N_{TTP}, T_{TTP})$
   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: $[((Hash(Pi, M_{ID}, V_{ID}, N_{TTP2} * N_{V2} * N_{M2}, T_{TTP2}))S_{KTTP}, N_{TTP2}, T_{TTP2})$
   $P_{KAIK1}]SK_{AIK1}$
   $= (Hash(Pi, M_{ID}, V_{ID}, N_{TTP2} * N_{V2} * N_{M2}, T_{TTP2}))S_{KTTP}, N_{TTP2}, T_{TTP2})$.
   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.

6. SECURITY DISCUSSION

To address the common threats in section 3, 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 [6] and [8], we discuss how properties can be evaluated as presented in Section 4. 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 [18]. It is a tool for the formal analysis of security protocols under the perfect cryptography assumption. Scyther uses the Dolev-Yao adversarial model [9]. 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 [19] 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.

7. RELATED WORK

The proliferation of mobile devices has changed the requirement of computing ecosystem. The end-user 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, remote attestation is one of the important 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 [20, 21, 22]. In other hand, some researchers argue that this attestation mechanism has its own shortcomings, such as privacy, flexibility and scalability [11, 12, 14, 23]. For this reason, the literature [14, 24, 25] 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. [26] 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 [25]. To consider the problems of trust in cloud monitoring system, Awad et al. [27] 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. [28] 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. [29] 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. Excepting the method accounts for the security properties, this work has some similar points with ours as it will be discussed in high level in the Section 8 where we come up with the fact that *Virtual Machines is everywhere* term for discussion.

In the mobile computing context, although without the existence of hardware root of trust, Mohammad et al. [30] 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 [31] 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. [32] 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 which, they omitted a trusted party which is responsible for translation between software measurement and properties.

All in all, the pros and cons of most existed approaches in remote attestation are summarized in [13]. 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 how to generate them automatically have not been answered sufficiently by the above mentioned works. This is the major difference of our work in comparison with others. In our work, we consider everything is property i.e. functions, attributions,

re-computed nonce, etc. We propose to use a token with random properties to authenticate mutually the mobile user and the involved parties. While not mentioning clearly about the token based authentication [31], we define in detail the token content and how it works. Due to the revocation and the update of property that happen highly in nature, the approach without trusted third party [33] may increase the complexity of the system when the verifier is not aware of revoked properties in real time [34]. To overcome this, we introduce the random properties list which is generated by the application service automatically, regardless the update or backup. Its integrity is ensured and verified by a reserved PCR and an active TTP respectively. Last but not least, since TCG has introduced the second generation of TPM [35], TPM 2.0, with more functionalities to support mobile devices in 2014, this makes the existence of hardware root of trust in mobile devices to be more popular. We also take this opportunity for the feasibility of our research.

8. CONCLUSION AND FUTURE WORK

By considering the token with random property, our work 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 paper 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 analysed it with the security proofs. In this paper, our work 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.

This work is the foundation for the next steps where we consider the term of *Virtual Machines is everywhere*, since Berger et al. [36] designed a virtual TPM (vTPM) architecture enabling physical TPM to be served by multi virtual environments on the trusted platform and the approach in [37] allowing to implement property-based sealing and attestation in vTPM. Our future work is to adapt this PTA protocol to secure another mobile cloud computing model based on the Cloudlet notion. In that work, we aim to make our solution more adaptive and context-aware by relying on the reputation of the Cloudlets.

REFERENCES

1. Henky Agusleo & Neeraj Arora, *The Road to Cloud Nine' How Service Providers Can Monetize Consumer Mobile Cloud*, Whitepaper, Cisco Internet Business Solutions Group (IBSG), Feb. 2013.
2. T. Le Vinh, S. Bouzefrane, J.-M. Farinone, A. Attar, and B. P. Kennedy, *Middleware to Integrate Mobile Devices, Sensors and Cloud Computing,* Procedia Comput. Sci., vol. 52, pp. 234243, 2015.
3. *Best Practices for Enabling Employee-owned Smart Phones in the Enterprise*, Intel Whitepaper Dec, 2011 .
4. *Best Practices for Enabling Employee-owned Smart Phones in the Enterprise*, Intel Whitepaper Feb 2012 .
5. A. M. D. O. S. Hofmann and B. W. E. Witchel, *Cloaking Malware with the Trusted Platform Module*. SEC 2011 Proceedings of the 20th USENIX conference on Security.
6. A. N. Khan, M. L. Mat Kiah, S. U. Khan, and S. A. Madani, *Towards secure mobile cloud computing: A survey,* Future Gener. Comput. Syst., vol. 29, no. 5, pp. 12781299, Jul. 2013.
7. S. Pearson, *Trusted Computing Platforms: TCPA Technology in Context*. Upper Saddle River, NJ, USA: Prentice Hall PTR, 2002.

8. K. N. McGill, *Trusted mobile devices: Requirements for a mobile trusted platform module,* Johns Hopkins Apl Tech. Dig., vol. 32, no. 2, p. 544, 2013.
9. S. Bouzefrane and L. V. Thinh, *Trusted Platforms to Secure Mobile Cloud Computing,*IEEE HPCC 2014, pp. 10681075.
10. T. C. G. Admin, *TPM Library Specification, Trusted Computing Group*, 01-Oct-2014. .
11. L. Chen, R. Landfermann, H. Lhr, M. Rohe, A.-R. Sadeghi, and C. Stble, *A Protocol for Property-based Attestation,* in Proceedings of the First ACM Workshop on Scalable Trusted Computing, New York, NY, USA, 2006, pp. 716.
12. A. Nagarajan, V. Varadharajan, M. Hitchens, and E. Gallery, *Property Based Attestation and Trusted Computing: Analysis and Challenges,* 2009, pp. 278285.
13. R. V. Steiner and E. Lupu, *Attestation in Wireless Sensor Networks: A Survey,* ACM Comput. Surv., vol. 49, no. 3, pp. 131, Sep. 2016.
14. A.-R. Sadeghi and C. Stble, *Property-based Attestation for Computing Platforms: Caring About Properties, Not Mechanisms,* in Proceedings of the 2004 Workshop on New Security Paradigms, New York, NY, USA, 2004, pp. 6777.
15. E. Brickell, J. Camenisch, and L. Chen, *Direct anonymous attestation,* in Proceedings of the 11th ACM conference on Computer and communications security, 2004, pp. 132145.
16. B. Smyth, M. Ryan, and L. Chen, *Direct Anonymous Attestation (DAA): Ensuring privacy with corrupt administrators,* in European Workshop on Security in Ad-hoc and Sensor Networks, 2007, pp. 218231.
17. G. Coker, J. Guttman, P. Loscocco, J. Sheehy, and B. Sniffen, *Attestation: Evidence and trust,* in International Conference on Information and Communications Security, 2008, pp. 118.
18. C. Cremers and S. Mauw, *Operational Semantics and Verification of Security Protocols*. Berlin, Heidelberg: Springer Berlin Heidelberg, 2012.
19. C. J. Cremers, *The Scyther Tool: Verification, falsification, and analysis of security protocols,* in International Conference on Computer Aided Verification, 2008, pp. 414418.
20. H. Tan, W. Hu, and S. 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, 2011, pp. 916.
21. J. G. Beekman, J. L. Manferdelli, and D. Wagner, *Attestation Transparency: Building secure Internet services for legacy clients,* 2016, pp. 687698.
22. D. Fu and X. Peng, *TPM-based remote attestation for Wireless Sensor Networks,* Tsinghua Sci. Technol., vol. 21, no. 3, pp. 312321, 2016.
23. T. Rauter, A. Hller, N. Kajtazovic, and C. Kreiner, *Privilege-Based Remote Attestation: Towards Integrity Assurance for Lightweight Clients,* 2015, pp. 39.
24. E. Gallery, A. Nagarajan, and V. Varadharajan, *A Property-Dependent Agent Transfer Protocol,* in Trusted Computing, vol. 5471, L. Chen, C. J. Mitchell, and A. Martin, Eds. Berlin, Heidelberg: Springer Berlin Heidelberg, 2009, pp. 240263.
25. N. Borhan and R. Mahmod, *Platform Property Certificate for Property-based Attestation Model,* Int. J. Comput. Appl., vol. 65, no. 13, 2013.
26. S. Xin, Y. Zhao, and Y. Li, *Property-Based Remote Attestation Oriented to Cloud Computing,* 2011, pp. 10281032.
27. A. Awad, S. Kadry, B. Lee, and S. 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, 2014, pp. 934940.
28. Z. Liu, X. Wang, Y. Liu, D. Guo, and X. Zhu, *Client Oriented Remote Attestation Model in Cloud Environment,* Int. J. Secur. Its Appl., vol. 9, no. 10, pp. 395404, Oct. 2015.
29. V. Varadharajan and U. Tupakula, *Counteracting security attacks in virtual machines in the cloud using property based attestation,* J. Netw. Comput. Appl., vol. 40, pp. 3145, Apr. 2014.
30. M. Nauman, S. Khan, X. Zhang, and J.-P. Seifert, *Beyond kernel-level integrity measurement: enabling remote attestation for the android platform,* in International Conference on Trust and Trustworthy Computing, 2010, pp. 115.
31. Aurelien Francillon, Quan Nguyen, Kasper B.Rasmussen, Gene Tsudik, *A minimalist approach to remote attestation*, 2015. DATE'14 Proceedings of the conference on Design, Automation & Test in Europe .
32. K. Kostiainen, N. Asokan, and J.-E. Ekberg, *Practical Property-Based Attestation on Mobile Devices,* in Trust and Trustworthy Computing, vol. 6740, J. M. McCune, B. Balacheff, A. Perrig, A.-R. Sadeghi, A. Sasse, and Y. Beres, Eds. Berlin, Heidelberg: Springer Berlin Heidelberg, 2011, pp. 7892.
33. L. Chen, H. Lhr, M. Manulis, and A.-R. Sadeghi, *Property-Based Attestation without a Trusted Third Party,* in Information Security, vol. 5222, T.-C. Wu, C.-L. Lei, V. Rijmen, and D.-T. Lee, Eds. Berlin, Heidelberg: Springer Berlin Heidelberg, 2008, pp. 3146.

34. Y. Qin, D. Chang, S. Zhao, and Q. Zhang, *A Property-Based Attestation Scheme with the Variable Privacy,* 2011, pp. 16161623.
35. *TPM Library Specification,* Trusted Computing Group, 01-Oct-2014. .
36. R. Perez, R. Sailer, L. van Doorn, and others, *vTPM: virtualizing the trusted platform module,* in Proc. 15th Conf. on USENIX Security Symposium, 2006, pp. 305320.
37. A.-R. Sadeghi, C. Stble, and M. Winandy, *Property-based TPM virtualization,* in Information Security, Springer, 2008, pp. 116.