Kim et al. EURASIP Journal on Wireless Communications and Networking (2016) 2016:233
DOI 10.1186/s13638-016-0729-7

EURASIP Journal on Wireless Communications and Networking

RESEARCH Open Access

Secure user authentication based on the trusted platform for mobile devices

GeonLyang Kim*, JaeDeok Lim and JeongNyeo Kim*

Abstract

In recent years, the use of mobile devices including smartphones has increased significantly all over the world, and e-commerce using smartphones has also greatly increased. Furthermore, many people are using their smartphones to carry out certain aspects of their work according to the BYOD trend. Therefore, it is extremely important that mobile device users are authenticated securely by remote servers when using their smartphones. Digital certificates are one of the many solutions available for authentication, but they are easy to copy and leak. Mobile device services need to properly manage registered devices and users, and trusted means of authenticating their identities are needed. In this paper, we propose a secure certificate-based user authentication framework using the trusted mobile zone (TMZ) system into which the trusted platform is built. The TMZ system is a secure mobile device into which a hypervisor is built on the mobile device, and in which the hypervisor separates the mobile device into a normal zone and a secure zone. Android OS operates in the normal zone on the TMZ systems, and secure OS is run in the secure zone at the same time. The trusted platform is built in the normal zone and the secure zone in order to provide the user with secure services. In this paper, we propose a TMZ system founded on the TEE system of the global platform. The TMZ system provides a secure execution environment in which to store sensitive data and execute security functions securely. In conclusion, we describe the experimental results of generating the signature data in the TMZ system.

1 Introduction

The use of smartphones has increased dramatically in the last few years. Indeed, the report of Strategy Analytics stated that universal smartphone shipments had increased by 12 % each year, reaching the record figure of 1.4 billion units in 2015 [1]. The IDC study published a report on global smartphone shipments over the next 5 years, predicting that worldwide smartphone shipments will hit a record 1.86 billion units by 2019 [2].

The use of mobile devices such as smartphones, laptops, and tablets has increased significantly and the concept of cloud computing has emerged, coming under the spotlight as a latent technology [3]. Mobile cloud computing technology enables cloud computing in the mobile environment and tries to resolve various problems related to security, performance, and environment among others. Cloud computing in the mobile environment is threatened by various forms of cyber attack such as malware, spyware, and viruses because mobile devices are linked to the network and widely used. Thus, the security requirements for mobile devices have been increasing in line with the progress of mobile cloud computing. Numerous studies about mobile security programs for trusted mobile cloud computing have been carried out [4–7], while virtualization technologies for cloud computing have attracted the attention of the IT industry and the business world.

A lot of organizations would like to introduce bring your own device (BYOD) whereby employees utilize their privately owned smart devices at work. However, there are extremely significant challenges as employees very often use their own smart devices at work; and interest in how to prevent the leakage of security data of their enterprises from malicious applications or codes has grown considerably. Many organizations are now looking at mobile virtualization as a solution to the significant challenges posed by the BYOD culture. By concurrently running two operating systems on a smartphone, one can execute personal apps and services in

* Correspondence: glkim@etri.re.kr; jnkim@etri.re.kr
Information Security Research Division, Electronics and Telecommunications Research Institute, Daejeon, South Korea

one OS and business services in a more secure OS that is isolated from it.

Researchers have already confirmed the great characteristics of virtualization technologies from the security perspective. The most notable characteristic is a feature that ensures isolation between virtual machines that are run concurrently on the same physical system [4], and the isolation of the hypervisor responsible for managing the virtual machines from the virtualized operating systems. The perfect isolation of the virtual machines and the hypervisor managing them provides a secure framework in which the security system continues to run correctly even when numerous malicious codes exist in the virtual machines.

Since the development of mobile communications and the dissemination of mobile devices have rapidly grown and the services based on mobility have been increased, the authentication of users or mobile devices in mobile networks is very important. Traditional authentication methods such as the username and password are used in many cases of user authentication by a remote authentication server. But there are drawbacks to using password-based authentication. For example, the authentication system requires long and complex passwords to prevent attackers from cracking or guessing them. In addition, it makes it more difficult for users to recall their passwords. This raises the possibility that the users will not manage their passwords securely and may easily reveal them to attackers. There are lots of password threats, such as password capturing, password guessing, password cracking, and password replacing [8]. The move to a system of password-free login using biometric authenticators is being accelerated through standardization by the FIDO Alliance. Recently, the ARM TrustZone-based Trusted Execution Environment (TEE) was designed to deliver enhanced security from scalable software attacks and common hardware attacks at a lower cost to the market, and to provide solutions for the FIDO security requirements, such as to ensure the integrity of the device, to keep key material confidential from unauthorized access, to maintain the confidentiality and integrity of sensitive processes, to maintain the confidentiality of sensitive input data, and to protect sensitive display data [9].

The RADIUS protocol encrypts and transmits user passwords between the client and the RADIUS server in order to eliminate the possibility of user passwords being exposed, although it might not offer sufficient security for user passwords [10]. The use of user passwords on networks where there are many threats always carries with it the possibility that the passwords will be exposed. The Transport Layer Security (TLS) protocol uses the X.509 certificate to authenticate the client and the server [11]. As such, it needs a method of management that will protect the certificate file and the private key pair securely. For certificate-based authentication of the users or mobile devices in mobile network environments, because the certificate and the private key pair are frequently spill due to the vulnerabilities of mobile devices, the theft and misuse of the certificate pairs can undermine the trust of certificate-based authentication. Therefore, a method is needed to increase the reliability of the authentication process.

In this paper, we describe a TEE-based Trusted Mobile Zone (TMZ) system that builds the trusted platform into mobile devices using virtualization technology, and propose a secure certificate-based user authentication framework using the TMZ system that can increase the reliability of the authentication process performed for mobile devices in mobile networks. The TMZ system with trusted platform is a good solution to enhance mobility management security. The protection of the private key through hardware such as TrustZone or HSM can provide more secure authentication, but the TMZ system coupled with the trusted platform provides a structure that increases the strength of security through software only, without the cost of additional hardware.

2 Background

2.1 Assurance level of certificate-based user authentication

The NIST published SP 800-63, which offers technical guidelines for federal agencies that are developing electronic authentication. It describes the authentication of remote internet-based users such as employees or of private individuals attempting to connect with the IT systems of government organizations over open networks. It also describes the technical requirements for each of four levels of assurance in the fields of authentication protocols, identity proofing, management processes, and so on. The SP 800-63 guidelines make up for OMB guidance, serving as e-authentication guidance for federal agencies. The OMB guidance describes the definition of the required level of authentication assurance related to the risks from an authentication error, provides the criteria for deciding the level of e-authentication assurance demanded for specific applications and transactions for organizations, founded on their risks and their possibility of occurrence, and also provides the technical requirements for choosing the authentication solutions using the risk assessment. The OMB guidance describes briefly a five-step process by which organizations should decide their levels of e-authentication assurance. The NIST published SP 800-63 to provide guidelines on implementing the third step [12].

Step 1. Conduct a risk assessment of the government system.

Step 2. Map the identified risks to the appropriate assurance level.
Step 3. Select the technology based on e-authentication technical guidance.
Step 4. Confirm whether the implemented system has met the required assurance level.
Step 5. Periodically reassess the information system to determine the technology refresh requirements.

Organizations can choose e-authentication technologies that satisfy the requirements of the necessary assurance level after conducting a risk assessment for the target system and comparing the identified risks of the target system and the appropriate assurance level. OMB guidance describes the definition for each of four assurance levels, of which level 1 is the highest and level 4 the lowest level of assurance. The traditional mode of authentication system asserts the following three elements as the foundation of authentication: "something you know," "something you have," and "something you are". The first of these, "something you know," is the knowledge factor (e.g., a password, a PIN), while "something you have" is the possession factor (e.g. an ID badge or cryptographic key) and "something you are" is the inherence factor (a fingerprint, other biometric data).

Multi-factor authentication refers to the utilization of more than one of the three elements mentioned above [12]. The strength of the authentication system is primarily determined by the number of elements integrated by the system. For example, systems integrating two elements are regarded as being stronger than those which utilize only one element, and systems that integrate all three elements are stronger than those which only integrate two elements. It is not multi-factorial authentication that the "something you know" includes several things.

A cryptographic key is saved on a disk or on other media and needs to be triggered through the possession factor of authentication in order to utilize it. The authentication is achieved by attesting possession and control of the key. The authenticator is extremely dependent on the specific cryptographic paradigm, but it is generally some type of signed message. For instance, the multi-factor software cryptographic token is the possession factor in a "certificate verify" message of the TLS protocol, and the token can be triggered by the knowledge factor or the inherence factor.

Among the four assurance levels of the OMB guidance, level 3 offers multi-factor authentication on remote network-based IT systems, demanding at least two elements of authentication. Authentication is founded on attestation of the possession of authentication tokens through a cryptographic paradigm. It demands cryptographic strength paradigms that preserve the authentication tokens against not only all the threats at level 2 but also against verifier impersonation attacks.

The uses of electronic transactions on smartphones have increased significantly, and the certificate-based electronic signature authorization system is built by using the public key infrastructure (PKI) for secure information transmission and identification, and then the service provider. In this paper, we describe the secure certificate-based user authentication method. The certificate is a multi-factorial software cryptographic token; and, in this respect, certificate-based user authentication can be called "multi-factorial authentication" because the user has the certificate file and knows the password required to decrypt the private key [12, 13].

"Something you have," as the second authentication factor, means that the user has to have a hardware type (e.g., OTP device) or software type (e.g., digital certificate) of authentication token. The hardware type of token offers low portability and convenience because the user owns the physical form, whereas the software type of token can compensate for the former's disadvantage of low portability and convenience. That said, as it is stored in a logical form, it has the drawback of carrying a high risk of leakage.

2.2 Trusted Execution Environment

The TMZ system is designed based on the TEE system of the global platform and implements the secure zone as software. The TMZ system constructs the secure zone and the normal zone in smart mobile devices through hypervisor technologies. The android OS runs in the normal zone, and the secure OS runs in the secure zone, which is separated from the normal zone. The TMZ system blocks the leakage of data from the secure zone.

ARM works together with numerous cooperative relationships and industry forum committees to carry forward the standards. ARM has joined hands with the global platform in order to standardize security and certification [14, 15]. In fact, the global platform sets the definition of the TEE client API and the TEE internal API back in 2010 and 2011, but the TEE functional API has not yet been defined [16–18].

The TEE software architecture is designed so that the trusted applications offer separated and reliable capabilities that can be used by service providers via the client applications [19]. The relationship between the major software systems components is outlined in the block architecture as shown in Fig. 1. The TEE is an isolated and secure execution environment that operates with the rich OS concurrently and provides trustworthy services to the client applications in that environment. The TEE exposes sets of APIs to enable communication from the rational expectations equilibrium (REE) and others

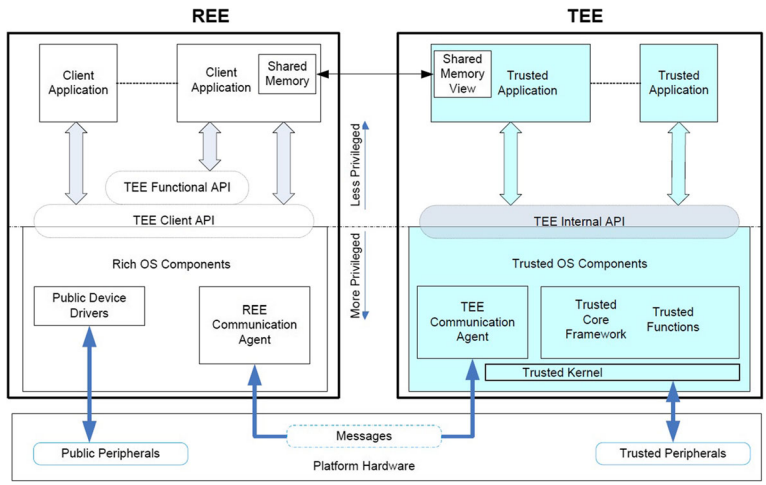

Fig. 1 TEE software architecture

to guarantee the functionality of the trusted application software within the TEE [19].

The TEE functional API will offer to the client applications various REE software interfaces that will be the programming interfaces dedicated to exposing inbuilt TEE capabilities for the client application developers [19]. The TEE client API executes efficient communications between client applications and trusted applications [17]. The client applications of REE can access the TEE components and exchange data with the trusted applications of TEE through the TEE client API as a low-level communication interface. The REE communication agent provides REE support for messaging between the client application and the trusted application.

The Trusted Core Framework provides the OS like functionality to trusted applications, while the trusted functions provide support facilities for application developers. Both the trusted functions and the Trusted Core Framework make use of the scheduling function and other OS management functions provided by the Trusted Kernel [19]. The TEE communication agent is a special case of a Trusted Core Framework function API. It works with its peer, the REE communication agent, to safely transfer messages between the CA and the TA. The trusted applications interface to the rest of the system via the APIs exposed by trusted OS components. The TEE internal API defines the fundamental software capabilities of a TEE [18]. Other APIs may be defined to support interface to further proprietary trusted functions.

When a client application creates a session with a trusted application, it connects to an "Instance" of that trusted application. A trusted application instance has a physical memory space which is separated from the physical memory space of all the other trusted application instances. A session is used to logically connect multiple commands invoked in a trusted application. Each session has its own state, which typically contains the session context and the context of the task executing the session.

2.3 Mobile virtualization: hypervisor

In recent years, many mobile virtualization technologies have progressed to the point where they can handle the policy configuration for detailed mobile device management and the complete separation of containers using the hypervisor, which is also called a virtual machine monitor. The hypervisor is a thin layer of codes that allows several virtual machines to run concurrently on a single physical computer. It utilizes low-level codes in software or firmware to allocate the resources of a physical computer to several virtual machines in real time. The hypervisor exists in two forms, type 1 and type 2, as shown in Fig. 2. Type 1 supports hardware virtualization because it runs directly on the hardware, while type 2 performs software virtualization because it runs as an application on the top of a host operating system environment.

The type 1 hypervisor operates on the next above the hardware of mobile devices. Thus, it can access hardware resources such as CPU or USIM directly. It is sometimes called a "native hypervisor" or a "bare metal hypervisor". The type 1 hypervisor can control several virtual machines and offers complete isolation and security functions compared to others of its kind. As such, a mobile device based on the type 1 hypervisor can optimize the performance of each of the virtual machines. As for the type 2 hypervisor, it runs as an application on top of the host mobile operating system environment at the second level and on the guest operation systems that operate on the next above the type 2 hypervisor. The performance and compromise of the

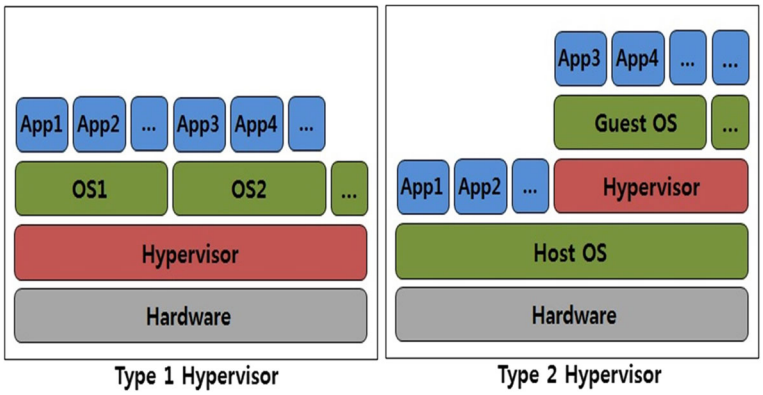

Fig. 2 Hypervisor types

guest OS are heavily dependent on the host OS. It is inherently less secure and may run slower than native apps because it does not work at the hardware level. The hypervisor technology has a drawback in that it has to be partnered and perform the work with OEMs. Recently, many smartphones have not been able to offer the hardware level of virtualization [20].

3 Proposed framework

3.1 Trusted mobile zone system

The TMZ system architecture described in this paper is similar to the TEE system architecture [19], and the secure zone of TMZ system is embodied in the software rather than the hardware. The following Fig. 3 depicts the architecture of the TMZ system.

The TMZ system separates the normal zone and secure zone through the hypervisor as shown in Fig. 3. In addition, the system builds the trusted platform consisting of the normal zone and the secure zone on smart mobile devices. The trusted platform, which consists of the AC agent, NZ-security API, TCM API, NZ communication agent, SZ communication agent, SZ-security API, and secure functions, ensures that the services requested from the security applications in the normal zone are securely executed in the secure zone. We call a smart mobile device in which the secure zone is isolated from the normal zone and which is equipped with the trusted platform the TMZ system.

The open OS runs in the normal zone, while the secure OS runs concurrently in the secure zone isolated from the normal zone. The open OS of the TMZ system is Android OS. The trusted platform of the normal zone consists of the AC agent, NZ-security API, TCM API, and NZ communication agent, whereas the trusted platform of the secure zone consists of the SZ communication agent, SZ-security API, and secure functions. If we

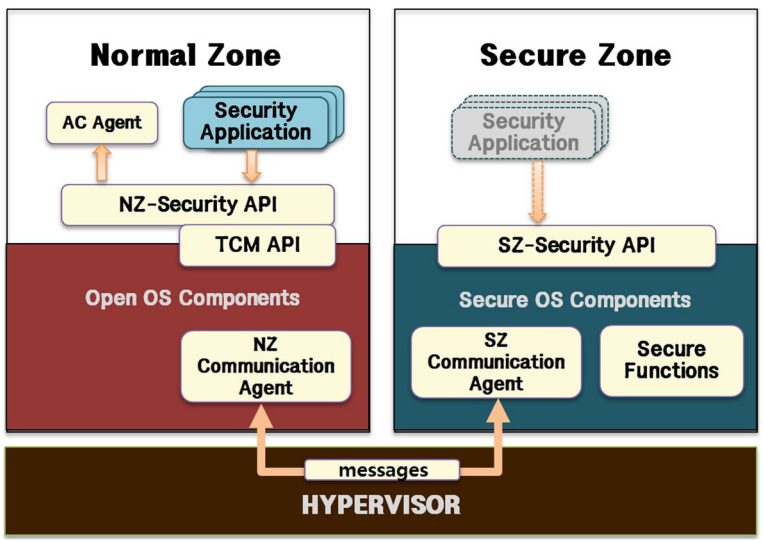

Fig. 3 TMZ system architecture

compare the architectures of the TMZ system and the TEE system, the NZ-security API of the TMZ system may be mapped to the TEE functional API of the TEE system. The client application and the trusted application may be mapped to the security application in the normal zone and the security application in the secure zone. The secure functions, SZ-security API, SZ communication agent, NZ communication agent, and TCM API may be mapped to the trusted functions, TEE internal API, TEE communication agent, REE communication agent, and TEE client API. The AC agent exists only in the TMZ system and not at all in the TEE system.

Each trusted application exposes a TA interface. Trusted core framework implementation calls the TA interface to relay communication between the client application and the trusted application. Once the trusted core framework has called one of the TA entry points, the TA can make use of the TEE internal API to access the facilities of the trusted OS. The TEE internal API consists of the trusted core framework API, trusted storage API for data and keys, cryptographic operations API, time API, and TEE arithmetical API [19]. The TA interface is a subgroup of the trusted core framework API. The TEE internal API can be said to consist of the TA interface and other APIs to access the facilities of the trusted OS. If the client application in the normal zone requests secure services through the REE communication agent, the trusted core framework implementation calls the TA interface, and the TA may use the TEE internal API to access the facilities of the trusted OS, including the trusted functions.

Now, the TMZ system does not consider security applications in the secure zone. So, the TMZ system has no SZ-security API for security applications of the secure zone mapped as the TA interface for the trusted application of the TEE system. In other words, when security applications of the normal zone request the secure services through the NZ communication agent, secure OS components do not call the interface of the security application in the secure zone but rather the SZ-security API in order to access the facilities of the secure, OS including the secure functions.

The security applications of the normal zone (required to use the secure functions of the secure zone are accessible only through the NZ-security API. The trusted platform of the TMZ system is independent on the hypervisor type. ETRI's VIMO has the type 1 and type 2 hypervisors. The type 1 hypervisor is more secure than the type 2 hypervisor, but the former is less flexible than the latter because it is easily ported and built, depending on the upgrade of the Android OS. We built the trusted platform in the TMZ system, with VIMO included, like ETRI's type 2 hypervisor built into the Galaxy S3, and installed the μC/OS-II as a secure OS in the secure zone. However, we have been unable to install and operate security applications in the secure zone up to the present time.

The security applications of the normal zone offer users secure services using the secure functions of the secure zone, unlike other applications of the normal zone. The security applications of the normal zone use the NZ-security API to access the secure functions of the secure zone and transmit requests into the secure zone through the TCM API and the NZ communication agent. The NZ communication agent calls the secure functions through the SZ communication agent and the SZ-security API, receives the return value of the SZ-security API from the SZ communication agent, and transmits it to the security applications of the normal zone through the NZ communication agent. In this paper, we do not consider ways of protecting the sensitive data of security applications from the malicious code in the normal zone. Methods of protecting the sensitive data of security applications—such as reverse engineering prevention, forgery prevention, and code obfuscation—are needed because the sensitive data transmitted from the NZ-security API through the encrypted channel are decrypted at security applications, and the data thus decrypted are then provided to the user while he is using the secure services. The sensitive data of the security applications include the encryption keys for encrypting channels between the normal zone and the secure zone. Such encryption keys are created in the secure zone and shared between the normal zone and the secure zone before distributing them to the TMZ systems. We are currently designing a key management mechanism that creates and exchanges new encryption keys for new sessions between the normal zone and the secure zone.

When the security application of the normal zone calls the NZ-security API in order to use the secure functions of the secure zone, the procedure of the TMZ system is as follows: first, the NZ-security API calls the TCM API with the access control data created by the AC agent. Then, the TCM API transforms the request data into the encrypted message and sends it to the NZ communication agent, which then transfers the encrypted message to the SZ communication agent through the connected channel between the normal zone and the secure zone. The SZ communication agent calls the secure functions in order to check whether the security application and the user triggering the security application have the access rights for the secure zone. If the security application and the user do not have access rights, the SZ communication agent closes the sessions of the security applications and the user in order to block access to the secure zone. If the security applications and the user do have

access rights, however, the SZ communication agent calls the SZ-security API in order to use the secure functions of the secure zone and returns the result value for the secure functions. The components of the trusted platform in the TMZ system are described as follows:

(i) AC agent
When the security applications access the secure zone, the SZ communication agent of the secure zone needs the access control-related data in order to determine whether or not to block access to the security applications. The AC agent generates the data needed for access control of the security applications, and the NZ-security API transfers the requests for the security applications to the TCM API with them.

(ii) NZ-security API
The NZ-security API is mapped to the TEE functional API of TEE system. The NZ-security API provides the security applications with a set of OS-friendly API in the normal zone. The developers of the security applications in the normal zone can implement TMZ-based secure services with the NZ-security API. The NZ-security API transmits the requests of the security applications to the TCM API with the access control-related data generated by the AC agent. The definition of the TEE functional API is a task of the global platform TEE deliverables roadmap and has not yet been defined. The NZ-security API, as TEE functional API of TEE system, is described in the next chapter of this paper.

(iii) TCM API
The TCM API is mapped to the TEE client API of the TEE system. The TCM API transforms the format of the data transferred from the security applications and the NZ-security API and generates messages in order to transmit them to the secure zone. The TCM API encrypts the messages before sending them to the secure one through the NZ communication agent.

(iv) NZ communication agent
The NZ communication agent is mapped to the REE communication agent of the TEE system. The NZ communication agent requests the creation of the channels and sessions of the security applications to the secure zone in order to communicate with the secure zone, manages the multi-channels, and manages the concurrent connections of multiple security applications.

(v) SZ communication agent
The SZ communication agent is mapped to the TEE communication agent of the TEE system. The SZ communication agent manages the security sessions for multiple security applications, controls access to the applications according to the authentication results of the applications and users, and manages the communication messages in order to transfer the results of the secure functions to the normal zone. It encrypts the communication messages before sending them to the normal zone.

(vi) SZ-security API
The SZ-security API is mapped to the TEE internal API of the TEE system. The SZ-security API provides a set of OS-friendly APIs for the security applications in the secure zone. The developers of the security applications in the secure zone can implement TMZ-based secure services with the SZ-security API. However, the TMZ system does not consider the security applications in the secure zone. When the security applications of the normal zone request secure services through the NZ communication agent, the secure OS components do not call the interface of the security application in the secure zone, but rather the SZ-security API, which provides secure services for the security applications of the normal zone by calling the secure functions in the secure zone.

(vii) Secure functions
The secure functions consist of the access control function, encryption and key management function, and data management function. The security data (e.g., the private key) and security process (e.g., signature generation) are safe because the secure functions are carried out in the secure zone where, unlike the normal zone, there are no threats such as malware, spyware, viruses, and so like. Detailed descriptions of the secure functions are provided as follows:

a) Access control function
This function executes the authentication of applications and users that attempt to access the secure zone and controls their access. When authentication of the applications or users fail or when they do not have access rights for the file data, this function returns the failure value to the normal zone and the SZ communication agent blocks their access to the secure zone. This access control function is executed in the secure zone according to access control policies that are stored in the secure zone. This function calls the data management function in order to gain the access control policies.

b) Encryption and key management function
This encryption and key management function carries out the encryption or decryption of sensitive data transmitted from the normal zone through the white-box cryptography table, generates the signature data using the private key stored in the

secure zone, and so on. It also executes the functions for managing the encryption key and the seed value in order to generate the encryption key. This function supports various encryption algorithms, modes of operation, and the hash algorithms. The sensitive data transferred from the normal zone are stored after encryption in the secure zone. When the security applications in the normal zone request the sensitive data of the secure zone, the data stored in the secure zone are transmitted to the normal zone after decrypting.

c) Data management function
This function manages file data such as the insertion, updating, and deletion of security data transmitted from the normal zone securely. This function calls the encryption and key management function for the encryption/decryption of file data and calls the access control function to check the access rights of applications and users. All data transmitted from the normal zone are encrypted by the white-box cryptography table and stored in the form of files in the secure zone. The seed for generating the encryption key is stored in the file metadata. When the security data are stored in the secure zone, the hash value of each file is generated by the hash functions and the information of the file data, including the hash value, is recorded in the file metadata. In addition, the integrity of the file data is verified whenever the file is read.

d) The data management function
This is called from the normal zone in order to store the certificate file and the private key file securely in the secure zone, and the data management function calls the encryption and key management function in order to encrypt them. The encryption and key management function is called from the normal zone in order to generate the signature data, and the encryption and key management function calls the data management function in order to get the private key. The private key is provided after checking its integrity and decrypting it.

If the private key is managed in the normal zone where numerous threats exist, the possibility of leakage of the private key is very high because the private key file can be copied easily. As such, the trusted platform of the TMZ system provides a separate structure that increases the strength of the security. The private key is stored in the secure zone isolated from the normal zone after encrypting it through the white-box cryptography table stored in the secure zone, and the signature data are also created in the secure zone using the private key that was decrypted through the white-box cryptography table. The private key is safe because the encryption key exists with the algorithm of the white-box cryptography table in the secure zone, and the private key is stored after encrypting it through the white-box cryptography table in the secure zone.

Malicious applications or users may try to access to the secure zone in order to leak the sensitive data stored inside it. A secure channel of communication between an application of the normal zone and the secure zone is established upon the successful completion of the two-step authentication of an application and a user. The trusted platform always checks the access rights of all the applications that call the ND-security API to use the secure zone and can execute user authentication at the time it is required. The trusted platform of the TMZ system can prevent the leakage of sensitive data such as the private key, because it blocks the access of malicious applications or unauthorized users through the two-step authentications of applications and users.

3.2 The definition of NZ-security API as TEE functional API

We define the NZ-security API in the TMZ system as the TEE functional API of the TEE system. The NZ-security API is called for security applications of the normal zone in order to make use of the secure functions of the secure zone. The NZ-security API of the TMZ system is classified into five groups, i.e., session management API, access control API, file management API, data encryption API, and signature management API. The descriptions of the NZ-security API groups are as follows [21]:

(i) Session management API
The channel means a logical connection between a security application of the normal zone and the secure functions of the secure zone and is assigned to all applications of the normal zone that are going to communicate with the secure zone. The security applications of the normal zone request channel creation to the secure zone with the access control data generated by the AC agent through the session management API in order to make use of the secure functions of the secure zone. The session is needed in order to communicate securely between the normal zone and the secure zone. The session is generated after the creation of the channel and the successful authentication of the user who triggers the security application. The security applications of the normal zone call the session management API with the user's input data (e.g., PIN, fingerprint) for user authentication. Upon the completion of the channel and the session between the normal zone and the secure zone, the security

applications can take advantage of other NZ-security APIs.

(ii) Access control API
The access control API is an API group for authenticating the user who triggers the security application when the security application of the normal zone seeks access to the secure zone. This access control API is similar to the session management API in that the security applications call in order to generate a secure session between the normal zone and the secure zone. This access control API is also called with the user's input data for user authentication by the security applications. If the failure result received by the access control API from the secure zone is repeated several times because the user's input data are wrong or another such reason, the SZ communication agent may block the secure session between the normal zone and the secure zone according to the access control policy stored in the secure zone, whereupon the security application of the normal zone will not be able to access the secure zone.

(iii) File management API
The file management API executes the function of managing data or files that are stored in the secure zone. The file management API consists of the file insertion API, file update API, file search API, file size search API, file list search API, and file deletion API among others. When the security applications of the normal zone call the file management API in order to store files in the secure zone, all the files transmitted to the secure zone through the file management API are stored in the secure zone after encryption. And, when the security applications call the file management API in order to search the files in the secure zone, the files stored in the secure zone are transferred to the security application of the normal zone after decryption. Even if the decrypted data are transferred from the secure zone to the normal zone, they are safe because the session between the NZ-security API of the normal zone and the SZ communication agent of the secure zone is secure.

(iv) Data encryption API
The data encryption API has the ability to encrypt or decrypt data transmitted from the security applications of the normal zone, to create and manage the encryption key or the decryption key, to create and exchange public data in order to share the same encryption key, and so on. The data encryption API only performs the function of encrypting or decrypting data and does not perform the function of storing data. The data encryption API is composed of the data encryption API, data decryption API, key exchange API, and so on.

(v) Signature management API
The signature management API group consists of the APIs related to authentication between the security application of the normal zone in the TMZ system and a remote authentication server. It consists of random number generation API, signature creation API, and signature verification API. The private key is required for signature generation in the TMZ system, and the signature is verified by the remote authentication server for the purpose of user authentication. The public key is required for signature verification of the mobile device, and the process is necessary for the remote server's authentication. The public key and the private key, both of which are required for the authentication process, are managed through the data management API.

3.3 Secure user authentication method

The TMZ system separates the normal zone and the secure zone using the hypervisors and builds the trusted platform in mobile devices. The TMZ system, including the trusted platform, provides security applications with a secure user authentication service within a secure execution environment that is isolated from the normal zone, where numerous threats such as malware, spyware, and viruses exist [22, 23]. The TMZ system provides secure certificate-based user authentication that meets level 3 of the e-authentication assurance levels.

Users store certificate files in a location that can be accessed and copied easily. In addition, certificate files carry a high risk of being taken over by a malicious code. Thus, the private key should never be saved anywhere in a plaintext form that is not encrypted [24]. The simplest security method for protecting the private key is to encrypt it with passwords and to save the encrypted result. However, the passwords may often be guessed very easily, while the private key can be decrypted very easily by using the passwords. Thus, users should select their passwords very cautiously. Some attacks are made more difficult when the encrypted key is saved and managed on the disk of a computer that cannot be accessed over a network. The best way to do this is for users to save and manage the encrypted key in a computer that cannot be accessed by others or on a portable media that they can separate and carry around with them after completing the authentication procedure. Users who require very strong security should use tamper-resistant devices in order to manage their private key securely.

Android OS, as the open platform, has been installed in a large number of smart mobile devices. As such, it is

very important that the private key be protected securely in mobile devices based on Android OS. If one suspects that an attacker has stolen one's private key file, then one may also assume that the attacker is able to intercept encrypted messages that arrive in accordance with the corresponding public key [24] and counterfeit the signature data of one's documents while other people recognize continuously the public key as one's public key. In this respect, leakage of the private key may give rise to serious consequences. Thus, we can know that it is vitally important that one's private key be managed securely once again.

Many studies on protection of the private key from unauthorized access have been conducted or are under way. When a certificate and a private key are stored, the secure key store system encrypts the private key using the encryption key dependent on the device and saves it securely [25]. Even in the event that a private key is leaked, the leaked private key cannot be utilized on other devices. There is also a study on the use of a graphical password to enhance password security when protecting the private key derived from the password [26]. There is another proposed scheme in which mobile agents can create non-detachable digital signatures with the forward security of the original signer's private key [27]. It is not necessary for mobile agents to transport the private key in order to create digital signatures, and the private key will not be compromised.

We use the Public Key Cryptography Standards #5 (PKCS#5) [13] to protect the private key. The user inserts a password to decrypt the private key when generating signature data, and as it is necessary to protect the password that a user inserts using a keyboard, the TMZ system uses a secure keyboard.

We have developed the TMZ system in which the trusted platform is built into mobile devices, and the TMZ system performs the secure certificate-based user authentication procedure with the authentication server and the certificate validation server. Using random number challenges and digital signatures eliminates the need to transmit passwords for authentication, which in turn reduces the threat of their being compromised. Such a compromise would allow an attacker to use the same information to authenticate himself repeatedly. The procedure for certificate-based user and server authentication is as shown in Fig. 4. In Fig. 4, we can see the secure user authentication procedure using the TMZ system.

At first, the TMZ system requests user authentication to the authentication server (1), whereupon the authentication server creates a random number challenge with a value of R_S (2), and transmits it to the TMZ system (3). The value of R_S is retained by the TMZ system.

The utilization of random number challenges in the certificate-based user authentication procedure can also block an attacker attempting to intercept the authentication tokens, including the signature data created by other users, and prevent the attacker from using them successfully later on [28]. However, the random number challenge should be newly created for

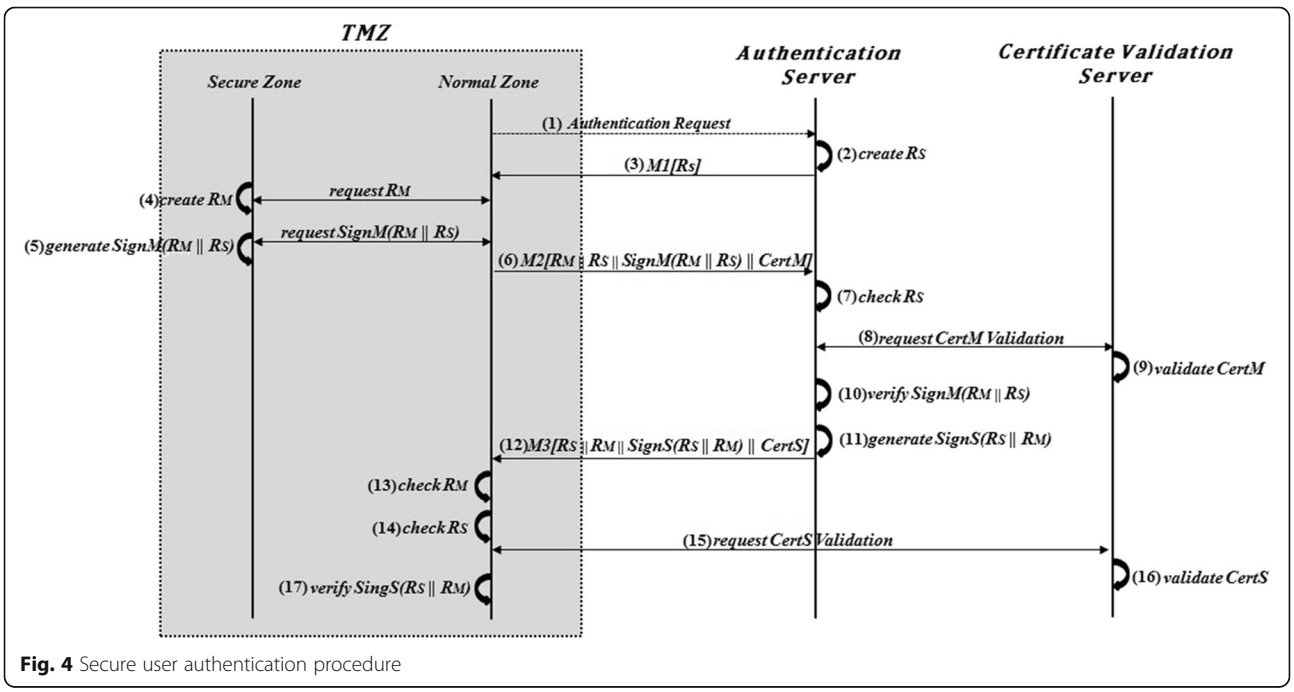

Fig. 4 Secure user authentication procedure

each authentication token exchange procedure. A security method for blocking replay attacks also depends on the random number challenges that were created repeatedly in the low probability. The TMZ system uses the FIPS approved by the random number generator.

The TMZ system retains the value of R_S, creates a random number challenge with a value of R_M (4), generates $SignM(R_M \,||\, R_S)$ as the signature data of R_M and R_S (5) by using the private key that is saved in the secure zone securely, concatenates the data, attaches the *CertM* as the certificate of the TMZ system, and transmits the message *M2* to the authentication server (6). It is the most important to keep the user's private key securely. If the user fails to maintain the private key securely, it may result in an attacker masquerading as the user by using the user's private key to sign the authentication token. The private key is safe because it is encrypted and stored in the secure zone, the integrity of the private key is checked after decrypting, and the $SignM(R_M \,||\, R_S)$ is generated in the secure zone. By utilizing the private key in order to create the signature data for the authentication token, it becomes almost impossible for an attacker to masquerade as another user. Nevertheless, the TMZ system still relies on passwords to decrypt and access the private key [13]. Therefore, the password for decrypting the private key based on PKCS#5 must be kept secure. The TMZ system protects the password for decrypting the private key through the secure keyboard and APIs that prevent screen capture. It can also be protected more securely through the TMZ system into which the type 1 hypervisor is built. Indeed, a TMZ system into which the type1 hypervisor is built can provide the user with more secure services by blocking the leakage of the private key than a system into which the type 2 hypervisor is built.

Pseudo code-1 is a pseudo code that performed on the TMZ system in the user authentication procedure of Fig. 4 (3) to (6). Pseudo code-1 is as follows:

The authentication server checks whether the value of R_S maintained in step (2) is identical to the value of R_S included in the message *M2* transmitted from the TMZ system (7) and also requests the validation of *CertM* as the TMZ system's certificate to the certificate validation server (8). The certification validation server carries out checks to determine whether the CRL includes the *CertM*, whether the validity date of *CertM* has expired or not, whether the "KeyUsage" field of *CertM* includes the "digitalSignature" value or not, and whether the "Issuer" field of *CertM* has the DN of the upper authority for the validation of the certificate or the certificate chain (9) [29].

The authentication server verifies the signature data $SignM(R_M \,||\, R_S)$ received from the TMZ system after the certificate validation server has validated the certificate (10), generates $SignS(R_S \,||\, R_M)$ as the signature data of R_M and R_S (11), concatenates the data, attaches the *CertS* as the certificate of the authentication server, and transmits the message *M3* to the TMZ system (12).

The TMZ system checks whether the value of R_M maintained in step (4) is identical to the value of R_M included in the message *M3* (13) and whether the value of the R_S searched from the message *M1* is identical to the value of the R_S included in the message *M3* (14). Then, it requests the validation of *CertS* as the authentication server's certificate to the certificate validation server (15). The certification validation server performs the validation of the certificate or the certificate chain (16). The validation of the certificate or the certificate chain includes verifying the signature data of each certificate in the certificate chain and validating whether each certificate in the certificate chain has expired or been revoked by their issuer. Finally, the TMZ system verifies the signature data $SignS(R_S \,||\, R_M)$ received from the authentication server (17).

Pseudo code-2 is a pseudo code that performed on the TMZ system in the user authentication procedure of Fig. 4 (12) to (17). Pseudo code-2 is as follows:

```
Pseudo code-1.
  begin
    if receive M1 message from Authentication Server and retains RS of M1 message
      begin
        retrieve and retain the random data RS from M1 message
        request create_random(RM, rsize) to Secure Zone
        retain the random data RM
        request generate_sign(RM || RS, rsize, SignM, ssize) to Secure Zone
        request get_certificate(CertM) to Secure Zone
        generate and send M2[RM || RS || SignM(RM || RS) || CertM] message
      end
  end
```

```
Pseudo code-2.
  begin
      if receive M3 message from Authentication Server

      begin
        if R_M of M3 message == R_M generated in step (4) then
            if Rs of M3 message == Rs received from Authentication Server in step (3) then
                retrieve CertS from M3 message
                request validate_certificates(CertS) to Certificate Validation Server
        else
            the authentication procedure is stopped and the authentication result is failure
      end

      else
      the authentication procedure is stopped and the authentication result is failure

      retrieve Rs, R_M, SignS, CertS from M3 message
      call verify_sign(Rs || R_M, rsize, SignS, ssize) with CertS in Normal Zone
  end
```

Figure 5 shows the screenshots of the user authentication request and the authentication process result in TMZ system. When the user requests the authentication process through the security application, the TMZ system prompts you for your password. The user password is needed to decrypt the private key used to generate the signature data by using the random data received from the authentication server and the random data generated from the secure domain. When the user enters a password, the TMZ system generates the signature data and sends the random data and the signed random data to the authentication server. Then, the authentication server verifies the signed random data with the random data and transmits the result of the authentication process to the TMZ system.

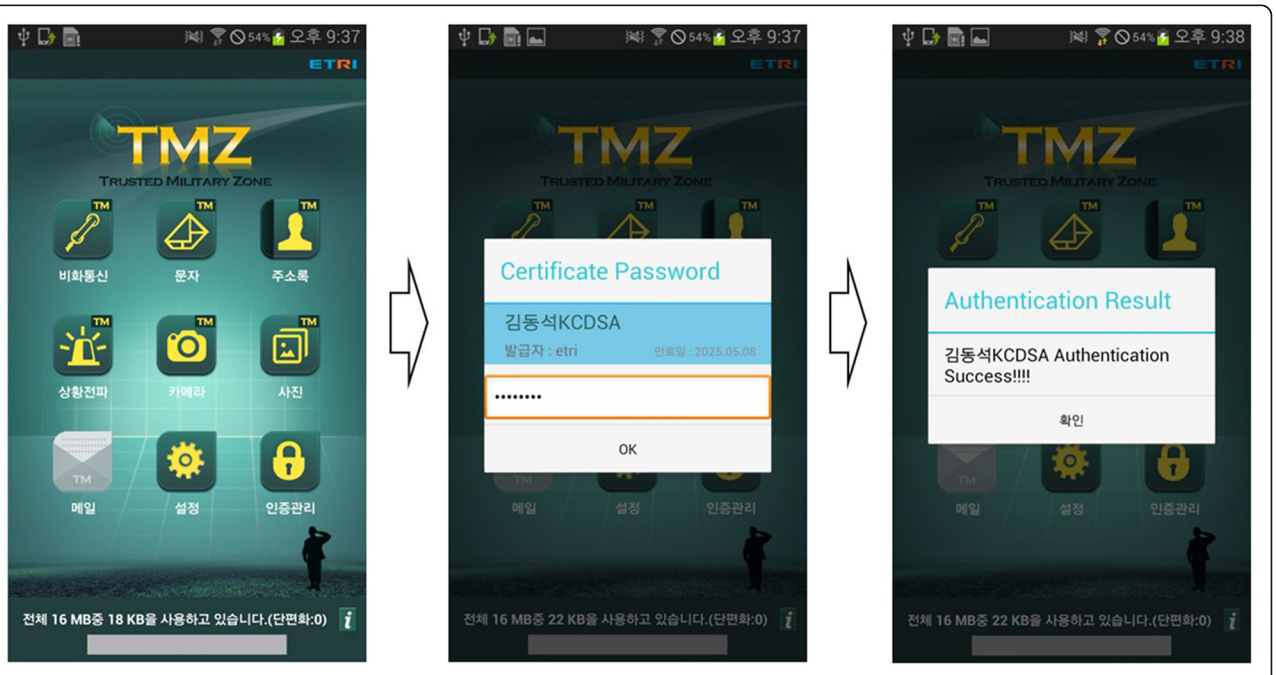

Fig. 5 Screenshots of the user authentication in TMZ system

4 Results of the experiment

We implemented the TMZ system perform to store the private key file and to generate signature data securely in the secure zone in order to prevent the illegal leakage of the private key file. For this chapter, we measured the performance for the KCDSA-based signature generation in the TMZ system. We performed experiments on the mobile device according to the specifications shown in Table 1. We built the TMZ system, with VIMO included, like ETRI's type 2 hypervisor built into the Galaxy S3 and installed the μC/OS-II as a secure OS in the secure zone.

Our experiments to measure the performance of signature generation were carried out in two ways, one in the secure zone and the other in the normal zone based on inter-domain communication (NZ communication agent and SZ communication agent). The security applications of the normal zone requested the signature generation to the secure zone through inter-domain communication and received the return value and the signature data created from the secure zone through inter-domain communication. Therefore, the performance measure experiment of signature generation in the normal zone based on inter-domain communication is required from the user's point of view.

The first way concerns performance obtained only by generating the signature data in the secure zone, whereas the second way concerns the performance of the overall system through inter-domain communication from the secure zone to the normal zone. In the second way, we measured inter-domain communication through the normal channel and the encrypted channel, respectively. We measured the performance of each way a total of 20 times and calculated the average values for each. The results of these performance measurements of signature generation based on the normal channel and encrypted channel, and in the secure zone, are all shown in Fig. 6. The performance times were measured in microseconds (μsec).

The average performance time value of signature generation through the normal channel is 1268 (μsec), the average performance time value of signature generation through the encrypted channel is 1425 (μsec), and in the case of signature generation in the secure zone, it is 1245 (μsec). The difference in the value of signature generation by the normal channel and signature generation in the secure zone is about 23 (μsec). It can be the overhead of inter-domain communication or the hypervisor; there is little overhead of inter-domain communication or hypervisor. Meanwhile, the difference in the value of signature generation by the normal channel and signature generation by the encrypted channel is about 157 (μsec), which can be said as the encryption overhead of the inter-domain communication channel, and it is large. Thus, it is considered that there is a need to improve the performance of inter-domain communication through the encrypted channel.

Table 1 Specifications of the experimental device

CPU	Exynos4412 Cortex-A9 Quad-core 1.4GHz
RAM	2GB DDR2 SDRAM
Kernel version	3.0.31
Android version	Android 4.1.2
Secure OS	μC/OS-II
Hypervisor	VIMO 1.0

5 Usage cases

Figure 7 briefly depicts the EAP-TLS authentication procedure using the TMZ system. The EAP-TLS authentication procedure [30] is as follows. The TMZ system initiates a TLS connection procedure by sending its identity (EAP-response/identity) to the EAP-TLS server. Then, the mutual authentication procedure of the TLS connection procedure is executed by using the certificates of TMZ system and EAP-TLS server. The TMZ system and EAP-TLS server might share the TLS master secret through the premaster secret value that the TMZ system generates randomly. The TMZ system and EAP-TLS server can generate the MSK (master session key) through TLS master secret and the EAP-TLS server delivers the MSK to the AP. The TMZ system and the AP generate an encryption key (PTK/GTK) that is used in the wireless section from the same MSK and initiates a secure data communication after four-way handshaking process.

The EAP-TLS protocol provides the mutual authentication through the certificates of the mobile device and the EAP-TLS Server, and the certificates are important for the mutual authentication. When the certificate of the mobile device is forged or deleted, the mobile device authentication cannot be performed correctly. The TMZ system can be used for the secure management of the certificate of the mobile device.

6 Conclusions

As the utilization of smartphone devices has increased greatly, individual smartphones are being widely utilized for business purposes and e-commerce nowadays. As such, a method of authenticating users securely with the remote server is needed. As there is a user authentication solution based on digital certificates, in this study, we propose a secure user authentication framework based on digital certificates.

First, we described the architecture of the TEE-based TMZ system and the components of the trusted platform—such as the AC agent, NZ-security API, TCM API, NZ communication agent, SZ communication agent, SZ-security API, and secure functions. We also

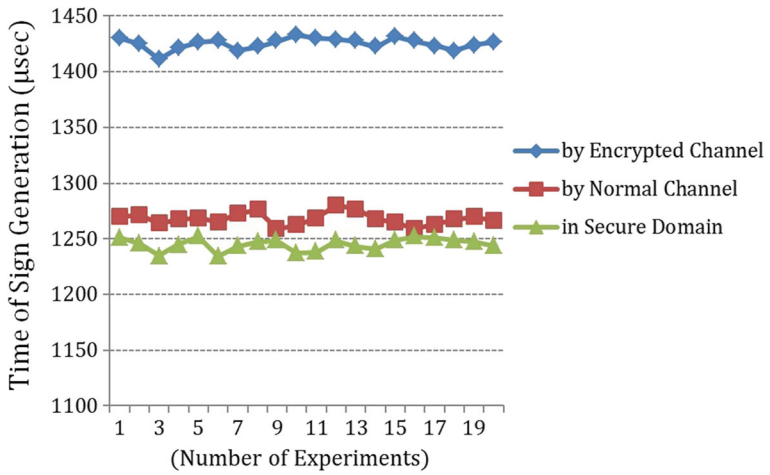

Fig. 6 Results of the performance measurement of signature generation

defined the NZ-security API as the TEE functional API of the TEE system, which is a part of the global platform TEE deliverables roadmap.

The TMZ system constructs the normal zone and the secure zone separately using virtualization technologies and builds the trusted platform into smartphone devices. The TMZ system, including the trusted platform, provides a separate and secure execution environment that is isolated from the normal zone, where there are many threats to security applications. The TMZ system

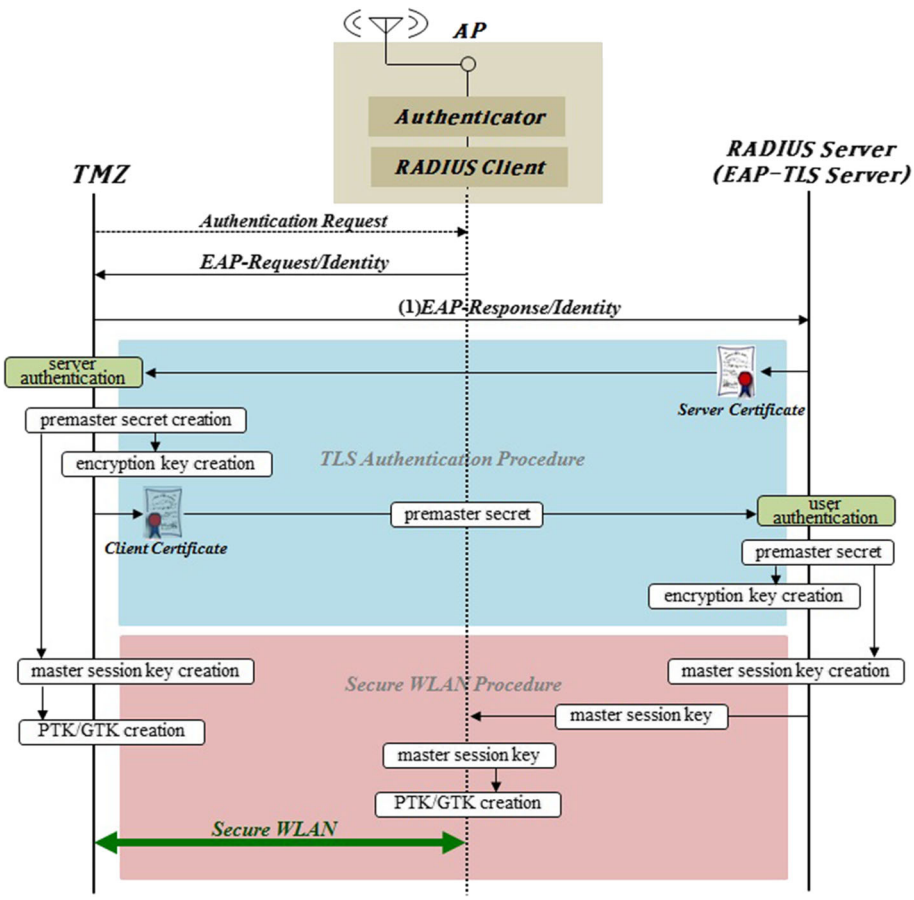

Fig. 7 EAP-TLS authentication procedure

provides certificate-based user authentication securely and meets level 3 of the e-authentication assurance levels. The certificate is a multi-factor software cryptographic token while certificate-based user authentication is a multi-factorial authentication. But the digital certificate is a software-type token that is stored in a logical form and thus has the drawback of carrying a high risk of leakage. The TMZ system prevents the leakage of the digital certificate through the trusted platform, and it can also be protected more securely in a TMZ system into which the type 1 hypervisor is built.

We described the secure certificate-based user authentication procedure. The TMZ system performs secure certificate-based user authentication based on the secure zone with the authentication server and the certificate validation server.

The use of random number challenges in the certificate-based user authentication procedure can also block an attacker's attempts to intercept the authentication tokens, including the signature data created by other users, and prevent the attacker from reusing them successfully later on. The TMZ system includes the secure zone, which is isolated from the normal zone in which the open OS is installed, and the trusted platform, which is built into smartphone devices. The TMZ system based on the trusted platform prevents the leakage of the private key because the digital certificate and the private key pair are stored and managed securely in the secure zone, and signature generation using the private key is executed securely in the secure zone. Meanwhile, the password for decrypting the private key is protected through the secure keyboard and APIs that prevent screen capture.

In the future, we will implement the TMZ system in smartphone devices for which the kernel version of Android OS is 6.0, and the VIMO hypervisor will be updated according to upgrades of Android OS. We are currently working on improving the performance of encrypted inter-domain communication between the NZ communication agent and the SZ communication agent. Finally, we plan to implement a key management mechanism that creates and exchanges the new encryption keys periodically in the future.

Acknowledgements

This work was supported by a grant awarded by the Institute for Information & Communications Technology Promotion (IITP) and funded by the Korean government (MSIP) [R0101-16-0195, Development of an EAL 4 level military fusion security solution for protecting against unauthorized access and ensuring a trusted execution environment in mobile devices].

Received: 1 April 2016 Accepted: 15 September 2016
Published online: 29 September 2016

References

1. L Sui, Strategy Analytics: Global Smartphone Shipments Hit a Record 1.4 Billion Units in 2015. (Strategy Analytics Web, 2016), https://www.strategyanalytics.com/strategy-analytics/news/strategy-analytics-pressreleases/strategy-analytics-press-release/2016/01/27/strategy-analytics-global-smartphone-shipments-hit-arecord-1.4-billion-units-in-2015#.V8UY9fmLRhF. Accessed Feb 2016.
2. A Scarsella, W Stofega, Worldwide Smartphone Forecast Update 2015–2019: December 2015. (IDC Research Web, 2015), https://www.idc.com/getdoc.jsp?containerId=US40734415. Accessed Feb 2016.
3. S Shin, T Kwon, A Survey of Public Provable Data Possession Schemes with Batch Verification in Cloud Storage. Journal of Internet Services and Information Security. **5**(3), 37–47 (2015)
4. F Gadaleta, R Strackx, N Nikiforakis, F Piessens, W Joosen, On the effectiveness of virtualization-based security, in Proceedings of IT Security, (Freiburg, Germany, 2012)
5. D Oh, I Kim, K Kim, S-M Lee, WW Ro, Highly Secure Mobile Devices Assisted with Trusted Cloud Computing Environments. ETRI Journal **37**(2), 348–358 (2015)
6. D Huang, Z Zhou, L Xu, T Xing, Y Zhong, Secure Data Processing Framework for Mobile Cloud Computing, Paper presented at IEEE INFOCOM 2011 Workshop on Cloud Computing (Shanghai, China, 2011), pp. 614–618
7. HT Dinh, C Lee, D Niyato, P Wang, A survey of mobile cloud computing: architecture, applications, and approaches. Wireless Communications and Mobile Computing **13**(18), 1587–1611 (2013)
8. NIST, Special Publication 800-118(Draft), K Scarfone, M Souppaya, Guide to Enterprise Password Management. (2009)
9. ARM, White Paper, R Coombs, Securing the Future of Authentication with ARM TrustZone-based Trusted Execution Environment and Fast Identity Online (FIDO). (2015)
10. IETF, RFC 2865, C. Rigney, S. Willens, A. Rubens, W. Simpson, Remote Authentication Dial In User Service (RADIUS). (2000)
11. IETF, RFC 5246, T. Dierks, E. Rescorla, The Transport Layer Security (TLS) Protocol Version 1.2. (2008)
12. NIST, Special Publication 800-63-2, WE Burr, DF Dodson, EM Newton, RA Perlner, WT Polk, S Gupta, EA Nabbus, Electronic Authentication Guideline. (2013)
13. IETF, RFC 2898, B. Kaliski, RSA Laboratories, PKCS#5: Password-Based Cryptography Specification Version 2.0. (2000)
14. TrustZone. (ARM Web, 2016), http://www.arm.com/products/processors/technologies/trustzone/index.php. Accessed Feb 2016
15. ARM, White Paper, TrustZone API Specification Version 3.0. (2009)
16. Device Specifications. (Global Platform Web, 2016), http://www.globalplatform.org/specificationsdevice.asp. Accessed Feb 2016
17. GlobalPlatform, WhitePaper, TEE Client API Specification v1.0. (2010)
18. GlobalPlatform, WhitePaper, TEE Internal API Specification v1.0. (2011)
19. GlobalPlatform, WhitePaper, TEE System Architecture v1.0. (2011)
20. D Jaramillo, B Furht, A Agarwal, *Virtualization Techniques for Mobile Systems* (Springer Link, Switzerland, 2014), pp. 5–20
21. G Kim, Y Jeon, J Kim, The Secure Urgent Situation Propagation System Using the Mobile Security Solution, in Proceedings of The Second International Conference on Computer Science, Computer Engineering, and Social Media (Lodz, Poland, 2015), pp.109-114
22. A Skovoroda, D Gamayunov, Securing mobile devices: malware mitigation methods, Journal of Wireless Mobile Networks, Ubiquitous Computing, and Dependable Applications. 6(2) 78–97 (2015)
23. B Rashidi, C Fung, A Survey of Android Security Threats and Defenses, Journal of Wireless Mobile Networks, Ubiquitous Computing, and Dependable Applications. 6(3) 3–35 (2015)
24. RSA Laboratories: HOW SHOULD I STORE MY PRIVATE KEY? (EMC Corporation Web, 2016) http://www.emc2.nu/emc-plus/rsa-labs/standards-initiatives/store-private-key.htm. Accessed Feb 2016
25. Y Park, S Kim, D Lee, The Secure Key Store to Prevent Leakage Accidents of the Private Key and Certificate, Journal of The Korea Institute of Information Security & Cryptology. (2014). doi:10.13089/JKIISC.2014.24.1.31
26. B-H Kang, B-S Kim, K-K Kim, Securing the Private Key in the Digital Certificate Using a Graphic Password. The Journal of Society for e-Business Studies **16**(4), 1–16 (2011)
27. S Yang, Q Zhao, Q Liu, Secure Mobile Agents in eCommerce with Forward-Secure Undetachable Digital Signatures. ETRI Journal **37**(3), 573–583 (2015)
28. NIST, FIPS 196, Entity Authentication Using Public Key Cryptography. (1997)
29. ITU-T, Recommendation X.509, Information technology - Open Systems Interconnection - The Directory: Public-Key and Attribute Certificate Frameworks. (2012)
30. IETF, RFC 5216, D Simon, B Aboba, R Hurst, The EAP-TLS Authentication Protocol. (2008)