Trusted Access Control System for Smart Campus

Mohamed Amine Bouazzouni*, Emmanuel Conchon†, Fabrice Peyrard*, Pierre-François Bonnefoi†
*University of Toulouse; INP; IRIT 2 rue Charles Camichel, Toulouse, France.
Email: {mohamedamine.bouazzouni, fabrice.peyrard}@enseeiht.fr
†University of Limoges, XLIM, UMR CNRS 6172, 123 avenue Albert Thomas, 87060 Limoges, France.
Email: {emmanuel.conchon, bonnefoi}@unilim.fr

***Abstract*—Many access control systems are still based on the first generation of contactless technologies like RFID or NFC despite well known cloning weakness. Furthermore, the cost of the deployment of secure cards for large organizations (DESFIRE for instance) is expensive. Also, These systems do not always check authentication of the holders of RFID tags or NFC cards. In this paper, we will present a proposal of an architecture to build a secure access control system based on Trusted Execution Environments (TEE) and Identity Based Encryption (IBE) mechanisms. We also identify the challenges to overcome before deploying such an architecture.**

***Index Terms*—OP-TEE, Trusted Execution Environments, Identity Based Encryption, RSA, AES, secure access control.**

1. Introduction and context

The need of security and privacy led many organizations to build control access systems for critical infrastructures such as offices and residences. In University Campus for instance several buildings have restricted access and can be accessed either by students, academics or administrative staff. Most of these systems are based on RFID (Radio-Frequency IDentification) or low-cost NFC (Near Field Communication) tags. However, these technologies are vulnerable to some attacks [1] [2] allowing the cloning of the tag/card to gain access to the facility.

The weaknesses on these technologies are mainly due to the limited storage and computational power. This limitation does not allow the developer to use complex cryptographic algorithms that could allow to address the cloning issue. Furthermore, no user or platform authentication is performed. Then, any user even non-authorized can access the facility if he has a stolen access card. Even using a secure card like DESFIRE for instance, induces two other drawbacks: Deployment cost and no user and/or platform authentication.

To address these issues, new access control systems have emerged relying on a smartphone which replaces the tag/card. These new systems are based on the concept of a virtual card where a smartphone can be used as a substitute to a regular smart card. Such operation is called dematerialization[1]. These new card-less systems enable the use of complex cryptographic algorithms to perform a user authentication. These algorithms mainly rely on PKI [3] (Public Key Infrastructure) where a couple of encryption keys is provided to every user. This couple is composed of a private key that remains secret and is used to decrypt an encrypted message and of a public key that can be used by anyone to encrypt message for the owner of the public key. A major drawback of PKI is the ability to provide the public key to every user in a secure way. A commonly used solution is based on digital certificates that are emitted by trusted certification authority to authenticate the public key owner. But, the diffusion of these certificates and their revocation when the certificate is expired for instance are still problematic.

To avoid the use of certificates, a solution is Identity Based Encryption (IBE) that has been introduced by Shamir in 1984 [4]. To sum up, a user public key can be replaced by the identity of the user (e.g. his email) avoiding the use of digital certificates. IBE mechanisms can thus be deployed easily without the need to any Public Key Infrastructure (PKI). A IBE mechanism is relying on a Private Key Generator (PKG) which uses a master secret to derive the public key based on the identity. In [5], an access control system based on IBE has been proposed. However, no secure environment was used to secure the PKG so that it can potentially be compromised. Indeed, the storage of the master secret as well as enforcing the confidentiality and the integrity of the generation process are the main challenges to deal with.

The storage of encryption keys as well as the secure execution of crytographic operations is a major issue especially on a mobile device. In the recent years, several trusted mobile computing solutions have been proposed among which are the Trusted Execution Environment (TEE). TEE is a combination of a hardware part (processor) and a software part (Secure Operating System) that allows the secure storage and processing of the data on a smartphone. A main drawback of TEE is the use of an hardware component that is fully controlled by the manufacturer. Indeed, it does not allow the user to use this component in an easy way and often require a prior manufacturer agreement before deploying a new secure application on it. But, new fully software based solutions such as OP-TEE can be used to bypass this agreement. This kind of solution can then be considered as a good candidate for the storage of the PKG

1. In the remaining of the paper, the word dematerialization will mean replacing a physical card with a virtual one saved on a smartphone

DOI 10.1109/UIC-ATC-ScalCom-CBDCom-IoP-SmartWorld.2016.95

in an IBE-based solution for trusted access control.

In this paper we propose a new access control architecture based on TEE and IBE for smart Campus. This architecture is a part of the neOCampus operation [6] that aims to provide new software and hardware components for the university campus of tomorrow. The reminder of this paper is organized as follow. First, we give an overview of the neOCampus project and of our proposed architecture. Then, the Identity Based Encryption mechanism that we propose to use is presentedIn section 4 we present Trusted Execution Environments detailing how they work and how they can be used to secure the PKG. The general architecture is presented in section 6 with a discussion of open research opportunities.

2. The neOCampus operation

The objective of the neOCampus operation is to design new products and services related to ambient systems. It consists of many software and hardware interconnected devices for the digital campus of tomorrow that are sustainable and intelligent and that combine innovative educational materials. To achieve this, neOCampus deploys sensors, storage facilities, tracking devices, simulation and innovative materials in existing university buildings and around the campus to increase users' quality of life and reduce the overall energy consumption.

So far, students and university staff have a card based on the Mifare classic 1K NFC cards. These cards are used to access to restricted areas of the university such as buildings. Since these cards have vulnerabilities allowing an attacker to clone it, the university has switched these cards for DESFIRE NFC cards. These cards are more secure than the previous one and can perform simple cryptographic operations. However, there is still two major drawbacks: Deployment costs and user authentication. Indeed, when a Mifare 1K costs in average 0.02$, a DESFIRE card is three times more expensive. Also, these cards are not able to perform a user authentication. For instance, if a malicious person steals a card from a regular user, he will be able to access these facilities without a problem. Therefore, the overall security is only partially enforced by these new cards.

For these reasons, as a part of the neOCampus operation, we propose to implement a new access control mechanism based on the dematerialization of the card into a smartphone and on the use of a secure cloud server. This server will be used to perform a secure mutual authentication between readers and users in order to increase the security level of the overall system. To achieve this mutual authentication, we propose to rely on Identity Based Encryption that avoid the exchange of public keys between users. It will be coupled with the use of Trusted Execution Environments to enforce the overall system security.

3. Identity Based Encryption

The Identity Based Encryption scheme was first designed to address some issues induced by the usage of the asymmetric encryption related to the security of the exchange of the public key. The common response to this problem is to use a Public Key Infrastructure (PKI) that issues a certificate to derive trust from a unique source, known as the Certificate Authority (CA), in order to securely link an entity's identity to a particular public key. More precisely, a signature performed by the CA ensures the trusted bind between an identity and a public key. Deploying and using a PKI is very challenging as the user needs to enroll into the PKI and provide some proof of its identity in order to obtain a certificate. A strong level of security is achieved as users could trust each other through the trusted third-party, but some issues arise with the need to verify if a certificate has been revoked, especially in a context of an embedded environment with limited network connectivity or where the network hardware interface could not be fully trusted.

In order to avoid some issues of certificate-based solution, the IBE mechanism uses directly the identity of a user (such as his email for instance) or a device for authentication, encryption and decryption.

This approach is a trade-off between a straightforward process of binding an identity to an asymmetric pair of keys and the usage of a new kind of trusted third-party acting as a mandatory key escrow (it is capable of producing all the asymmetrical keys used). Moreover, adoption of IBE has been hampered by the need to adapt common security protocol and hardware to a new cryptosystem and to provide the security proofs for all the required cryptographic operations (uniqueness of one key, signature, encryption and decryption). In the remaining of the section, we will expose how to build an IBE system and the two main IBE schemes: online and offline.

3.1. IBE components

Every IBE system is based on four basic algorithms as defined by Shamir [4].

- **Setup**: This algorithm generates the global system parameters and the master secret key.
- **Key extraction**: The master secret key is used by a Private Key Generator (PKG) to extract every user's private key. In this algorithm lies the key escrow problem. Indeed, as the owner of the master secret key, the PKG is then able to generate every private key to have access to every encrypted messages.
- **Encryption**: The encryption of a message is made thanks to the public ID of the receiver.
- **Decryption**: The decryption is made with the corresponding private key provided by the PKG.

The first IBE scheme fully satisfactory in terms of security and performances has been proposed by Boneh and Franklin [7] in 2001. It is based on pairing on elliptic curves

with performances similar to ElGammal's algorithm and a security proof in the random oracle model. To achieve this, it proposes an instantiation of Shamir's encryption algorithm that is performed thanks to a master public key and a hash function applied to the identity. Waters in [8] has proposed a transposition of Boneh and Franklin's model in the standard model that still rely on pairing but without random oracle. However as these two schemes rely on pairing on elliptic curves practical implementations has not been available for a long time. Therefore, more practical IBE schemes based on existing cryptographic standards has emerged such as Callas [9].

3.2. IBE scheme from Callas

Callas [9] proposed an IBE scheme that can be integrated to the common RSA cryptosystem providing the same cryptographic capabilities. The advantage of this solution is its ability to use off the shelve components compliant with the RSA cryptosystem. This IBE scheme is intended to be as secure as RSA is and sustain future security challenges such as attacks on hash functions [10]. The IBE components they used are described bellow. Moreover, it relies only on well-known cryptographic operation that are widely spread in programming languages libraries as well as tools.

3.2.1. System Setup. The algorithm below describes the System Setup of the IBE solution proposed by Callas. First, an Identity token named *IDT* is generated by applying and Identity Digest Function (IDF) on the PKG master secret K_{pkg} and on the identity of the user as presented as in Equation 1.

$$IDT = IDF(K_{pkg}, Identity) \tag{1}$$

The IDF function can be an HMAC or a CBC-MAC or any other pseudo-random function. Another solution is to use an asymmetric crypto-system such as RSA where the K_{pkg} could be an RSA key and the IDT could be the result of the RSA encryption of the Identity by the master key. Since the IBE makes an unique association between an identity and a key, we have to pay attention to the padding which can disrupt the system determinism. Note that the selection of the IDF function is critical and that the security of the underlying system is relying on it to provide an acceptable security level for the IDT.

The second step is to seed a Random Number Generator (RNG) function with the IDT. This RNG function has to be different of the IDF to increase the overall system complexity. The result of this operation will then be used in the key extraction process.

3.2.2. Key Extraction. The algorithm presented in Figure 3.2.2 shows the key extraction process for this IBE scheme. It can be noted that the user has to authenticate to the PKG to receive his private key. This critical communication can be achieved through a regular SSL/TLS communication. The public key can be provided by to any

```
IDT = IDF(K_pkg, Identity);
RNG_seed(IDT);
Key_Pair = KeyGenerator(RNG);
if(!authenticated(user)) then
        return Pub_Key;
else
        return Key_Pair;
```

Figure 1. Key extraction algorithm

user authenticated or not. At this point the two users are able to communicate and to encrypt/decrypt messages.

This Key extraction step is the main difference between Callas and the previous IBE schemes as it induces a communication between the PKG and the user to find a public key. This is caused by the use of RSA where it is not possible to provide a way to generate a public key based only a master public key and the receiver identity. Therefore this operation has to be performed by the PKG. Nonetheless, due to its simplicity in terms of implementation we choose to rely on Callas Scheme as a starter for the proof of concept.

After exposing IBE, we notice that all the security of such a mechanism is dependent on the secure storage of the master secret and the secure processing of the key generation operation. In order to achieve this, the use of a secure environment is mandatory. In the next section, we will present the secure environment chosen to implement our IBE-based solution: The Trusted Execution Environments (TEE).

4. Trusted Execution Environment overview

4.1. General architecture

TEEs [11] [12] [13] are a combination of a hardware and a software parts. Moreover, the system is divided into two execution environments.

- The first environment is the Rich Execution Environment (REE) also called Normal World Execution Environment. This environment represents the standard OS of the smartphone such as Android for instance. The term Rich describes the extensive features of the OS such as camera management, telephony capabilities and so on. These features significantly increase the attack surface.
- The second environment is the Trusted Execution Environment. It represents the Secure OS responsible for performing sensitive processing such as cryptographic operations. It also has the capability to secure the display and the input by using a secure communication channel to connect the processor to the I/O peripherals. OP-TEE and OPEN-TEE presented in section 4.2 are examples of Secure OS.

One important feature of the TEE is the secure storage where direct user access from the REE is forbidden. This

storage provides security to keys used for cryptographic operations that are performed into the REE.

A secure boot process is also needed. It enables Rich OS and Secure OS integrity checking. The secure process follows the steps bellow:

1) Read a trusted ROM (locked at manufacturing),
2) Check Signature and integrity of the Secure OS,
3) Set up the Secure OS,
4) Transfer the control to the Secure OS.

In the TEE world, two heterogeneous systems coexist and therefore it is necessary to set isolation rules between them to avoid data leak. Thus, we need a third mode called Monitor mode. This mode allows context saving and switching between the Rich OS and the Secure OS. This mode is triggered to perform security operations into the TEE.

The Secure OS is a limited instruction set OS. This constraint is necessary to reduce the attack surface. It schedules sensitive applications running on it. These applications are called trustlets. Trustlets executes secure instructions like cryptographic operation : key generation, data encryption and decryption. The Secure OS manages the resources between all the trustlets. A trustlet is a secure application which runs on the TEE. It must be signed by the chip manufacturer and signature verification is performed before loading on the TEE. Some Application Programming Interfaces (APIs) can be added to manage extra features but they have to be checked to prevent eventual security breaches.

So far, the majority of Secure OS used for TEE are proprietary which makes it difficult to get access to the TEE to test a developer trustlet. In fact, if a third party wants to get access to a TEE to test its application, this application has to be signed with the private key of the manufacturer. The manufacturers are reluctant to sign third party applications. This restriction is motivated by the wish to get a high security level by having only one source of application provisioner. This is the biggest drawback for TEE expansions. Developers continue to advocate for change access rights to TEE to deploy their own applications.

4.2. Secure OS overview

At the time of writing, several implementations of Secure OS for TEE are available. The most notable ones are: Sierra TEE, Genode, Trusted Language Runtime, OP-TEE and TrustKernel T6.

- **Sierra TEE**: According to the OS maker [14], the Sierra TEE OS performs an integrity management process and several scanner checks. Among these scanners, Sierra TEE checks the integrity of the Android file system, the Android OS, processes running and the interrupt table. These checks are performed to ensure the security of the Android execution. It also allows to perform key management, device management, Data Right Management (DRM) and I/O operations securisation. Furthermore, for secure input/output, the peripherals communicate directly with the TEE. In this case, the Android OS cannot intercept passwords and sensitive information from I/O operations.
- **Genode**: Genode is a secure OS with a very low complexity [15]. Its source code is approximately 10,000 lines. This feature is very important as it allows a simple security verification of the OS. Moreover, Genode manages the execution of touchscreen driver and hardware as well as the frame buffer driver. These executions are run in the secure world ensuring secure interactions between the TEE and the user. Genode also controls the Graphical User Interface (GUI) textboxes used to collect user passwords and screen used to display secure information from the TEE. This OS ensures that no information leakage occurs.
- **Trusted Language Runtime (TLR)**: This OS [16], based on a *.NET platform*, relies on a multiple trustboxes[2] system. Each trustbox is like a container isolated from the others. The device manufacturer initializes a pair of key (public/private) in a trustbox to allow remote verification and attestation. For now, TLR is not able to communicate securely with users because the I/O operations are managed by the REE. This allows attackers to intercept, modify and alter the data flowing between the users and the TEE.
- **TrustKernel T6**: T6 is a secure OS for ARM processors with TrustZone capability. It can run simultaneously the secure OS with one of the multiple Rich OS supported (Android, Linux, etc.) [17]. It provides strong security properties and enhances the ease use. The TrustKernel team provides all the source code and the support. T6 is provided with multiple libraries like LibC and OpenSSL to facilitate the user application development. Finally, T6 is fully compatible with the GlobalPlatform specifications [11].
- **OP-TEE project**: OP-TEE [18] is a secure OS for TEE developed by STMicroelectronics in collaboration with LINARO [19] that is fully compatible with the GlobalPlatform specifications [11]. OP-TEE consists of a client API, a Linux kernel driver and a Secure OS. As mentioned previously in this section, the switching between the Rich OS and the Secure OS is managed by a monitor mode. OP-TEE has also a multi-core capability. The unique constraint is that only one core can be in the secure world at the same time.

Regards to the difficult deployment of a solution on a real device due to the manufacturer restrictions, the use of an open environment seems to be a good alternative. In our work, we choose to use OP-TEE to implement and deploy our secure control access system based on the Identity Based Encryption mechanism.

2. A trustbox is a runtime environment which protects the confidentiality and the integrity of code and datas

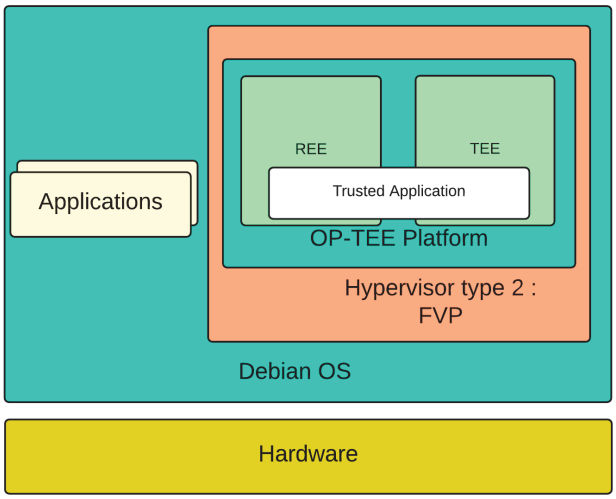

Figure 2. OPTEE Architecture

5. The open source TEE implementation

OP-TEE [18] is a Linaro [19] and STMicroelectronics project with the objective to release a totally virtualized and open source TEE. Indeed, in OP-TEE the platfom is either virtualized by using Qemu[3] [20] or Fixed Virtual Platform (FVP) [21] or consists of a physical development card of ARM called JUNO[22]. For the virtualized platform, a Debian based OS is needed to download and install OP-TEE. The OP-TEE implementation follows the GlobalPlatform[11] standards. We can execute a trusted application on OP-TEE by loading the application in the Rich OS side first. When the application needs to perform a sensitive operation, it uses the API to invoke this operation in the Secure World side (TEE). The TEE sends the response to the REE side for display or other purposes. A debug and web interface are provided to facilitate the management of the platform. Figure 2 represents our architecture of OP-TEE. We use FVP to virtualize OP-TEE which can be run on the top of a Debian OS.

Since almost all the implementations of TEE are proprietary, using such a platform is linked to the signature of agreements with the manufacturers. The use of OP-TEE allows bypassing such restrictions. OP-TEE provides certified APIs and other cryptographic libraries like LibTomCrypt [23] to ease application development. In the next section, we will present our implementation of an IBE in OP-TEE and highlight the advantages of such implementation in this secure environment.

6. A control access system for a smart campus

In previous sections, we saw that IBE is a very interesting mechanism to deliver an encryption key according to an identity. In an environment where the identity is verified by a specific organism (emails for instance), the identity management problem of IBE is avoided. In our case of study, we want to build a trusted access control system for a smart campus based on secure communication between the user and the ”card reader”. Our IBE implementation follows the proposal of Callas’s scheme.

However, as described in section 3, the storage of the master secret of the IBE is challenging. Thus, we consider using OP-TEE in a server to secure the master key storage and the sensitive PKG operations (key generation, encryption and decryption) as described in Figure 3. As previously discussed, an OP-TEE application is divided into two parts: a host application running in normal world and a trusted application running in the secure world. The host sends the data to encrypt to the trusted application where the IBE system is implemented. Figure 3 shows the architecture we select to deploy our secure building access control.

6.1. Proposed architecture

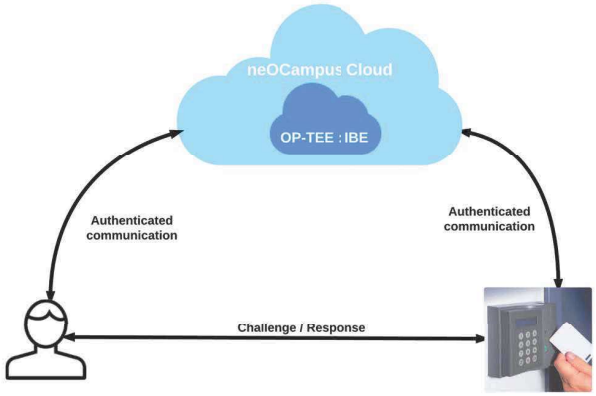

Figure 3. Trusted access control architecture based on IBE

Our architecture is composed of three parties: A user, a secure server where the IBE is deployed and a card reader.

- **User**: The user owns a smartphone acting as an NFC contact-less card. He communicates with the secure server through a secure communication that is assumed to be based on SSL/TLS for authentication purposes and to receive the encryption keys.
- **Secure server**: It consists of an OP-TEE server used to host the IBE mechanism and an Authentication Server. The OP-TEE sever is responsible for the key generation and encryption/decryption operations. The Authentication Server is used to authenticate the users of the system (it can be an LDAP server or an active directory for instance) based on a regular login/password. This authentication server also maintains a database of access rights of every user of the system.

3. Qemu is a generic and open source machine emulator and virtualizer

- **The card reader**: Basically, it is a classical card reader. It can either authorize or decline the access to a user based on the answer provided by the authentication server. Only legitimate readers are able to communicate with the secure server. To achieve mutual authentication, it relays challenges and responses exchanged between the secure server and the user to determine if the user is legit.

The current card used now in the campus are not able to perform a user and/or reader authentication therefore the use of smartphone-based solution improves the overall security. In our proposal, both reader and user are authenticated by the secure server. Indeed, the used IBE mechanism enforces that if the user and the reader can use its identity to decrypt the received message, they are legitimate entities of the system.

6.2. Challenges

At the time of writing, a proof of concept of the proposed trusted access control system is under development to make a performance evaluation. But, several challenges have still to be tackled.

6.2.1. Encryption key storage for a decentralized solution. : The current proposition relies on a fully centralized solution where every cryptographic operation are performed in the secure server. But, it is investigated to provide a decentralized solution where the smartphone can be used to perform parts of the encryption/decryption operations. To achieve this, TEE solutions has to be deployed in the smartphone to store encryption keys and to perform cryptographic operation in a secure way.

6.2.2. Identity management. : In an IBE-based system, the management of the identities is critical. In our use case the identity of a user can be his email address or a student card number. With this option, we are sure that every user has an exclusive identity. But the identity of readers has also to be enforced before a large dissemination of the system.

6.2.3. neOCampus cloud authentication. : If a user and/or a reader wants to use the trusted access control system, he has to be authenticated to the neOCampus cloud. We propose to use the existent architecture based on a LDAP server or an Active Directory.

6.2.4. Deal with the NFC timeout. : The NFC communication has a timeout of 5 ms [24]. It is critical to design an authentication protocol which can tackle these time constraints. Another solution consists of dividing the communication into two steps. In the first step, the user requests reader information and communicates with the secure server. In the second step, he transmits the encrypted challenge by tapping his smartphone on the reader. This solution can limit the impact of the communication time with the secure server on the timeout.

6.2.5. scaling problem. : The number of students using this system to authenticate can be very large. Moreover, they usually come up at the university campus at the same time so that we have to be sure that the proposed system is responsive enough. This issue is linked to the problem of using a fully centralized solution that may not be able to support a large number of simultaneous communications.

7. CONCLUSION

In this paper, we proposed a secure access control system based on TEE and IBE for university campus. First, we gave a detailed presentation of IBE highlighting the pros and cons. Then we exposed the TEE architecture and presented OP-TEE that provides a secure OS enabling secure storage of encryption keys and secure computation of cryptographic operation that improves the security of the IBE's PKG. A trusted access control architecture is then proposed based on a TEE cloud architecture relying on OP-TEE. As a future work, we plan to improve the overall access control protocol with a more decentralized solution. More specifically, it is plan to use the TEE on the smartphone to embed securely the different encryption keys and to perform some cryptographic operations to reduce the number of communications with the secure cloud. From the secure server standpoint, it is investigated to implement our solution on a JUNO [22] card which is an hardware based TEE to improve scalability.

Acknowledgment

This research is part of the neOCampus[6] project to promote a demonstrator of connected campus, innovative, smart and durable. The authors wish to thank the committee for its support.

References

[1] A. Mitrokotsa, M. Beye, and P. Peris-Lopez, *Unique Radio Innovation for the 21st Century*, ch. Security Primitive Classification of RFID Attacks, pp. 39–63. Springer, 2011.

[2] A. Mitrokotsa, M. R. Rieback, and A. S. Tanenbaum, "Classifying RFID attacks and defenses," *Information Systems Frontiers*, vol. 12, no. 5, pp. 491–505, 2010.

[3] R. Housley and T. Polk, *Planning for PKI: best practices guide for deploying public key infrastructure*. John Wiley & Sons, Inc., 2001.

[4] A. Shamir, "Identity-based cryptosystems and signature schemes," in *Advances in Cryptology, Proceedings of CRYPTO '84, Santa Barbara, California, USA, August 19-22, 1984, Proceedings*, vol. 196 of *Lecture Notes in Computer Science*, pp. 47–53, Springer, 1984.

[5] A. Juels and M. Szydlo, "Attribute-based encryption: using identity-based encryption for access control." http://www.arijuels.com/wp-content/uploads/2013/09/JS04.pdf. [Online; accessed 12 April 2016].

[6] "neOCampus." https://www.irit.fr/neocampus/. [Online; accessed 12 April 2016].

[7] D. Boneh and M. K. Franklin, "Identity-based encryption from the weil pairing," in *Advances in Cryptology - CRYPTO 2001, 21st Annual International Cryptology Conference, Santa Barbara, California, USA, August 19-23, 2001, Proceedings*, vol. 2139 of *Lecture Notes in Computer Science*, pp. 213–229, Springer, 2001.

[8] B. Waters, "Efficient identity-based encryption without random oracles," in *Advances in Cryptology - EUROCRYPT 2005, 24th Annual International Conference on the Theory and Applications of Cryptographic Techniques, Aarhus, Denmark, May 22-26, 2005, Proceedings*, vol. 3494 of *Lecture Notes in Computer Science*, pp. 114–127, Springer, 2005.

[9] J. Callas, "Identity-based encryption with conventional public-key infrastructure," *PGP Corporation Palo Alto, California, USA jon@ pgp. com*, 2005.

[10] X. Wang, D. Feng, X. Lai, and H. Yu, "Collisions for hash functions md4, md5, haval-128 and ripemd.," *IACR Cryptology ePrint Archive*, vol. 2004, p. 199, 2004.

[11] GlobalPlatform, "TEE System Architecture." http://www.globalplatform.org/specificationsdevice.asp.

[12] ARM, "ARM Security Technology. Building a Secure System using TrustZone Technology." http://infocenter.arm.com/help/topic/com.arm.doc.prd29-genc-009492c/PRD29-GENC-009492C_trustzone_security_whitepaper.pdf. [Online; accessed 12 April 2016].

[13] "M-Shield mobile security technology, techical report." http://focus.ti.com/pdfs/wtbu/ti_mshield_whitepaper.pdf. [Online; accessed 12 April 2016].

[14] "Sierra TEE virtualization system for TrustZone."

[15] "Genode operating system framework."

[16] N. Santos, H. Raj, S. Saroiu, and A. Wolman, "Trusted Language Runtime (TLR): enabling trusted applications on smartphones," in *Proceedings of the 12th Workshop on Mobile Computing Systems and Applications*, pp. 21–26, ACM, 2011.

[17] "T6 : The TrustedKernel secure OS for TrustZone processors." https://www.trustkernel.com/products/tee/t6.html. [Online; accessed 12 April 2016].

[18] "OP-TEE official wiki page." https://wiki.linaro.org/WorkingGroups/Security/OP-TEE. [Online; accessed 12 April 2016].

[19] "LINARO security working group official website." https://wiki.linaro.org/WorkingGroups/Security. [Online; accessed 12 April 2016].

[20] Qemu, "Qemu, a generic and open source machine emulator and virtualizer.." http://wiki.qemu.org/Main_Page. [Online; accessed 12 April 2016].

[21] ARM, "Fixed Virtual Platform." http://www.arm.com/products/tools/models/fast-models/foundation-model.php. [Online; accessed 12 April 2016].

[22] ARM, "Juno ARM Development Platform." http://www.arm.com/products/tools/development-boards/versatile-express/juno-arm-development-platform.php. [Online; accessed 12 April 2016].

[23] LibTomCrypt, "LibTomCrypt cryptographical library." http://www.libtom.net/. [Online; accessed 12 April 2016].

[24] ISO/IEC, "NFC Technology Full specification." https://www.iso.org/obp/ui/#iso:std:53424:en. [Online; accessed 12 April 2016].