Using Asynchronous Collaborative Attestation to Build A Trusted Computing Environment for Mobile Applications

Lei Zhou
School of Information Science and Engineering, Central South University,
Changsha, 410083, China
Email: l.zhou@csu.edu.cn

Fengwei Zhang
Department of Computer Science,
Wayne State University,
Detroit, 48202, USA
Email: fengwei@wayne.edu

Guojun Wang*
School of Computer Science and Educational Software,
Guangzhou University,
Guangzhou, 510006, China
*Correspondence to: csgjwang@gmail.com

***Abstract*—Nowadays, mobile applications like mobile payments become more popular in public life, but users are eager to acquire a trusted execution system to protect its secrets. This paper presents the design, implementation, and evaluation of the Trusted Computing Environment for the Mobile Application, which protects the integrity and confidentiality of the software from the risk of untrusted mobile platform. The mechanism based on TrustZone and non-interactive verifiable computing provides the Asynchronous Collaborative Attestation Mechanism (ACAM) to runtime attest the active system. TrustZone is a strong hardware-feature based isolated execution technique, but it can not defend against the timing attacking. ACAM can effective prevent this kinds of risk by asynchronous remote attestation and reduces the overhead from real-time protection. The security analysis and evaluation shows that the approach has major potential in mobile trusted computing system and provides a higher secure level environment for mobile users.**

***Keywords*—Trusted Computing Environment; Asynchronous Collaborative Attestation; TrustZone; Verifiable Computing;**

I. Introduction

In recent years, mobile applications enter an explosive growth stage that it greatly facilitate the users to enjoy the Internet service. For example, the enterprise are more desirable to offer the mobile device to the employees for company business. However, to prevent the leakage of company secrets which caused by malicious program running in the employee's mobile device is a challenge [1].

One may hope the Access Control Policy (ACP) can produce a secure computing environment for the company's special software [2]. Unfortunately, increasing sophistication of malware and illegal operations make them detection more difficult. the application of ACP becomes more complexity and leaves more vulnerabilities that can be attacked by malware.

To build a mobile Trusted Execution Environment (TEE) simply and effectively, typical TEE techniques have proposed for local machine security attestation like fTPM [3], Moat [4], etc. Those approaches are eager to offer a trust base in its low lever system, and create a trusted chain to high-level applications. Also, research has proposed the approximative real-time attestation mechanisms by hardware-feature based isolated execution technique. These solutions each have coverage limitations and introduce substantial overhead. The problem is especially critical for mobile environment where memory limitations and energy cost of detection impose substantial limits on the resources that a system can dedicate to remote malware detection. In addition, mobile system and applications tend to update rapidly to meet the newest requirement, it will result in a worse situation due to the continuously changing of security policy and its trust base.

To solve the shortcoming of local TEE techniques that the mobile devices may not be trustworthiness and the real-time local attestation further occupy the computing ability, remote attestation techniques have been proposed to offer a lightweight and effective methods for application verification of remote mobile client. Equipped with a remote attestation method, users are more easy to prevent a remote illegal software from accessing the secrets and legal program. The Third Trusted Party (TTP) [5] is a common technique which usually used to verify the certification of remote software. Khaldi et al. [6] proposed a new way to promise module safe in untrusted entity by introducing the TTP to protect data integrity and service validation. While the limitation of the remote attestation is obviously that attester will be failure to catch a trust result from the attestee process if the attestee work an error interactive communication with the attester. Furthermore, remote attestation brings a time delay to attester which might be attacked by timing attacks.

Thus, building completely real-time trusted remote system is not a easy work. In this paper, we propose an asynchronous collaborative attestation mechanism to implement an interactive attesting on the mobile client's system, which combines the advantages of two techniques: TrustZone-based TEE [7]–[9] and verification computing based remote attestation [10]–[12]. TrustZone is a newest special secure execution platform that the normal program cannot access its running space. Verifiable computing is a non-interactive approach to attest the correctness of remote client's functions. Then, implementing such a secure architecture can effective avoids the disadvantages of other similar methods.

In summary, our goal in this paper is eager to provide a realtime trusted device for the mobile clients, The work is proposed and makes the following contributions. First, we provide a complete design of the asynchronous collaborative attestation architecture describing the components running in the trusted environment, in the OS, and in the untrusted application. Second, we describe our scheme about secure system implementation with TrustZone and verifiable computing methods. We present the challenges related to the TrustZone compatibility and portability issues of target mobile system, namely Android. Third, we present a security analysis and evaluation of our system.

The paper is organized as follows. In Section 2 we discuss the related work. We provide asynchronous collaborative attestation system architecture in Section 3. In Section 4 we analysis the security and efficiency about our approach. Finally, we wrap up with the conclusion in Section 5.

II. Related Work

The ultimate goal for attestation system is to build a security execution environment for the mobile user. In this section, we present our assumptions and threat model, describe the TrustZone based detection mechanism and VC-based remote attestation, and discuss some of our design choices about how to make a clean execution.

A. Assumptions and Threat Model

We define such a scenario: Company (Enterprise-employer) provide an office environment for its employee. In the era of the information, employees are eager to work on computer device to improve their efficiency. However it creates several problems, the most serious one is the security risk. For example, employer provides the own devices for employees to run the company owned service applications. Meanwhile, users are hope to run some personal software for privacy and secrets.

In summary, the security requirements defined as follow:

- employer fear that the employees install malicious software or run illegal operations in the mobile system to leak the company's secrets.
- employees fear that the employer install malicious software in the mobile system to steal the employees' personal information.

However, the attacks from the malicious application and the service provider are easy to cause a leakage of the mobile system secrets. The features of malicious service provider:

- Provide the device to user and it have bundled the own software into the device's custom software package.
- Some unknown software has been tied up on normal functions.
- Those unknown software are hard to discriminate between the benign and the malicious.
- Those unknown software are hard to remove which get the high privilege and can easy to reinstall once the user update it's system from the same malicious service provider.

Thus, the best way to solve this problem is to construct an isolate execution environment in mobile system, and limit the running software. However it is hard to reject the software install in mobile device system if they still enjoy the convenience from the large-scale service provider, the feasible method to stop those malicious software running is to construct a trusted execution environment (TEE) in the system which has the higher privilege than the bundled software from SP.

In our threat model, an adversary is capable of attacking mobile system and putting malicious software into user's platform, and it cannot be found by the user, because the mobile user does not know what is running in its mobile computing environment at initial stage and whether the system is compromised, because the mobile device are provided by the employer at first. Then, when the mobile application running in the system tend to update or replaced by a new one, malicious applications are easy to invade the mobile system. To solve those problems, we intent to design a ACAM method to build a trusted execution environment in the mobile system.

B. TrustZone for Isolated Execution Environment

Various of Access Control Policy (ACP) [2] are purposely to assure the correctness of computational resources access. It is effectively most of time, but become more and more complexly. Also, the execution of ACP is based on the operation system, if some kernel-based attacks happen in mobile system, all ACPs are seem to lost their effect. Furthermore, even the enterprise $Apple$ do the most strict access control policy and encrypt algorithms, for example, people who want to bypass the security mechanism by using brute-force, the device will start the "auto-erase" function, which use to erase all the data in device. But the $Apple$'s own software $custom - signed$ can easy bypass those functions and get the key data by using an unknown backdoor. Besides, this is the worse situation in other types of system, like Android and Microsoft Windows.

Hence, the new security mechanism should be improved to protect our system security from the attacks which implemented by the malicious service provider. The famous mobile CPU manufacturer has launched the new security hardware-feature security technique - TrustZone [13] on ARM architecture, which is a hardware security extension of the ARM processor architecture.

TrustZone divides the hardware and system software into two worlds: Secure world and normal world, which is able to build an isolate computing environment base on three main reasons: i) The physical memory space can only be accessed by TrustZone-based process; ii) secure world use the special system control; and iii) only few outside approved instructs can be used to trigger the status switching. Hardware barriers are established to prevent normal world components from accessing secure world resources; the secure world is not restricted. Specifically, the memory system prevents the normal world from accessing [14]:

Then, to construct a Rich Execution Environment(REE) in normal world, it can select a general operating system, like Android or Linux. However, the system running in secure world should be OP-TEE or Customized Linux [15] because of low computational ability and memory space. TrustZone is regarded as a external security area that limited number of partners (device manufacturers and TrustZone OS providers) can deploy the TEE services [16], which is due to the TEE security level and trustworthiness is maintained by adopting restrictions, and only strictly verified applications can be deployed in the TEE. Thus, we define those verified applications as trust base, and modify it as verification functions, to build a trust chain to normal world for universal applications attestation. This can be researched to reach our ultimately goal for designing $clean$ system, a framework whereby individual users can build an isolated execution environment for its applications.

C. Non-interactive Verifiable Computing Mode

Except the traditional cryptographic algorithm for mobile data protection, the effective method to prevent the secrets leakage from illegal software attacking the mobile system is to build a Trusted Execution Environment based on Trusted Computing or other secure hardware-based approach. While the TPM-based and the TrustZone-based TEE approaches have its weakness for verifying the dynamic changed mobile applications, while the remote attestation method is better choice to runtime mobile security verification.

Non-interactive verifiable computing (NIVC) is a concrete system for efficiently verifying general computations while making only cryptographic assumptions. In particular, NIVC supports public verifiable computation [17], [18], which allows an untrusted worker to produce signatures of computation. Then, to be a provable security algorithm, the verification system update it as a remote attestation method to verify whether the functions in untrusted system can work out the correct computing.

Initially, the challenger chooses a function and generates a public evaluation key and a public verification key. The public evaluation key and the encrypted input are sent to untrusted worker. The worker can choose one of the inputs (or verifiably use one provided by the challenger), compute the function, and produce a proof (or signature) to accompany the result. Anyone (not just the challenger) can then use the verification key to check the correctness of the worker's result for the specific input used. As an additional feature, NIVC supports zero-knowledge verifiable computation, in which the untrusted work convinces the challenger that it knows an input with a particular property, without revealing any information about the input.

Based on above features on NIVC, the remote attestation can be designed as a lightweight scheme and is feasible to build a real-time attestation system. we analysis that only key functions in attestee are attested and its trust base can be provided by TrustZone-based attestation.

III. Design

To access the feasibility and effectiveness of our approach, we combine TrustZone-based local attestation and remote attestation together, to build a entire life cycle applications attestation environment. In this section, we need to modify existing technology to develop optimized model. The architecture is shown in Figure 1.

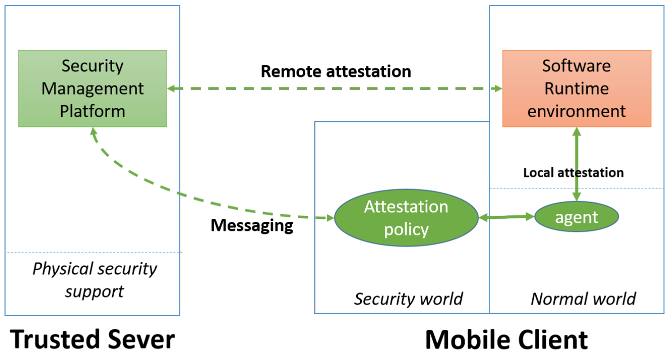

Fig. 1. The Architecture of Asynchronous Collaborative Attestation System

The model of the asynchronous collaborative attestation system contains two main parts: Trusted Sever (TS) and Mobile Client (MC). The TS is regard as trusted third parts to store the security policy and implement the management, its security is promised by other physical or software isolation mechanism which does not analysis in this paper. MC is divided into two components: The secure world and normal world. These is a trusted agent in normal world which is set of TrustZone driver, library or some other additional functional functions, and it is protected by TEE. Messaging between mobile TEE and TS is designed to share the security message and promise a correct asynchronous collaborative attestation.

A. TrustZone for Initial/Idle-Time Attestation

TrustZone is eager to reach the goal of providing an isolated execution environment (IEE) for mobile security software. Thus, we design a initial/idle-time attestation mechanism based on those IEE. The trusted attestation function running on isolation environment by TrustZone can be explained from two aspects:

(1) Inter Trusted Messaging: REE (in normal world) and TEE(in security world) are statically separated by isolate memory, when a REE process employs resources from the TEE, it is necessary to construct a communication channel for the transmission of messages between the two domains. The channel is normally to use the shared memory which can be accessed by those two domains. Before a REE application running, we verify its integrity and keep the original value id_o on TEE security store [19]. We design a monitoring function and bind it with TrustZone driver. The function is used to record the software running information including the initiation, implementing and stop steps on the shared memory space. Moreover, the functions tend to trigger the secure-monitor call (SMC) instruction and switch the REE to TEE for next attestation.

(2) Process of Software Attestation: After the argument of software entry point address is transferred to TrustZone by

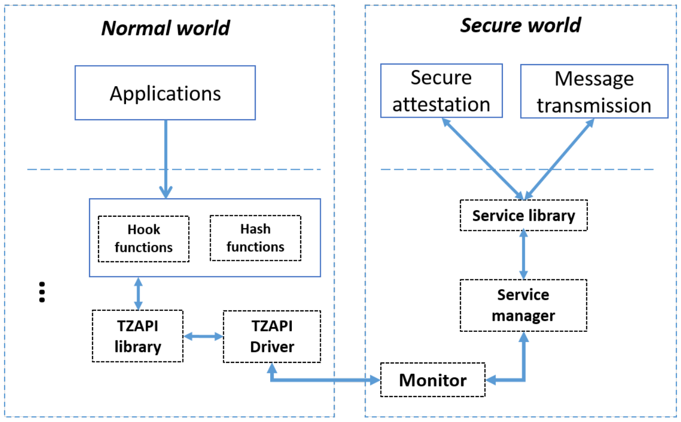

Fig. 2. The Process of TrustZone-based Attestation

aspect 1, the attest functions are able to access those memory blocks [20], and computer the integrity of verified software like this:

We get a first address a of mobile application A store in untrusted memory space and its size $size$, and we develop a memory acquisition function F_{amem} to fetch the code and data of software $mem(A)$:

$$mem(A) = F_{amem}\,(a, size)$$

Then, we get a more specific expression.

$$id = f_{SHA-256}\,(mem(A))$$

$f_{SHA-256}$ is one of the hash functions for identifying the application which store in security area. We get an one-to-one identity mapping between mobile application and hash value.

Before we confirm the id as an identity, the comparing function $F_{compare}$ needs compare with the origin identity id_o. If $1 = F_{compare}(id, id_o)$, we regard the software running in the safe state, 0 vice versa. In addition, there is no id_o compare to the id means an unknown mobile application running. The process are shown in figure 2

(3) External Trusted Messaging: Since the attestation function in TrustZone has been finished the process of mobile application's integrity verification, those application's information are recorded in secure store. Under the work of the optimization function, we get a fixed message including application identity, timestamp, security status, etc. Finally, the message is encrypted and send to TS for next analysis.

B. Remote Verification for Runtime Attestation

With the trust base, mobile client system is eager to design a trusted chain and verifiable system to report its device security status to a challenger. According to the TCG and IMA, the security status of a mobile system is evaluated by measuring integrity of loaded application components in REE. However, TrustZone-based attestation is unable to verify the runtime REE application because the REE should be suspended once the TEE starting. Then, the TrustZone-based attestation work out a static attestation result which defines as an initial status.

To measure the runtime status of each components of the mobile application efficiency and effectiveness, we design a verifiable computing based (VC-based) attestation mechanism, which consists of two main parties: an attestee (the Mobile Client), an attester (attestation challenger). The attestee is required to compute the verified arguments from the attester. Here, the initial trusted system state is assumed by TrustZone-based attestation. After a stage of certain application behaviors, the mobile system is changed to new state.

The initial system status including the mobile application information $Msg(A)$, $Msg(A) = A_{name}, A_{id}, A_{version}, ...$ While $H(x)$ is a list of hash functions, that $H = H_1, H_2, ..., H_t$,

Then, to an application A, a vector $X = H_1(A), ..., H_t(A)$ means its hash value under different hash functions. Thus, the following four steps are used to verify the correctness of application in mobile system.

- Acquire the verified application and input value: $A = F_{select}(Msg(A))$. Since the attester receives the message from attestee, it finds the correct application A by selection function F_{select}. the application parameter input $x = F(A)$, we define $F = H_i$ here, $i \in (1, ..., t)$.
- Generates the key of target application: $KenGen(F, \alpha) \rightarrow (P_k, S_k)$, Based on the security parameter α, the randomized key generation algorithm from attester generates a public key that encodes the application, the secret key is kept private by the attester.
- Transfer the verified argument: $ProGen\ S_k\,(x, i) \rightarrow (\sigma_x, \sigma_i, \varphi_x, \varphi_i)$, the problem generation algorithm from attester uses the secret key S_k to encode the application parameter input x and function parameter input i as the public values σ_x which are given to the attestee to compute with, and a secret value φ_x which is kept private by the server, the same process to input i.
- Compute the verified application: $Compute\ P_k(\sigma_x, \sigma_i) \rightarrow (\sigma_y)$, using the server's public key P_k and the encoded input σ_x, σ_i, the attestee computes an encoded version of the function's output $y = F(x)$.
- Verify the computational result: $\{0, 1\} \rightarrow Verify(S_k, \varphi_x, \varphi_i, y, \sigma_y)$, using the secret key S_k and the secret "decoding φ_x, the verification algorithm from attester converts the attestee's encoded output σ_y into the output of the application, e.g., $y = F(x)$ or outputs 0 indicating that σ_y does not represent the valid output of F on x.

An encrypted attestation result from the attestee is sent to the attester and compare with original status. Runtime Attestation is a high overhead service for the local device, In the current implementation, we directly create a hash computing module and place it into TrustZone-based security area in normal world. If the TrustZone-based attestation finish the initial stage proof, the remote attestation stage begin to go on real-time verification. The attester records the software identity and feedback the MC status switch trigger command.

To keep the communicational messages in a secure state, data from attestee is automatically encrypted when it is sent to TS, We assume that communication channel is safety enough. The messaging mechanism from TS to MC TEE is the same process as the TrustZone-based messaging. However, the message content has some difference that only contains the verified functions's status, the value of application integrity, etc, which use to support the TEE controlling the REE. Before the message transferring, the switch instruct should be triggered at the end of the process of remote attestation.

C. Asynchronous Collaborative Attestation Mechanism

Since TrustZone-based local attestation and VC-based remote attestation can not construct a trusted execution environment individually. The former is due to the suspension of normal world while the system switch to secure world, which will lead to a non-real time verification. The latter is due to lack of trust base to promise the integrity of verified functions even those functions may work out correct result but may additionally do other malicious computing. Thus, we combine those two approaches to build a trusted computing environment for mobile applications by asynchronous collaborative attestation, which will effectively eliminate the negative parts of those two approaches. The following steps are shown in Figure 3.

Booting stage: After the secure physical booting on mobile device, mobile system first to start the TrustZone system. Since memory space of REE can be accessed by TEE functions which has the higher privilege, TrustZone-based attestation will check the integrity of memory block about REE software. The original integrity value has been store in TrustZone secure area, it provide a $clean$ state REE for users.

Runtime stage: i)once a mobile application begin to run in the system, hooking function in REE kernel catch this message and will trigger the SMC instruct to switch the system to TEE. The application information (including memory address, size,etc.) will send to TEE at the same time. Then, the TrustZone-based attestation method verify its integrity and transfer those result to attester. ii) After received a initial trusted application, the remote attestation will runtime monitor the status of process. It will prompted the hash function in TrustZone-Protect area to compute the integrity value of application, and verify it on attester part. iii) A little function added in remote attestation module which record the idle time and trigger the switch to TrustZone-based attestation again.

In the summary, the safety of the mobile system is determined by the work of those two different attestation methods. It is possible that adversary succeeds to produce an malicious application to stole the secrets from the key process. We design a trust base with TrustZone-based attestation method to verify the initial system environment. And with the asynchronous collaborative attestation method to promise a runtime security system.

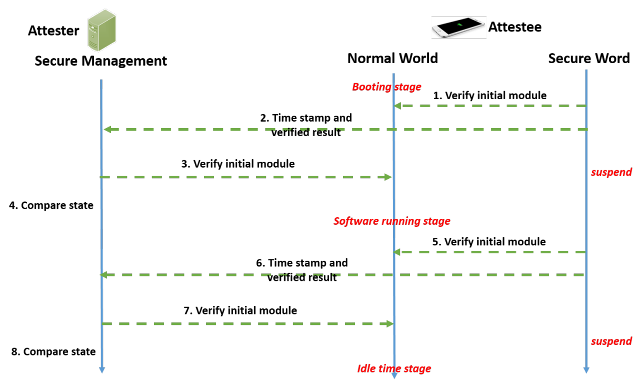

Fig. 3. The Process of RA-based Attestation

IV. Security Analysis And Evaluation

The asynchronous collaborative attestation method oriented to shared mobile computing services, it solves the attestation of trusted status of mobile application running environment in mobile device system, which promise a trusted execution environment for tenants to enjoy the employer's services and protect the secrets leakage from malicious attacking and internal illegal operation by employers. Compared with the TCG remote attestation mechanism, TrustZone-based attestation and individually VC-based remote attestation mechanism, it has following characteristic.

First, through TCG-technique research has been proposed, we solve the popularized mobile client with TrustZone technique and without special TPM hardware. Not only the initial time of system, we can attest the execution environment under the system idle time, which improves the security and do not work a bad effect to running mobile application.

Second, TrustZone-based attestation is unable to runtime attest the normal system application running, but it can provide a trust base for remote attestation, which can runtime check the application integrity without blocking the normal process running.

Third, the messaging mechanism between the remote attester and local TrustZone system will share the security information about the normal world application, which improves the reliability of the asynchronous collaborative attestation method.

A. Security Analysis

Throughout section 3, we discussed in detail how asynchronous collaborative attestation method builds a trusted computing environment for mobile clients, including remote function verification and local application attestation. In this section, we first summarize how these features fulfill the required security guarantees. Afterwards, we discuss how the system prevents other possible attack scenarios.

Security Guarantees: As mentioned in Section 3, $ACAM$ provides two principal security guarantees. First, the TrustZone technique guarantees that the REE cannot break the TEE's isolation. Second, it guarantees that switching from the normal world to the security world cannot expose the address space protection.

B. *Efficiency*

Attacks against the Trusted Environment: If the malicious application bypass the REE's conventional defense mechanism, and hajack the TrustZone-driver or other key functions. VC-based attestation will encrypted compute the integrity of those functions. If there is no or an error result was acquired, it will trigger the TrustZone instruct for next attestation. Nevertheless, the exact same risk faces the cross-validation by those two mechanisms.

In fact, ACAM profoundly enhances the system security in this case. If vulnerability exists in the REE security tool, then the extent of the attack will be limited to the same privilege level of the kernel. On the other hand, if the same security tool is hosted by the hypervisor or by TrustZone, then such attack would have an even higher impact by compromising these security sensitive system components.

V. Our Construction

We introduced the asynchronous collaborative attestation mechanism (ACAM), to build a trusted computing environment for running mobile application. We assume that the REE applications may be attacked by malicious software to generate a risk for user's secrets leakage. ACAM splits the mobile system into trusted and untrusted domain by leveraging TrustZone and Non-interactive Verifiable Computing approaches. Then, ACAM protects the entire life cycle of REE applications while the TrustZone-based attestation method spends an initial and idle-time verification, and VC-based remote attestation method provides the runtime verification. The messaging mechanism between the remote attester and the local TrustZone is under effective encrypted approach with the aid of the secure channel, to promise the security of the remote attestation. Through analysis of the security and efficiency evaluation, ACAM effectively provides a trusted environment for mobile clients.

similar approach has been tested by other teams, however, we proposed a new model and implementation scheme, it still has a lot of work to improve our ultimately goals. Our future work will focus on building a trusted and non-repudiation protocol between mobile clients and services provider under these proposed techniques, like hardware-assisted interrupts, remote verification, etc.

VI. Acknowledgments

This work is supported in part by the National Natural Science Foundation of China under Grant Numbers 61632009 and 61472451, and High Level Talents Program of Higher Education in Guangdong Province with Funding Support Number 2016ZJ01.

References

[1] S. Chakraborti, D. P. Acharjya, and S. Sanyal, "Application security framework for mobile app development in enterprise setup," *Computer Science*, 2015.

[2] R. Abdunabi, I. Ray, and R. France, "Specification and analysis of access control policies for mobile applications," in *ACM Symposium on Access Control MODELS and Technologies*, 2013, pp. 173–184.

[3] H. Raj, S. Saroiu, A. Wolman, R. Aigner, and etc, "ftpm: A software-only implementation of a TPM chip," in *25th USENIX Security Symposium, USENIX Security 16, Austin, TX, USA, August 10-12, 2016.*, 2016, pp. 841–856.

[4] R. Sinha, S. Rajamani, S. Seshia, and K. Vaswani, "Moat: Verifying confidentiality of enclave programs," in *ACM Sigsac Conference on Computer and Communications Security*, 2015, pp. 1169–1184.

[5] P. X. Wang and H. Q. Zhou, "Research on cloud security model based on trusted third party on multi-tenant environment," *Computer Science*, 2014.

[6] A. Khaldi, K. Karoui, N. Tanabene, and H. Ben Ghzala, "A secure cloud computing architecture design," in *IEEE International Conference on Mobile Cloud Computing, Services, and Engineering*, 2014, pp. 289–294.

[7] N. Santos, H. Raj, S. Saroiu, and A. Wolman, "Using arm trustzone to build a trusted language runtime for mobile applications," vol. 49, no. 4, pp. 67–80, 2016.

[8] M. Jeon, S. Kim, and H. Yoo, "Inter-guestos communications in multicore-based arm trustzone," vol. 42, no. 5, pp. 551–557, 2015.

[9] J. Jang, C. Choi, J. Lee, N. Kwak, S. Lee, Y. Choi, and B. Kang, "Privatezone: Providing a private execution environment using arm trustzone," vol. PP, no. 99, pp. 1–1, 2016.

[10] V. Vu, S. Setty, A. J. Blumberg, and M. Walfish, "A hybrid architecture for interactive verifiable computation," in *Security and Privacy*, 2013, pp. 223–237.

[11] H. Lipmaa, *Succinct Non-Interactive Zero Knowledge Arguments from Span Programs and Linear Error-Correcting Codes*. Springer Berlin Heidelberg, 2013.

[12] R. Gennaro, C. Gentry, and B. Parno, *Non-interactive Verifiable Computing: Outsourcing Computation to Untrusted Workers*. Springer Berlin Heidelberg, 2010.

[13] J. Jang, S. Kong, M. Kim, D. Kim, and B. B. Kang, "Secret: Secure channel between rich execution environment and trusted execution environment," in *Network and Distributed System Security Symposium*, 2015.

[14] P. Colp, J. Zhang, J. Gleeson, S. Suneja, E. De Lara, H. Raj, S. Saroiu, and A. Wolman, "Protecting data on smartphones and tablets from memory attacks," *ACM SIGPLAN Notices*, vol. 50, no. 4, pp. 177–189, 2015.

[15] H. Sun, K. Sun, Y. Wang, and J. Jing, "Trustotp: Transforming smartphones into secure one-time password tokens," in *ACM Sigsac Conference on Computer and Communications Security*, 2015, pp. 976–988.

[16] W. Enck, P. Gilbert, S. Han, V. Tendulkar, B. G. Chun, L. P. Cox, J. Jung, P. Mcdaniel, and A. N. Sheth, "Taintdroid: An information-flow tracking system for realtime privacy monitoring on smartphones," in *Usenix Symposium on Operating Systems Design and Implementation, OSDI 2010, October 4-6, 2010, Vancouver, Bc, Canada, Proceedings*, 2014, pp. 393–407.

[17] C. Costello, C. Fournet, J. Howell, M. Kohlweiss, B. Kreuter, M. Naehrig, B. Parno, and S. Zahur, "Geppetto: Versatile verifiable computation," pp. 253–270, 2015.

[18] B. Parno, J. Howell, C. Gentry, and M. Raykova, "Pinocchio: Nearly practical verifiable computation," in *IEEE Symposium on Security and Privacy*, 2013, pp. 238–252.

[19] X. Ge, H. Vijayakumar, and T. Jaeger, "Sprobes: Enforcing kernel code integrity on the trustzone architecture," *Computer Science*, 2014.

[20] H. Sun, K. Sun, Y. Wang, and J. Jing, "Reliable and trustworthy memory acquisition on smartphones," *IEEE Transactions on Information Forensics & Security*, vol. 10, no. 12, pp. 2547–2561, 2015.