DAA-TZ: An Efficient DAA Scheme for Mobile Devices using ARM TrustZone
(Full Version)*

Bo Yang[1], Kang Yang[1], Yu Qin[1], Zhenfeng Zhang[1], and Dengguo Feng[1,2]

[1] Trusted Computing and Information Assurance Laboratory
Institute of Software, Chinese Academy of Sciences, Beijing, China
[2] State Key Laboratory of Computer Science, Institute of Software
Chinese Academy of Sciences, Beijing, China
{yangbo,yangkang,qin_yu,zfzhang,feng}@tca.iscas.ac.cn

Abstract. Direct Anonymous Attestation (DAA) has been studied for applying to mobile devices based on ARM TrustZone. However, current solutions bring in extra performance overheads and security risks when adapting existing DAA schemes originally designed for PC platform. In this paper, we propose a complete and efficient DAA scheme (DAA-TZ) specifically designed for mobile devices using TrustZone. By considering the application scenarios, DAA-TZ extends the interactive model of original DAA and provides anonymity for a device and its user against remote service providers. The proposed scheme requires only one-time switch of TrustZone for signing phase and elaborately takes pre-computation into account. Consequently, the frequent on-line signing just needs at most three exponentiations on elliptic curve. Moreover, we present the architecture for trusted mobile devices. The issues about key derivation and sensitive data management relying on a root of trust from SRAM Physical Unclonable Function (PUF) are discussed. We implement a prototype system and execute DAA-TZ using MNT and BN curves with different security levels. The comparison result and performance evaluation indicate that our scheme meets the demanding requirement of mobile users in respects of both security and efficiency.

Keywords: DAA, Privacy, Mobile Devices, ARM TrustZone, PUF.

1 Introduction

With the development of wireless communication network as well as modern mobile devices, a variety of mobile applications have been realized to provide users convenient and comprehensive services. Depending on these achievements, online interactive applications such as mobile payment, mobile ticketing, mobile shopping and mobile voting, are benefiting people's daily lives. However, with the widespread use of mobile services, users are faced with the risk of privacy

* An extended abstract of this paper appears in TRUST 2015.

disclosure. Generally, authenticating the users' legitimate identity is regarded as one prerequisite for access to those remote application services. This authentication is associated with an individual mobile device, a SIM card or a service account. As a result, when a user logins to enjoy services, his personal information, perhaps involving his real identity, locations, bank accounts or records of network behaviors etc., is potentially linked to each other and leaked to service providers [33]. And what is worse, the personal information could be further shared with some third parties, for example, to send consumers behaviorally targeted advertisements [12]. Thus, the issue of information leakage is seriously threatening mobile users' personal privacy and information security.

On PC platform, the analogous problem can be effectively solved by DAA [3], which is standardized by the Trusted Computing Group (TCG). DAA allows an embedded processor on a motherboard, called Trusted Platform Module (TPM), to anonymously attest the certain statements about the configuration of the host machine as well as its legitimate status to remote third parties [2]. The key requirement behind DAA is that this attestation is done in a way that maintains the privacy of the machine (i.e., the user). On the other hand, DAA is an anonymous credential system designed specifically to encapsulate security-critical operations within TPM, and the sensitive data including secret signing key and parameters are well protected by TPM. An adversary hardly shares a legitimate user's credential by just stealing related data on the host to gain unauthorized access to remote services. The DAA [3] is originally proposed based on strong RSA assumption. For better computing efficiency and shorter signature length, researches constructs several DAA schemes based on elliptic curves and bilinear maps [4,11,5,10], which we call ECC-DAA. Moreover, Chen [8] designs a DAA scheme requiring less TPM resources. Bernhard et al. [2] give a new security model and a generic construction of DAA protocol on it. Xi et al. [32] first add the property of forward anonymity. To date, DAA has gained lots of favor with industry and standard bodies [9,27,28], which renders it better prospects for practical applications than other anonymous credential systems [34].

For mobile platform, DAA is still attractive as an alternative anonymous authentication solution. Unfortunately, previous DAA schemes are exclusively designed for the model of TPM inside a host. The prevalent mobile devices are rarely equipped with special-purpose chip like TPM, so that the direct use of DAA on mobile platform would cause trouble. Opaak [17] is a simplified DAA scheme for mobile devices, but its executable codes and sensitive data are easily compromised or stolen by malwares.

The technique of Trusted Execution Environment (TEE) on mobile devices could lend us a helping hand. Isolated from a Rich Execution Environment (REE) where the Guest OS runs, TEE aims to protect sensitive code execution and assets. As an example of providing TEE for embedded devices, ARM TrustZone [31] has been used to execute security-critical services [26,25]. By TrustZone, TEE's resources are physically isolated from REE, such that adversaries in REE hardly access them directly [15]. As a hardware-based security extension of ARM architecture, TrustZone is widely supported and applied by

many mobile manufacturers. Leveraging TrustZone, some solutions are proposed to construct software-based ECC-DAA with security-critical codes running within TEE, and treat REE as the role of "host". Wachsmann et al. [30] put forward an authentication scheme based on DAA and TLS but without user-controlled linkability. Yang et al. [33] present LAMS for anonymous mobile shopping, which is compatible with four unmodified ECC-DAA schemes. Given by Zhang et al. [34], Mdaak provides a general and flexible DAA framework for mobile devices. Nevertheless, these solutions neither implement pre-computation for anonymous signing nor consider comprehensive protection for DAA sensitive data. Furthermore, in these solutions, implementing unmodified or simply modified DAA in TrustZone brings much extra meaningless overhead and security issues. First, DAA does not concern the number of interactions between TPM chip and the host, while it is a problem for mobile devices using TrustZone. Each switching the context between TEE and REE carries both performance overhead and power consumption. In practice, these cannot be neglected, especially if the system or software is complicated, or the switch action needs to be triggered frequently [21]. When many switch actions occur, data transmission and protection for DAA procedure are also cumbersome, memory-consuming and time-consuming. Second, extra operations are demanded in original DAA for the sake of the limited bandwidth of TPM, which seems superfluous for TrustZone. Last but not least, TPM itself is hardware-based root of trust with inside root key for sensitive data management, while TrustZone does not definitely provide this root. To the best of our knowledge, there is no DAA scheme specially designed to adapt for mobile devices using TrustZone.

Our Contributions. In this paper, based on elliptic curves and bilinear maps, we propose an efficient DAA scheme (DAA-TZ) for mobile devices who are resource-constrained as compared with PC platform. DAA-TZ enables remote service providers to authenticate mobile users' trusted status or legitimate information without disclosing users' identity. DAA-TZ makes full use of ARM TrustZone and modifies the traditional interactive model as well as procedure of original DAA. The main signing efficiency for users is improved without the expense of security. Our work is summarized as follows.

- This is the first complete work that designs an efficient DAA scheme deeply integrated with TrustZone and expressly for mobile devices. In order to reduce the time delay and space overhead, the scheme minimizes the switch times of TrustZone for the frequent signing phase.
- According to the ecosystem of mobile devices, DAA-TZ supports manufacturers to acquire a batch of credentials through cooperative and trusted channels, and then embed them into devices before they leave factory. Users could immediately execute anonymous signing after getting devices.
- The pre-computation is carefully added into DAA-TZ, so that the on-line anonymous signing at most needs only three exponentiations on elliptic curve which is thought of as quite expensive computation.

- DAA-TZ utilizes the on-chip SRAM PUF to reproduce a root key seed and further create keys serving different purposes. The mechanism for sensitive data management using related keys is presented in detail.
- We implement a prototype system with full functions of DAA-TZ. The evaluation on it with two types of curves is performed. Both theoretical comparison and testing results show the high efficiency of our scheme.

Paper Outline. We provide the preliminary information in Section 2. Section 3 describes the system model and assumptions. Section 4 details the design for DAA-TZ, which is comprised of the architecture using TrustZone, the sensitive data management and the scheme procedure. Section 5 analyses the security properties that DAA-TZ satisfies. Our implementation and evaluation are showed in Section 6. Finally, Section 7 concludes the whole paper.

2 Preliminaries

2.1 Notation

Throughout this paper, λ denotes the security parameter. We use $a \leftarrow S$ to denote sampling a from a set S uniformly at random. We also use $1_{\mathbb{G}}$ to denote the identity element of a group $\mathbb{G}$. For any group $\mathbb{G}$, $\mathbb{G}^*$ denotes $\mathbb{G}\backslash\{1_{\mathbb{G}}\}$. $\mathsf{MAC}_k(m)$ denotes the message authentication code for a message m computed with the secret key k, and $\mathsf{Enc}_k(m)$ denotes a ciphertext of a message m produced with the symmetric key k.

2.2 Bilinear Groups

Bilinear groups consist of three (multiplicatively written) groups $\mathbb{G}_1$, $\mathbb{G}_2$ and $\mathbb{G}_T$ of prime order p equipped with a bilinear map $e : \mathbb{G}_1 \times \mathbb{G}_2 \rightarrow \mathbb{G}_T$. Let g_1 and g_2 be generators of $\mathbb{G}_1$ and $\mathbb{G}_2$ respectively. We define $\Lambda = (p, \mathbb{G}_1, \mathbb{G}_2, \mathbb{G}_T, e, g_1, g_2)$ to be a description of bilinear groups parameters.
The map $e : \mathbb{G}_1 \times \mathbb{G}_2 \rightarrow \mathbb{G}_T$ must satisfy the following properties:

1. **Bilinear.** for any $u \in \mathbb{G}_1$, $v \in \mathbb{G}_2$ and any $a, b \in \mathbb{Z}$, $e(u^a, v^b) = e(u, v)^{ab}$.
2. **Non-degenerate.** $e(g_1, g_2) \neq 1_{\mathbb{G}_T}$.
3. **Computable.** the map e is efficiently computable.

In this paper, we only consider the Type-3 pairings [13], thus $\mathbb{G}_1 \neq \mathbb{G}_2$ and there is no known efficiently computable isomorphism between $\mathbb{G}_1$ and $\mathbb{G}_2$.

2.3 Cryptographic Assumptions

The security of DAA-TZ is based on the Decisional Diffie-Hellman assumption in $\mathbb{G}_1$ ($\mathrm{DDH}_{\mathbb{G}_1}$) and the blind 4-LRSW (B-4-LRSW) assumption [2] which is a variant of the original LRSW assumption [16]. The $\mathrm{DDH}_{\mathbb{G}_1}$ assumption and the B-4-LRSW assumption are stated as follows:

Assumption 1 ($\mathbf{DDH}_{\mathbb{G}_1}$). Given (g_1, g_1^a, g_1^b) for $a, b \leftarrow \mathbb{Z}_p$, it is hard to distinguish g_1^{ab} from a random element of $\mathbb{G}_1$.

Assumption 2 (B-4-LRSW). Given the bilinear group parameters Λ, (g_2^x, g_2^y) for $x, y \leftarrow \mathbb{Z}_p$ and an oracle that on input of $g_1^m \in \mathbb{G}_1$ outputs $(A, A^y, A^{x+mxy}, A^{my})$ for $A \leftarrow \mathbb{G}_1^*$, it is hard to output $(m^*, A^*, B^*, C^*, D^*)$ such that $m^* \in \mathbb{Z}_p^*$, $A^* \in \mathbb{G}_1^*$, $B^* = (A^*)^y$, $C^* = (A^*)^{x+m^*xy}$ and $D^* = (A^*)^{m^*y}$ where $g_1^{m^*}$ was never queried to the oracle.

2.4 ARM TrustZone

ARM TrustZone [20] is a hardware-based security extension technology incorporated into ARM processors. It enables a single physical processor to execute codes in one of two possible operating worlds: the normal world and the secure world. Accordingly, the system is separated into two domains and each domain has banked registers and memory to run the domain-dedicated OS and software. The isolation mechanisms of TrustZone are well defined. Access permissions are strictly under the control of the secure world that normal world components cannot access the secure world resources. As the processor only runs in one world at a time, to run in the other world requires context switch. A secure monitor mode exists in the secure world to control the switch and migration between the two worlds. To date, TrustZone has been popularized and applied by many mainstream mobile manufacturers to achieving secure applications [33].

2.5 Physical Unclonable Functions

Physical Unclonable Functions (PUFs) [23] are functions where the relationship between input (or challenge) and output (or response) is decided by a physical system. Randomness and unclonability are two significant properties of PUFs. The unclonability originates from random variations in a device's manufacturing process. With the help of a fuzzy extractor that eliminates the noise from the response, PUFs are able to implicitly "store" a piece of secret data. PUFs provide much higher physical security by extracting the secret data from complex physical systems rather than directly reading them from non-volatile memory. Additionally, PUFs are cost-effective, since they take the advantage of the results from a preexisting manufacturing process[34].

Strictly speaking, TrustZone just provides an isolated environment. Only equipped with a root of trust, it becomes a real "trusted" execution environment (TEE) [35]. Because TrustZone almost does not internally install an available root key, it loses the capability to offer a root of trust. To cover this shortage, a PUF can be employed to properly act as the root of trust. In this paper, DAA-TZ takes the secret data extracted from the PUF as a root key seed to generate other keys. We adopt SRAM PUF [14] that leverages the relationship between an SRAM cell's address for the challenge and its power up value for the response.

3 System Model and Assumptions

3.1 System Model

The system model of DAA-TZ is composed of four kinds of entities: mobile device $\mathcal{D}$, manufacturer $\mathcal{M}$, issuer $\mathcal{I}$ and verifier $\mathcal{V}$. In practice, there could be a number of $\mathcal{D}$ and $\mathcal{V}$, thus we use $\mathcal{D}_i$ and $\mathcal{V}_j$ to represent an individual unite respectively. $\mathcal{D}_i$ is directly accessed by a user and equipped with ARM processor having TrustZone extension technology. $\mathcal{M}$, who produces $\mathcal{D}_i$, performs embedding some credentials to each $\mathcal{D}_i$ in advance before it leaves factory. $\mathcal{I}$ is responsible for issuing credentials to legitimate (or trusted) $\mathcal{D}_i$. $\mathcal{I}$ could be an independent trusted authority or a part of mobile network provider. The procedure of issuing could be executed with either $\mathcal{M}$ or $\mathcal{D}_i$ respectively. Service providers play the role of $\mathcal{V}$ in this interactive model. $\mathcal{V}_j$ outsources some verification strategies to $\mathcal{I}$ for confirming the legitimacy of $\mathcal{D}_i$. The verification strategies may involves $\mathcal{D}_i$'s configuration, user's membership status or the accounting and billing (e.g., $\mathcal{V}_j$'s subscription fees accounted with user's mobile phone bill). With these strategies, $\mathcal{V}_j$ authorizes $\mathcal{I}$ to distribute service-related credentials to $\mathcal{D}_i$. When requesting $\mathcal{V}_j$ for a service, the user generates an anonymous signature based on the corresponding credential to attests his legitimacy with other necessary information, such as the integrity measurement values of executing applications on $\mathcal{D}_i$, the amounts of e-cash or the content of e-ticket. The specific information structure and content are particularly defined by the service protocol. $\mathcal{V}_j$ authenticates the user's request by verifying his signature without revealing his identity. In some scenarios, $\mathcal{I}$ and $\mathcal{V}$ could be one entity. Fig.1 illustrates the system model for our proposed scheme.

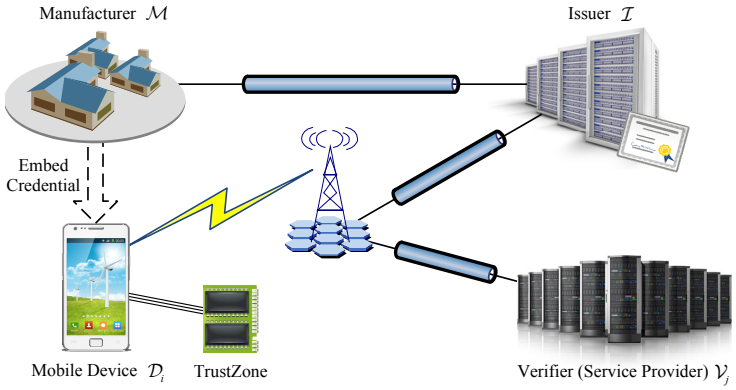

Fig. 1. System Model of DAA-TZ.

3.2 Assumptions and Threat Model

In the system model, there is a dedicated channel between $\mathcal{M}$ and $\mathcal{I}$, which could be either physical connection or other forms of out-of-band communication, for $\mathcal{I}$ issuing credentials via $\mathcal{M}$. To simplify our design, we assume that data communications between $\mathcal{M}$ and $\mathcal{I}$, and between $\mathcal{D}_i$ and $\mathcal{V}_j$ build on secure transport protocols, like TLS/SSL, which can provide confidentiality, authenticity and integrity protection. Note that the secure channel between $\mathcal{D}_i$ and $\mathcal{V}_j$ is only verifier-authentication (i.e., unilateral authentication) in case $\mathcal{D}_i$'s identity is revealed. Additionally, Public Key Infrastructure (PKI) is also assumed to realize authenticating $\mathcal{I}$. As a consequence, $\mathcal{M}$, $\mathcal{V}_j$ and $\mathcal{D}_i$ can accurately obtain public keys, public parameters and revocation list from $\mathcal{I}$ who displays the public information with certificate for being downloaded.

Actually, the establishment of the whole system requires some premised trust relationships. First, the cooperation is assumed to be credible between $\mathcal{I}$ and each $\mathcal{M}$ who ensures not to embed credentials to illegal $\mathcal{D}_i$ or the $\mathcal{D}_i$ without available TrustZone. Second, $\mathcal{V}_j$ trusts that, before issuing credentials, $\mathcal{I}$ always checks $\mathcal{D}_i$'s legitimacy by using verification strategies provided by $\mathcal{V}_j$. As a result, $\mathcal{V}_j$ would believe that the right credentials are in the right users' hands. Another accepted fact is that the user trusts $\mathcal{M}$ not to deliberately damage his $\mathcal{D}_i$'s security. Constrained by the market supervision and the force of law, the above-mentioned trust relationships are easily established and maintained.

Based on the assumptions, DAA-TZ protects against the following adversary:

- The adversary can attack the scheme itself by attempting to pretend entities, manipulate data transmission between entities and forge data.
- The adversary can perform any software-based attacks which compromise the mobile Rich OS or existing applications running in REE. DAA-TZ interfaces in REE are also available for the adversary.
- The adversary can physically access the mobile device. He can reboot the device and gain access to data residing on persistent storage.

However, we ignore the malicious behaviors of tampering with the TrustZone hardware or mounting side-channel attacks on PUF [19]. Moreover, pointed out by [30], since in general it is not possible to prevent simple relay attacks, we do not consider that an adversary just forwards a DAA signature from $\mathcal{D}_i$ to $\mathcal{V}_j$.

4 DAA-TZ Scheme for Mobile Device

In this section, we give the specific design for the architecture of trusted mobile device, and then present the key derivation and sensitive data management. Depending on these, DAA-TZ scheme are detailed afterwards.

4.1 The Architecture of Trusted Mobile Device

Leveraging TrustZone and PUF technology, we design the architecture of trusted mobile device specifically for DAA-TZ. The software-based implementation of

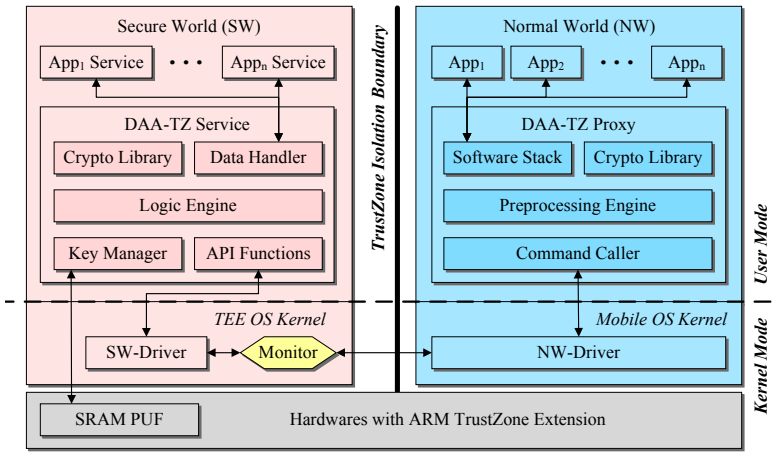

Fig. 2. Architecture of Trusted Mobile Device for DAA-TZ.

DAA-TZ functionality on existing hardwares targets at economy, flexibility and extensibility. Meanwhile, our architecture is designed to be compatible with the conventional running model of secure applications using TrustZone. Fig. 2 shows the detailed architecture with the way components interact with each other.

DAA-TZ functionality in the architecture contains two components: untrusted DAA-TZ Proxy in *normal world* (NW) and security-sensitive DAA-TZ Service in *secure world* (SW). In reality, SW instantiates TEE, while NW implements REE. DAA-TZ Service is isolated via TrustZone from all other codes running in NW. The components are formally described as follows.

DAA-TZ Proxy. It is the component visible for mobile applications in NW. Waiting for their DAA-TZ service requests, the proxy handles the parameters and preprocesses them. According to the request type, the proxy would call DAA-TZ Service for substantive computations of the scheme and finally return the results. DAA-TZ Proxy consists of the following four subcomponents:

- **Software Stack**: provides top DAA-TZ interfaces for mobile applications. It parses the service requests and gives back service response results.
- **Crypto Library**: offers cryptographic algorithm support for Preprocessing Engine. In NW, this library only supports exponentiations on elliptic curves.
- **Preprocessing Engine**: executes pre-computation for DAA-TZ when one of two conditions is satisfied: a new credential is generated successfully, or a pre-computed result is consumed correctly.
- **Command Caller**: formats calling command and interacts with DAA-TZ Service. It sends the command through the GP TEE Client API [22], requests to switch NW to SW via NW-Driver and waits for the returned values.

DAA-TZ Service. It is the core component to perform DAA-TZ critical computations and operations. The execution of the component codes is under the

well protection of TrustZone isolation mechanism. Five following subcomponents constitute DAA-TZ Service component:

- **API Functions**: receives a service request from DAA-TZ Proxy and parses the command. The functions transmit instructions to Logic Engine and waits for results that would be forwarded back to DAA-TZ Proxy.
- **Key Manager**: creates cryptographic keys using the unique root key seed extracted from SRAM PUF and provides keys to Data Handler.
- **Data Handler**: receives message to be signed from application service and seals or unseals sensitive data. To prevent adversary from forging message, Data Handler only receives message produced by application service in SW. Besides, using keys from Key Manager, Data Handler seals sensitive data to store them in the insecure persistent storage space of mobile device.
- **Crypto Library**: offers cryptographic algorithm support for Logic Engine and Data Handler. In SW, it supports bilinear maps, computations on elliptic curves, and other cryptographic operations.
- **Logic Engine**: executes the computations of security-sensitive parts of DAA-TZ scheme. Logic Engine reads necessary parameters and data to run operations relying on scheme specification.

Application and Application Service. The corresponding application should be launched if the user wants to enjoy a remote service from $\mathcal{V}_j$. The secure application released by $\mathcal{V}_j$ consists of two parts: App for NW and App Service for SW. App provides the GUI and basic functions. When App has the need to execute DAA-TZ procedures for remote service authenticating user's legitimacy with the message as service input, it calls DAA-TZ Proxy using its Software Stack. App could notify App Service in SW to prepare message to be signed through inter-domain communication mechanism supported by TrustZone [15]. App Service is trusted for processing security-sensitive data. It is sometimes non-existent if the remote service does not require secure computation or signed message as input.

Components in Kernels. SW-Driver in TEE OS Kernel and NW-Driver in Mobile OS Kernel handle the communication requests and responses with respect to switching the worlds. Implemented as Secure Monitor defined by TrustZone, the Monitor controls hardwares to fulfill the switching action.

Components in Hardwares. The hardware of mobile device support ARM TrustZone extension technology. Protected by TrustZone mechanism, SRAM PUF component is only accessible for SW.

4.2 Key Derivation and Sensitive Data Management

Prior to describing the concrete construction of our DAA scheme, we show how to derive various keys for different purposes using the root key seed extracted from SRAM PUF and how to utilize the derived keys to protect sensitive data.

Root Key Seed Extraction. We use the technique of SRAM PUF in [35] to extract the secret root key seed s, which is a unique bit string picked randomly

by $\mathcal{M}$ who "stores" it in $\mathcal{D}_i$ through the physical features of one SRAM inside $\mathcal{D}_i$. From SRAM PUF component, s is only reproduced and securely cached by Key Manager when $\mathcal{D}_i$ starts up every time in normal use. The confidentiality of s is rigidly guaranteed by TrustZone.

Key Derivation. Key Manager has the deterministic key derivation function $\mathsf{KDF} : \widetilde{\mathcal{S}} \times \widetilde{\mathcal{D}} \rightarrow \widetilde{\mathcal{K}}$, where $\widetilde{\mathcal{S}}$ is the key seed space, $\widetilde{\mathcal{D}}$ is a set of strings for statement of purposes with possible variables, and $\widetilde{\mathcal{K}}$ is the derived key space. Using the KDF, the device key pair which is analogous to the endorsement key pair in [27,28] and the storage root key for generating specific storage keys can be derived as $(dsk, dpk) \leftarrow \mathsf{KDF}_s(\texttt{`identity'})$ and $srk \leftarrow \mathsf{KDF}_s(\texttt{`storage\_root'})$ respectively, where s is the root key seed. Whereafter, we also use the storage keys derived from KDF with the storage root key srk to preserve sensitive data. The hierarchical structure of storage keys enhances the security for key usage. Note that all the derived keys are never stored permanently. Instead, they are regained via KDF with s at the same way when needed.

Sensitive Data Management. We can utilize the storage keys derived from the storage root key srk to seal the DAA-TZ's public parameters $params$, some mobile device $\mathcal{D}_i$'s credential $cred$ and a pair (f, T), where f is $\mathcal{D}_i$'s secret key and $T = g_1^f$ for some fixed basis g_1. The sealed results of these data can be stored in the insecure positions of NW.

- Protect integrity for $params$: $mk_{params} \leftarrow \mathsf{KDF}_{srk}(\texttt{`storage\_key'}, \texttt{`MAC'}, params, 0)$ and $blob_{params} \leftarrow \mathsf{Data\_Seal}(\texttt{`MAC'}, mk_{params}, params)$, where
$$blob_{params} := params||\mathsf{MAC}_{mk_{params}}(params).$$
- Protect integrity for $cred$: $mk_{cred} \leftarrow \mathsf{KDF}_{srk}(\texttt{`storage\_key'}, \texttt{`MAC'}, T, cred, 1)$ and $blob_{cred} \leftarrow \mathsf{Data\_Seal}(\texttt{`MAC'}, mk_{cred}, cred)$, where
$$blob_{cred} := cred||\mathsf{MAC}_{mk_{cred}}(cred).$$
- Protect confidentiality and integrity for (f, T): generate two kinds of keys by $(sk_f, mk_f) \leftarrow \mathsf{KDF}_{srk}(\texttt{`storage\_key'}, \texttt{`Enc+MAC'}, T, 2)$, and then $blob_f \leftarrow \mathsf{Data\_Seal}(\texttt{`Enc+MAC'}, sk_f, mk_f, f, T)$, where
$$blob_f := \mathsf{Enc}_{sk_f}(f)||T||\mathsf{MAC}_{mk_f}(\mathsf{Enc}_{sk_f}(f)||T).$$

Data Handler can use $\mathsf{Data\_Unseal}()$ to recover and verify the sensitive data from blobs with the related keys regained by Key Manager.

4.3 The Details of DAA-TZ Scheme

Some preliminary work needs to be done as premise to start the normal procedures of the scheme. Specifically, when $\mathcal{M}$ initializes $\mathcal{D}_i$ in the factory, $\mathcal{M}$ guides $\mathcal{D}_i$ in SW to use its root key seed to generate the unique device key (dsk, dpk) which could uniquely identity $\mathcal{D}_i$. Then, $\mathcal{M}$ issues a certificate $cert$ for the public key dpk to indicate $\mathcal{M}$'s recognition for $\mathcal{D}_i$. The certificate $cert$ contains the configuration information (e.g., whether TrustZone is available) of $\mathcal{D}_i$.

For $\mathcal{D}_i$-centered design, DAA-TZ scheme consists of seven phases: **Setup**, **KeyGen**, **Embed**, **Sign**, **Verify**, **Revoke** and **Rejoin**. First of all, **Setup** is executed to create the public parameters. After that, the issuer $\mathcal{I}$ can execute **KeyGen** to generate its public/private key pair according to the public parameters. In addition, **KeyGen** and **Embed** are compelled to execute sequentially before $\mathcal{D}_i$ leaves the factory. Then, other phases are able to execute correctly according to application requirements. We adopt the techniques in [2,10,32] to build DAA-TZ scheme. The phases of DAA-TZ scheme are presented in detail as follows.

Setup. Given a security parameter λ, picks the suitable bilinear groups parameters $\Lambda = (p, \mathbb{G}_1, \mathbb{G}_2, \mathbb{G}_T, e, g_1, g_2)$ described in Section 2.2 such that the bit-length of p is 2λ. In addition, chooses three independent collision-resistant hash functions:

$$\mathsf{H}_1 : \{0,1\}^* \rightarrow \mathbb{Z}_p,\ \mathsf{H}_2 : \{0,1\}^* \rightarrow \mathbb{G}_1,\ \mathsf{H}_3 : \{0,1\}^* \rightarrow \mathbb{Z}_p.$$

Finally, publish $(p, \mathbb{G}_1, \mathbb{G}_2, \mathbb{G}_T, e, g_1, g_2, \mathsf{H}_1, \mathsf{H}_2, \mathsf{H}_3)$ as the public parameters *params*. For each mobile device $\mathcal{D}_i$, $\mathcal{M}$ imports *params* to $\mathcal{D}_i$ and calls $\mathsf{Data\_Seal}()$ to seal the *params*. The resulting blob $blob_{params}$ is stored in $\mathcal{D}_i$.

KeyGen. This phase initializes the public/private key pair for the issuer $\mathcal{I}$ and generates a mobile device $\mathcal{D}_i$'s key.

- *Key Generation for Issuer.* Given *params* as input, $\mathcal{I}$ picks $x, y \leftarrow \mathbb{Z}_p^*$, and computes $X := g_2^x$ and $Y := g_2^y$. $\mathcal{I}$ sets (x, y) as the private key $sk_{\mathcal{I}}$ and publishes (X, Y) as the public key $pk_{\mathcal{I}}$. We assume that some mobile device $\mathcal{D}_i$ and verifier $\mathcal{V}_j$ could get the correct $pk_{\mathcal{I}}$ from $\mathcal{I}$ via verifying the certificate[3] for $pk_{\mathcal{I}}$. Besides, $\mathcal{I}$ initializes a revocation list RL as empty.
- *Key Generation for Mobile Device.* In SW of a mobile device $\mathcal{D}_i$, Logic Engine calls $\mathsf{DAATZ\_SW\_Create}()$ to generate key blob $blob_f$ that seals the $\mathcal{D}_i$'s secret key f and public information T. It runs as:

  $$blob_f \leftarrow \mathsf{DAATZ\_SW\_Create}(blob_{params}),$$

  where the API mainly has the following four operations:

  1) Unseal the blob $blob_{params}$ to get *params* by calling $\mathsf{Data\_Unseal}()$.
  2) Pick $f \leftarrow \mathbb{Z}_p^*$ and compute $T := g_1^f$.
  3) Call $\mathsf{Data\_Seal}()$ to seal the pair (f, T) to obtain the key blob $blob_f$.
  4) Output $blob_f$.

  $\mathcal{D}_i$ switches back to NW and stores $blob_f$ in its non-volatile memory.

Embed. In this phase, a credential *cred* for each mobile device $\mathcal{D}_i$ is produced and embedded into the device before $\mathcal{D}_i$ leaves the factory as follows.

1. $\mathcal{M}$ obtains T from the key blob $blob_f$ and sends T to $\mathcal{I}$ through the dedicated channel. Because of the cooperative relationship, $\mathcal{I}$ trusts $\mathcal{M}$ is asking for producing credential to embed into a $\mathcal{D}_i$ with legitimate configurations.

[3] Utilizing PKI solution, a Certificate Authority (CA) issues a public key certificate for $pk_{\mathcal{I}}$ to the issuer $\mathcal{I}$.

Thus, $\mathcal{I}$ does not check the validity of $\mathcal{D}_i$ any more. But in this phase, only the credentials that do not require other more strict verification strategies from $\mathcal{V}_j$ are permitted to issue.

2. On input of a group element T, $\mathcal{I}$ runs the following signature algorithm to generate a credential *cred* for T.

$$(A, B, C, D, c_{\mathcal{I}}, s_{\mathcal{I}}) \leftarrow \mathsf{SIG\_Cred}(params, sk_{\mathcal{I}}, T)$$

The signature algorithm has the following four steps:

1) Choose $a \leftarrow \mathbb{Z}_p^*$ and compute $A := g_1^a$, $B := g_1^{a \cdot y}$, $C := g_1^{a \cdot x} \cdot T^{a \cdot x \cdot y}$, $D := T^{a \cdot y}$ and $t := a \cdot y$.
2) Choose $r_{\mathcal{I}} \leftarrow \mathbb{Z}_p$ and compute $R_{\mathcal{I}1} := g_1^{r_{\mathcal{I}}}$, $R_{\mathcal{I}2} := T^{r_{\mathcal{I}}}$.
3) Compute $c_{\mathcal{I}} := \mathsf{H}_1(B||D||g_1||T||R_{\mathcal{I}1}||R_{\mathcal{I}2})$.
4) Compute $s_{\mathcal{I}} := r_{\mathcal{I}} + c_{\mathcal{I}} \cdot t \pmod{p}$.

Then, $\mathcal{I}$ sends $(A, B, C, D, c_{\mathcal{I}}, s_{\mathcal{I}})$ to $\mathcal{M}$.

3. $\mathcal{M}$ imports the tuple $(A, B, C, D, c_{\mathcal{I}}, s_{\mathcal{I}})$ into SW of $\mathcal{D}_i$. Logic Engine calls the API $\mathsf{DAATZ\_SW\_Join}()$ to check the elements from $\mathcal{I}$ and generate a credential blob $blob_{cred}$ to store:

$$blob_{cred} \leftarrow \mathsf{DAATZ\_SW\_Join}(blob_{params}, pk_{\mathcal{I}}, T, A, B, C, D, c_{\mathcal{I}}, s_{\mathcal{I}}),$$

where the API executes the following operations:

1) Call $\mathsf{Data\_Unseal}()$ to unseal the blob $blob_{params}$ to obtain $params$.
2) Compute $R'_{\mathcal{I}1} := g_1^{s_{\mathcal{I}}} \cdot B^{-c_{\mathcal{I}}}$ and $R'_{\mathcal{I}2} := T^{s_{\mathcal{I}}} \cdot D^{-c_{\mathcal{I}}}$.
3) Compute $c'_{\mathcal{I}} := \mathsf{H}_1(B||D||g_1||T||R'_{\mathcal{I}1}||R'_{\mathcal{I}2})$.
4) Check whether the relations $A \neq 1_{\mathbb{G}_1}$, $e(A, Y) = e(B, g_2)$, $e(C, g_2) = e(A \cdot D, X)$ and $c_{\mathcal{I}} = c'_{\mathcal{I}}$ hold.
5) If all the relations hold, set $cred := (A, B, C, D)$.
6) Call $\mathsf{Data\_Seal}()$ to seal *cred*, and output the resulting blob $blob_{cred}$.

Note that $\mathcal{M}$ could send simultaneously a set of $\{T_i\}_{i=1}^n$ to $\mathcal{I}$ by a single interaction, and obtain the corresponding credential set $\{cred_i\}_{i=1}^n$.

4. After a credential *cred* is successfully obtained by $\mathcal{D}_i$, TrustZone switches to NW and DAA-TZ Proxy executes pre-computation in the background to prepare for user's fast anonymous signing operation in the following **Sign** phase. Preprocessing Engine calls $\mathsf{DAATZ\_NW\_PreCmpt}()$ to generate a blinded credential:

$$(l, S, U, V, W) \leftarrow \mathsf{DAATZ\_NW\_PreCmpt}(blob_{params}, blob_{cred}),$$

where the algorithm consists of the following steps.

1) Get the prime p and credential *cred* by directly reading the plaintext part of $blob_{params}$ and $blob_{cred}$ respectively.
2) Parse *cred* as (A, B, C, D).
3) Choose $l \leftarrow \mathbb{Z}_p^*$ and compute $(S, U, V, W) := (A^l, B^l, C^l, D^l)$.
4) Output (l, S, U, V, W).

Preprocessing Engine stores the output (l, S, U, V, W). Now that all DAA-TZ related operations in $\mathcal{M}$ have been done, $\mathcal{D}_i$ is going to be delivered to the hand of a user.

Sign. This phase enables a user to anonymously attest the legitimacy of both his status and his mobile device.

1. App of a mobile device $\mathcal{D}_i$ connects a remote verifier $\mathcal{V}_j$. Then, $\mathcal{V}_j$ negotiates with $\mathcal{D}_i$ to decide a basename $bsn \in \{0,1\}^*$. If bsn is empty (i.e., $bsn = \perp$), it means that the signatures created by $\mathcal{D}_i$ are unlinkable. If bsn is a non-empty basename (i.e., $bsn \neq \perp$), the signatures produced by $\mathcal{D}_i$ are pseudonymous, i.e., signatures under the same basename could be linked and signatures under different basenames are unlinkable. In addition, $\mathcal{V}_j$ chooses a nonce $n_{\mathcal{V}_j} \leftarrow \{0,1\}^{2\lambda}$ and sends it to $\mathcal{D}_i$.
2. In NW, it is optional for App to notify App Service to prepare authentication information as message m to be signed. The notification is combined with the command for DAA-TZ Proxy to request generating signature.
3. On account of the request, the environment is switched into SW. App Service may need its user to securely input some information, such as password, to generate m. Then, Logic Engine calls DAATZ_SW_Sign() to create a DAA signature σ using the related pre-computation result as:

   $$\sigma \leftarrow \mathsf{DAATZ\_SW\_Sign}(blob_{params}, blob_f, blob_{cred}, bsn, n_{\mathcal{V}_j}, m, l, S, U, V, W),$$

   where the detailed process is presented as follows:

   1) Unseal the blobs to get $params$, f, T and $cred$ by calling Data_Unseal().
   2) If $bsn \neq \perp$, compute $J := \mathsf{H}_2(bsn)$ and $K := J^f$, else set $J, K := 1_{\mathbb{G}_1}$.
   3) Choose $r_{\mathcal{D}_i} \leftarrow \mathbb{Z}_p$ and compute $R_{\mathcal{D}_i 1} := J^{r_{\mathcal{D}_i}}$ and $R_{\mathcal{D}_i 2} := B^{l \cdot r_{\mathcal{D}_i}}$.
   4) Compute $c_{\mathcal{D}_i} := \mathsf{H}_3(J||K||S||U||V||W||R_{\mathcal{D}_i 1}||R_{\mathcal{D}_i 2}||bsn||n_{\mathcal{V}_j}||m)$.
   5) Compute $s_{\mathcal{D}_i} := r_{\mathcal{D}_i} + c_{\mathcal{D}_i} \cdot f \pmod p$.
   6) Output a signature $\sigma := (K, S, U, V, W, c_{\mathcal{D}_i}, s_{\mathcal{D}_i})$.

   After σ is generated successfully, the environment is switched back to NW with the returned outputs m and σ that are eventually sent to $\mathcal{V}_j$.
4. Through the above step, a pre-computation result (l, S, U, V, W) is consumed. Preprocessing Engine deletes the previous pre-computation result, then calls DAATZ_NW_PreCmpt() again to get a new pre-computation tuple (l', S', U', V', W') for the next use. This pre-computation process is executed parallelly in the background of NW without causing obvious time delays felt by the user.

Verify. In this phase, a verifier $\mathcal{V}_j$ checks whether or not a signature σ on a message m is valid.

1. $\mathcal{V}_j$ verifies (m, σ) by the means of calling verification algorithm Verify() as:

   $$res \leftarrow \mathsf{Verify}(params, pk_{\mathcal{I}}, bsn, n_{\mathcal{V}_j}, m, \sigma),$$

   where the algorithm runs in detail as follows:

   1) Parse σ as $(K, S, U, V, W, c_{\mathcal{D}_i}, s_{\mathcal{D}_i})$.
   2) For each $f_{rvk} \in RL$, if $W = U^{f_{rvk}}$, then set $res := false$ and abort.
   3) If $bsn \neq \perp$, compute $J' := \mathsf{H}_2(bsn)$, else set $J' := 1_{\mathbb{G}_1}$.
   4) Compute $R'_{\mathcal{D}_i 1} := J'^{s_{\mathcal{D}_i}} \cdot K^{-c_{\mathcal{D}_i}}$ and $R'_{\mathcal{D}_i 2} := U^{s_{\mathcal{D}_i}} \cdot W^{-c_{\mathcal{D}_i}}$.
   5) Compute $c'_{\mathcal{D}_i} := \mathsf{H}_3(J'||K||S||U||V||W||R'_{\mathcal{D}_i 1}||R'_{\mathcal{D}_i 2}||bsn||n_{\mathcal{V}_j}||m)$.

6) Check whether the relations $S \neq 1_{\mathbb{G}_1}$, $e(S,Y) = e(U,g_2)$, $e(V,g_2) = e(S \cdot W, X)$ and $c_{\mathcal{D}_i} = c'_{\mathcal{D}_i}$ hold.
7) If all the relations hold, then $res := true$, else $res := false$.

According to the verification result res, $\mathcal{V}_j$ decides whether to accept the message/signature pair (m, σ) and provide service for the owner of $\mathcal{D}_i$.

Revoke. In this phase, if a secrete key f has been revealed, it could be revoked.

1. $\mathcal{I}$ has the duty to censor the Internet. By collecting and analyzing daily logs, $\mathcal{I}$ tries to find the clue indicating the leakage of f and its declared valid credential $cred$. If the pair $(f, cred)$ is found, $\mathcal{I}$ checks whether $cred$ is a valid credential on the f with $params$ and $pk_{\mathcal{I}}$. For valid one, $\mathcal{I}$ adds f into RL, i.e., $RL := RL \cup \{f\}$.
2. $\mathcal{V}_j$ maintains locally a revocation list RL which is regularly updated from the revocation list published by $\mathcal{I}$.

Rejoin. This phase enables $\mathcal{D}_i$ to recreate a new secret key f' and obtain a related new credential $cred'$, when an old key f of $\mathcal{D}_i$ has already been revoked, or $\mathcal{D}_i$ intends to apply for a membership credential under a new public key created by either a new or an existing issuer[4]. We choose an IND-CCA secure public key encryption scheme $(\mathsf{ENC}, \mathsf{DEC})$ for the following usage with the device key pair (dpk, dsk). In particular, $C \leftarrow \mathsf{ENC}_{dpk}(M)$ and $M \leftarrow \mathsf{DEC}_{dsk}(C)$ denote that encrypting the message M with the public key dpk to obtain a ciphertext C, and decrypting the ciphertext C with the secret key dsk to recover a message M respectively. This phase is executed as follows.

1. DAA-TZ Proxy sends $\mathcal{D}_i$'s dpk and its certificate $cert$ to $\mathcal{I}$.
2. $\mathcal{I}$ checks whether dpk is valid with $cert$ for confirming $\mathcal{D}_i$'s configurations. Utilizing the verification policy from $\mathcal{V}_j$, $\mathcal{I}$ checks other properties of $\mathcal{D}_i$ and its owner. If all the checks are passed, $\mathcal{I}$ randomly chooses a key k for MAC operation, and a nonce $n_{\mathcal{I}} \leftarrow \{0,1\}^{2\lambda}$. $\mathcal{I}$ then adds $n_{\mathcal{I}}$ into a nonce list NL which is initially empty, i.e., $NL := NL \cup \{n_{\mathcal{I}}\}$. Next, $\mathcal{I}$ generates a commitment request by encrypting k and $n_{\mathcal{I}}$ with dpk, i.e., $comm_{req} \leftarrow \mathsf{ENC}_{dpk}(k||n_{\mathcal{I}})$. Finally, $\mathcal{I}$ sends $comm_{req}$ to $\mathcal{D}_i$.
3. DAA-TZ Proxy invokes DAA-TZ Service with inputting $comm_{req}$. In SW, $\mathsf{DAATZ\_SW\_Commit}()$ is called to generate a commitment response:

$$comm_{res} \leftarrow \mathsf{DAATZ\_SW\_Commit}(comm_{req}),$$

where the process is executed as follows:

1) Recover the device private key dsk using the root key seed s, i.e., $(dsk, dpk) \leftarrow \mathsf{KDF}_s(\texttt{'identity'})$.
2) Use the secret key dsk to decrypt $comm_{req}$: $k||n_{\mathcal{I}} \leftarrow \mathsf{DEC}_{dsk}(comm_{req})$.
3) Call $\mathsf{DAATZ\_SW\_Create}()$ to generate a blob $blob_{f'}$ associated with a new secret key f' and the corresponding $T' = g_1^{f'}$.
4) Compute $\tau \leftarrow \mathsf{MAC}_k(T'||n_{\mathcal{I}})$ and $comm_{res} := (\tau, n_{\mathcal{I}}, T')$.

[4] If the public key of an existing issuer has expired, it should refresh its public key by creating a new one and obtaining the corresponding certificate.

Finally, $\mathcal{D}_i$ switches back to NW and sends $comm_{res}$ to $\mathcal{I}$.

4. $\mathcal{I}$ first verifies whether $comm_{res}$ is valid by computing $\tau' \leftarrow \mathsf{MAC}_k(T'||n_{\mathcal{I}})$ and checking if the relations $n_{\mathcal{I}} \in NL$ and $\tau = \tau'$ hold. If hold, as in the **Embed** phase, $\mathcal{I}$ runs the $\mathsf{SIG\_Cred}()$ algorithm with the public parameters $params$, its secret key $sk'_{\mathcal{I}}$ and the T' contained in $comm_{res}$ as the input to obtain a tuple $comm_{cred} := (A', B', C', D', c'_{\mathcal{I}}, s'_{\mathcal{I}})$, i.e.,
$$(A', B', C', D', c'_{\mathcal{I}}, s'_{\mathcal{I}}) \leftarrow \mathsf{SIG\_Cred}(params, sk'_{\mathcal{I}}, T').$$
Finally, $\mathcal{I}$ sends the above tuple $comm_{cred}$ to $\mathcal{D}_i$.
5. As in the **Embed** phase, DAA-TZ Service in SW calls $\mathsf{DAATZ\_SW\_Join}()$ with the tuple $comm_{cred}$ to generate a credential blob $blob'_{cred}$, i.e.,
$$blob'_{cred} \leftarrow \mathsf{DAATZ\_SW\_Join}(blob_{params}, pk'_{\mathcal{I}}, T', A', B', C', D', c'_{\mathcal{I}}, s'_{\mathcal{I}}),$$
where $pk'_{\mathcal{I}}$ is the public key of $\mathcal{I}$.
6. Back to NW, Preprocessing Engine invokes $\mathsf{DAATZ\_NW\_PreCmpt}()$ to obtain a pre-computation tuple, i.e.,
$$(l', S', U', V', W') \leftarrow \mathsf{DAATZ\_NW\_PreCmpt}(blob_{params}, blob'_{cred}).$$

5 Security Analysis

In this section, we first review the desired security properties of DAA schemes for mobile devices, then analyze DAA-TZ satisfying these properties.

Security Properties. Informally, a DAA scheme for mobile devices should satisfy the following properties:

Anonymity. The anonymity property requires that anyone including the issuer cannot identify the signer of a DAA signature. In particular, we consider two types of anonymity as follows.

- *Forward Anonymity:* Given a DAA signature for $bsn = \perp$, anyone cannot decide whether the signature was produced previously by some mobile device even though it compromises the NW of the device later.
- *Pseudonymity:* Given two DAA signatures for different non-empty basenames, anyone cannot determine whether they are created by the same mobile device. Note that in this case, if the SW of a mobile device remains honest, its NW could not be corrupted by an adversary.

Traceability. No adversary can create a DAA signature which cannot be traced to a secret key originated from an execution of the **Embed** or **Rejoin** phase.

Non-frameability. For a target mobile device (picked adaptively by an adversary), the adversary cannot forge a DAA signature which could be traced to the secret key of the target device even though it compromises the NW of the target device.

Security Analysis. In the following security analysis, we assume that the underlying primitives (symmetric encryption, MAC and public key encryption) are secure in the DAA-TZ scheme. Moreover, for the sake of simplifying analysis, we

assume that the authenticated channel between the SW of a mobile device and an issuer, which is established by the "Encrypt-then-MAC" method with the device key pair (dpk, dsk), is secure. Actually, the method was adopted in the DAA schemes [3,10]. Besides, for simplicity, the nonce $n_{\mathcal{V}_j}$ from the verifier $\mathcal{V}_j$ is considered as a part of the message m. Because it is beyond the scope of this paper to provide a formal security proof, we here only give an informal security analysis to demonstrate that the proposed DAA-TZ scheme satisfies the above security properties. Nevertheless, according to the security proof of the forward anonymity property in [32], one can analogously prove that the DAA-TZ scheme is forward anonymous under the $\text{DDH}_{\mathbb{G}_1}$ assumption. Furthermore, according to the security proof in [2], one can analogously prove the DAA-TZ scheme to be pseudonymous under the $\text{DDH}_{\mathbb{G}_1}$ assumption, traceable under the B-4-LRSW assumption, and non-frameable under the discrete logarithm (DL) assumption. Now, we argue why the DAA-TZ scheme satisfies the above security properties.

Forward Anonymity. Given a DAA signature $\sigma = (K, S, U, V, W, c, s)$ for a pair $(bsn = \perp, m)$ and a target mobile device $\mathcal{D}_i$, an adversary $\mathcal{A}$ attempts to determine whether σ was signed previously by $\mathcal{D}_i$. First of all, recalling $K = 1_{\mathbb{G}_1}$, (c, s) is perfectly zero-knowledge in the random oracle model [1]. Thus, the tuple (S, U, V, W) is the only information that could be used by $\mathcal{A}$ to decide if $\mathcal{D}_i$ is the signer of σ. When $\mathcal{A}$ corrupts the NW of $\mathcal{D}_i$, it could have access to the API $\mathsf{DAATZ\_SW\_Sign}()$ and obtain the group element T, the $\mathcal{D}_i$'s credential $cred = (A, B, C, D)$ and the pre-computation result (l', S', U', V', W') for next signing operation. Note that the group element B being input to $\mathsf{DAATZ\_SW\_Sign}()$ is fixed as the integrity of the credential (A, B, C, D) is protected by the MAC scheme. Since the pre-computation result being input to $\mathsf{DAATZ\_SW\_Sign}()$ is controlled by $\mathcal{A}$, the pseudonyms $\{K_i = \mathsf{H}_2(bsn_i)^f\}_{i=1}^n$ included in the outputting signatures are only useful information for $\mathcal{A}$, where n is the number of the $\mathsf{DAATZ\_SW\_Sign}()$ requested by $\mathcal{A}$, bsn_i is picked by $\mathcal{A}$ for each $i \in \{1, \ldots, n\}$, and f is the secret key of $\mathcal{D}_i$. Furthermore, $\mathcal{A}$ could also obtain some previous signatures from $\mathcal{D}_i$ and signatures from other mobile devices. Under the $\text{DDH}_{\mathbb{G}_1}$ assumption, $\mathcal{A}$ cannot link the tuple (S, U, V, W) to any information obtained via the above manners. Thereby, the forward anonymity property of DAA-TZ is guaranteed.

Pseudonymity. Given two DAA signatures $\sigma_0 = (K_0, S_0, U_0, V_0, W_0, c_0, s_0)$ for $(bsn_0 \neq \perp, m_0)$ and $\sigma_1 = (K_1, S_1, U_1, V_1, W_1, c_1, s_1)$ for $(bsn_1 \neq \perp, m_1)$ such that $bsn_0 \neq bsn_1$, an adversary $\mathcal{A}$ attempts to decide whether the two signatures were created by the same mobile device. According to the above security analysis, under the $\text{DDH}_{\mathbb{G}_1}$ assumption, $(S_0, U_0, V_0, W_0, c_0, s_0)$ and $(S_1, U_1, V_1, W_1, c_1, s_1)$ do not reveal any information that may be used by $\mathcal{A}$ to link σ_0 and σ_1. Thus, K_0 and K_1 are the only data that might help $\mathcal{A}$ link the above two signatures. For its purpose, $\mathcal{A}$ could obtain some DAA signatures from some mobile devices, and some group elements $\{T_i\}_{i=1}^n$ as well as the corresponding credentials $\{cred_i\}_{i=1}^n$ from the **Rejoin** phase, where n is the number of the executions of the **Rejoin** phase. Under the $\text{DDH}_{\mathbb{G}_1}$ assumption, all the information obtained by $\mathcal{A}$ could

not help it to link K_0 and K_1. Thus, the pseudonymity property of DAA-TZ scheme is guaranteed.

Traceability. An adversary $\mathcal{A}$ against the traceability property attempts to forge a DAA signature $\sigma = (K, S, U, V, W, c, s)$ on a basename/message pair (bsn, m), where the secret key f associated with σ cannot generate any group element T from an execution of the **Embed** phase or the **Rejoin** phase. Since the non-interactive zero-knowledge proof (c, s) is sound, it implies that $\mathcal{A}$ holds a valid credential $cred = (A, B, C, D)$ under the issuer $\mathcal{I}$'s public key $pk_{\mathcal{I}}$ and its secret key f which is different from all the secret keys from the **Embed** phase or the **Rejoin** phase. In addition, since the non-interactive proof $(c'_{\mathcal{I}}, s'_{\mathcal{I}})$ created by $\mathcal{I}$ in an execution of the **Rejoin** phase is perfectly zero-knowledge, the proof $(c'_{\mathcal{I}}, s'_{\mathcal{I}})$ does not reveal any information about $\mathcal{I}$'s secret key $sk_{\mathcal{I}}$. Thus, if the B-4-LRSW assumption holds, it is impossible for $\mathcal{A}$ to hold the valid credential $cred$ of the new secret key f. Hence, the traceability property of DAA-TZ is guaranteed.

Non-frameability. An adversary $\mathcal{A}$ against the non-frameability property attempts to forge a DAA signature $\sigma = (K, S, U, V, W, c, s)$ on a basename/message pair (bsn, m), which is associated with an honest mobile device $\mathcal{D}_i$'s secret key f. If $\mathcal{A}$ corrupts the NW of $\mathcal{D}_i$, it is able to have access to the DAATZ_SW_Sign() API and obtain $T = g_1^f$, the credential $cred = (A, B, C, D)$ on the f and the pre-computation result (l', S', U', V', W'). For the i-th DAATZ_SW_Sign() request issued by $\mathcal{A}$, it could get a signature $(K_i, S_i, U_i, V_i, W_i, c_i, s_i)$ on a message m_i where (S_i, U_i, V_i, W_i) is picked by $\mathcal{A}$ and $K_i = \mathsf{H}_2(bsn_i)^f$ if $bsn_i \neq \perp$ chosen by $\mathcal{A}$ or $K_i = 1_{\mathbb{G}_1}$ otherwise. Under the DL assumption, $\mathcal{A}$ cannot obtain the secret key f from T, $cred$ and $\mathsf{H}_2(bsn_i)^f$. Besides, (c_i, s_i) is a signature of knowledge [7]

$$SPK\{(f) : W_i = U_i^f \wedge K_i = \mathsf{H}_2(bsn_i)^{f}{}_{\text{ if } bsn_i \neq \perp}\}(bsn_i, m_i)^5$$

Thus, providing the DL assumption is true, it is infeasible to forge a signature of knowledge (c, s) on a new (bsn, m) under the secret key f. Therefore, $\mathcal{A}$ cannot forge a DAA signature $\sigma = (K, S, U, V, W, c, s)$ on (bsn, m). Consequently, the non-frameability property of DAA-TZ scheme is guaranteed.

Remark. For any pair (bsn, m) and its signature $\sigma = (K, S, U, V, W, c, s)$ on which the Verify() algorithm outputs *true*, the discrete logarithm $f = \log_U^W$ is the same as the secret key involved in V (i.e., $V = S^{x+fxy}$) as the bilinear property of the pairing e. Thus, no adversary could utilize a secret key in the revocation list RL to produce a DAA signature which is accepted by a verifier $\mathcal{V}_j$.

[5] Following the notation [6], $SPK\{(f) : W_i = U_i^f \wedge K_i = \mathsf{H}_2(bsn_i)^{f}{}_{\text{ if } bsn_i \neq \perp}\}(bsn_i, m_i)$ denotes the signature of knowledge on the message (bsn_i, m_i) that proves knowledge of f such that $W_i = U_i^f$ and $K_i = \mathsf{H}_2(bsn_i)^f$.

6 Implementation and Evaluation

In this section, we first present the prototype system of DAA-TZ from both aspects of hardware and software. Afterwards, we show the comparison of the proposed scheme to other four solutions. Finally, we give the performance evaluation and analysis based on our prototype system.

6.1 Implementation

Hardware Platform. To simulate real environment, we implement the role of manufacturer on one PC platform. Issuer and verifier are together implemented on another PC platform. For simulating mobile device, we leverage a development board Zynq-7000 AP Soc Evaluation Kit [29] to implement functions of DAA-TZ. It is TrustZone-enabled and equipped with ARM Cortex-A9 MPCore, 1GB DDR3 memory and On-Chip Memory (OCM) module including 256 KB SRAM. However, this SRAM is initialized by BootROM once the board is powered on, which prevents us from reading its initial data. Then we utilize an SRAM chip that is the type IS61LV6416-10TL [23] to act as our SRAM PUF. SRAM initial data is transferred to the Zynq development board by an FPGA implementation of Universal Asynchronous Receiver/Transmitter (UART) in Verilog hardware description language. A UART receiver in the Zynq board receives and stores the SRAM data in a RAM cache. Then the processor can fetch the SRAM data in the RAM cache via the bus.

Software Implementation. The software implementation on the development board for mobile device is divided into two parts. In secure world, we use Open Virtualization SierraTEE as the basic TEE OS which is compliant with GP's TEE Specifications [22]. For Key Manager of DAA-TZ Service, the fuzzy extractor of PUF is constructed on an open source BCH code [18]. For Crypto Library, we use OpenSSL-0.9.8y for general cryptographic algorithms, and Pairing-Based Cryptography (PBC) 0.5.14 library for computations of elliptic curves and pairings. We choose SHA256 for $\mathsf{H}_2()$, 1280-bit RSA key pair for the device key and 128-bit AES encryption with HMAC-SHA256 for sealing and unsealing operations. 2315 lines of code (LOC) in C language totally make up our components and auxiliary functions in secure world. In normal world, we run a Linux as REE OS with kernel 3.8.6. The SieraTEE project provides the Linux with NW-Driver and GP TEE Client API. , DAA-TZ Proxy totally comprises 1651 LOC. Besides we program one test application that could request executing DAA-TZ scheme. It contains 1068 LOC running in NW. In addition, there are several tens of thousands of LOC for implementing manufacturer, issuer and verifier.

6.2 Comparison

Dedicated for TrustZone security extension, DAA-TZ achieves some valuable and meaningful properties. In Table 1 we show how our scheme compares to existing solutions from the latest literature. These are ISC10 [30], BCL08 in LAMS [33],

Table 1. Comparison of DAA-TZ to other related solutions

	Computation Amount of On-line Signing		Switch Times of Signing	Pre-Compute	Controllable Pseudonym	Forward Anonymity	Data Management
	$bsn = \perp$	$bsn \neq \perp$					
ISC10	$4E_{\mathbb{G}_1}$	/	$\geqslant 1$	✘	✘	✘	✘
BCL08 (LAMS)	$3E_{\mathbb{G}_1}+4E_{\mathbb{G}_T}$ $+3P$	/	2	✘	✘	✘	✘
BL10 (Mdaak)	$4E_{\mathbb{G}_1}+E^2_{\mathbb{G}_1}+E_{\mathbb{G}_T}+P$		2	✘	✔	✘	✔
CPS10 (Mdaak)	$7E_{\mathbb{G}_1}$		1	✘	✔	✘	✔
DAA-TZ	$E_{\mathbb{G}_1}$	$3E_{\mathbb{G}_1}$	1	✔	✔	✔	✔

BL10 and CPS10 in Mdaak [34]. Likewise building on ECC-DAA and TrustZone technology, these solutions are all designed for mobile platform. The comparison focuses on the items related to mobile device, involving the most time-consuming computation amount of on-line anonymous signing (that leads user to wait for its instant result) depending on whether bsn is null, the switch times between SW and NW when executing signing, and other properties including realized pre-computation, user-controlled pseudonym, forward anonymity and sensitive data management using a root of trust. The notations used in the table are as follows: when we write $E_{\mathbb{G}_1}$ or $E_{\mathbb{G}_T}$, we mean the exponentiation is in group $\mathbb{G}_1$ or $\mathbb{G}_T$; $E^2_{\mathbb{G}_1}$ denotes 2 simultaneous exponentiations on $\mathbb{G}_1$, i.e. computing $a_1^{b_1} \cdot a_2^{b_2}$; P is short for pairing computation; the check mark (✔) denotes the solution considers or has this property while the X mark (✘) denotes not; the slash (/) indicates it does not support pseudonym.

From the table, it is clear that the proposed scheme not only has the least on-line computation amount and switch times for mobile devices in signing phase, but also provides all other listed properties. The signing phase is the most frequent execution and deeply influences the whole scheme's efficiency. Overall, the combination of these advantages endows DAA-TZ with a good user experience as well as high security.

6.3 Performance Evaluation

We measure the performance of DAA-TZ on the prototype system revolving around mobile device through a series of experiments with different parameters. Referring to ISO/IEC 15946-5 standard [24], we select two kinds of elliptic curves that are suitable for realizing Type-3 pairings. These curves are MNT curve with embedding degree 6 and BN curve with embedding degree 12. For testing various security levels of curves, we totally conduct 6 experiments respectively using MNT160, MNT224, BN160, BN192, BN224 and BN256, where each number denotes the approximate number of bits to optimally represent an element of the group $\mathbb{G}_1$. More precisely, MNT160 and BN160 provide 80-bit security level; MNT224 and BN192 provide 96-bit; BN224 and BN256 respectively provide 112-bit and 128-bit. Our experiments simulate the whole DAA-TZ running process

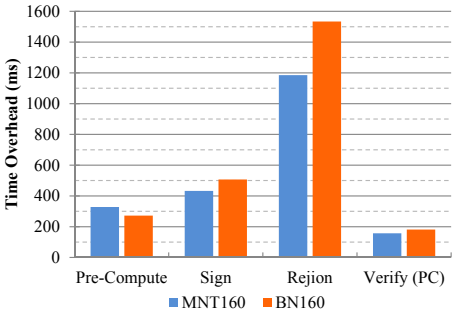

Fig. 3. Time overheads of the critical processes with 80-bit security level.

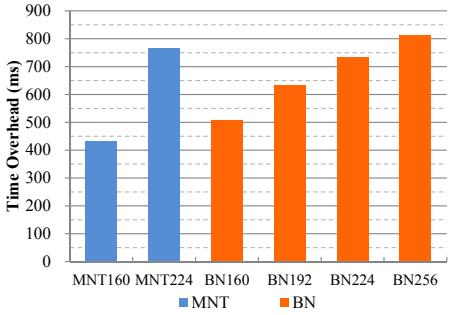

Fig. 4. On-line time overheads of **Sign** phase with different security levels.

covering instantiated protocol, TrustZone switch and sensitive data management. Each average experimental result is taken over 20 test-runs.

On MNT and BN curves with 80-bit security level, Fig.3 illustrates the average time overheads of critical processes including the computations of pre-compute, **Sign** (excluding pre-computation) and **Rejoin** on mobile device and **Verify** on PC for verifier. The results indicate that using both curves the frequent computations about either pre-compute or **Sign** only take less than 500 milliseconds (ms), while infrequent and time-consuming **Rejoin** spends less than 1550 ms. Even if the computation amount of **Verify** is quite large, the time overhead is indeed low on PC platform.

Fig.4 shows the average on-line time overheads of single **Sign** phase (excluding pre-computation) on mobile device using two curves with different security levels. From the figure, we can see that as the security levels increase, the time overheads of **Sign** phase have evident growth. The growth rate of overhead using MNT is higher than that using BN. Encouragingly, all the resulting overheads spend less than 820ms, which is completely acceptable for a mobile user.

Actually, a direct performance comparison of our competitive scheme to others is difficult because of four inevitable differences: the hardware platforms, the algorithm libraries, the selections of elliptic curves, and the completeness degrees of programming. Anyhow, according to our comparison and experimental results, DAA-TZ can be considered as a reasonably efficient scheme for mobile device. In regard to adopting modern mobile devices that are much more powerful than our development board and a more optimal library to implement elliptic curves and parings, the time overhead of our scheme could be further decreased.

7 Conclusion

In this paper, we propose DAA-TZ, a complete and efficient DAA scheme using TrustZone, to deal with the security and privacy issues specially for mobile users. DAA-TZ enables manufacture to embed credentials into devices and guarantees the minimal switch times of TrustZone during signing phase. Pre-computation is also carefully taken into consideration to raise scheme's efficiency. The root of

trust provided by SRAM PUF, key derivation and sensitive data management collectively enhance the security of our scheme. The implementation and evaluation convince that DAA-TZ is quite practical for mobile users. Our next step is to design the concrete secure applications based on DAA-TZ.

Acknowledgment. We thank Shijun Zhao and the anonymous reviewers for their valuable comments. This work was supported in part by grants from the National Natural Science Foundation of China (No.91118006, No.61202414 and No.61402455) and the National 973 Program of China (No.2013CB338003).

References

1. Bellare, M., Rogaway, P.: Random oracles are practical: A paradigm for designing efficient protocols. In: Proceedings of the 1st ACM Conference on Computer and Communications Security – CCS'93. pp. 62–73. ACM Press (1993)
2. Bernhard, D., Fuchsbauer, G., Ghadafi, E., Smart, N.P., Warinschi, B.: Anonymous attestation with user-controlled linkability. International Journal of Information Security 12(3), 219–249 (2013)
3. Brickell, E., Camenisch, J., Chen, L.: Direct anonymous attestation. In: Proceedings of the 11th ACM conference on Computer and communications security. pp. 132–145. ACM (2004)
4. Brickell, E., Chen, L., Li, J.: A new direct anonymous attestation scheme from bilinear maps. In: Trusted Computing-Challenges and Applications, pp. 166–178. Springer (2008)
5. Brickell, E., Li, J.: A pairing-based daa scheme further reducing tpm resources. In: Trust and Trustworthy Computing, pp. 181–195. Springer (2010)
6. Camenisch, J., Stadler, M.: Efficient group signature schemes for large groups. In: Kaliski, B. (ed.) Advances in Cryptology – CRYPTO'97. LNCS, vol. 1296, p. 410C424. Springer-Verlag (1997)
7. Chase, M., Lysyanskaya, A.: On signatures of knowledge. In: Dwork, C. (ed.) Advances in Cryptology – CRYPTO'06. LNCS, vol. 4117, p. 78C96. Springer-Verlag (2006)
8. Chen, L.: A daa scheme requiring less tpm resources. In: Information Security and Cryptology. pp. 350–365. Springer (2010)
9. Chen, L., Li, J.: Flexible and scalable digital signatures in tpm 2.0. In: Proceedings of the 2013 ACM SIGSAC conference on Computer & communications security. pp. 37–48. ACM (2013)
10. Chen, L., Page, D., Smart, N.P.: On the design and implementation of an efficient daa scheme. In: Smart Card Research and Advanced Application, pp. 223–237. Springer (2010)
11. Chen, X., Feng, D.: Direct anonymous attestation for next generation tpm. Journal of Computers 3(12), 43–50 (2008)
12. Commission, F.T., et al.: Mobile privacy disclosures: Building trust through transparency. Federal Trade Commission Staff Report (2013)
13. Galbraith, S., Paterson, K., Smart, N.: Pairings for cryptographers. Discrete Applied Mathematics 156(16), 3113–3121 (2008)
14. Guajardo, J., Kumar, S.S., Schrijen, G.J., Tuyls, P.: FPGA intrinsic PUFs and their use for IP protection. Springer (2007)

15. Jang, J., Kong, S., Kim, M., Kim, D., Kang, B.B.: Secret: Secure channel between rich execution environment and trusted execution environment. NDSS 2015 (2015)
16. Lysyanskaya, A., Rivest, R.L., Sahai, A., Wolf, S.: Pseudonym systems. In: Selected Areas in Cryptography. pp. 184–199. Springer (2000)
17. Maganis, G., Shi, E., Chen, H., Song, D.: Opaak: using mobile phones to limit anonymous identities online. In: Proceedings of the 10th international conference on Mobile systems, applications, and services. pp. 295–308. ACM (2012)
18. Morelos-Zaragoza, R.: Encoder/decoder for binary bch codes in c (version 3.1)
19. Oren, Y., Sadeghi, A.R., Wachsmann, C.: On the effectiveness of the remanence decay side-channel to clone memory-based pufs. In: Cryptographic Hardware and Embedded Systems-CHES 2013, pp. 107–125. Springer (2013)
20. ARM: Trustzone. `http://www.arm.com/products/processors/technologies/trustzone`, (last accessed May 5, 2015)
21. GENODE: An exploration of arm trustzone technology. `http://genode.org/documentation/articles/trustzone`, (last accessed May 1, 2015)
22. GlobalPlatform: Tee client api specification version 1.0 (2010)
23. Integrated Silicon Solution Inc: IS61LV6416-10TL. `http://www.alldatasheet.com/datasheet-pdf/pdf/505020/ISSI/IS61LV6416-10TL.html`
24. ISO/IEC: 15946-5: 2009 information technology-security techniques: Cryptographic techniques based on elliptic curves: Part 5: Elliptic curve generation (2009)
25. Proxama: `http://www.proxama.com/platform/` (2015)
26. Sansa Security: Discretix. `https://www.sansasecurity.com/blog/discretix-becomes-sansa-security/` (2014), (last accessed June 22, 2014)
27. Trusted Computing Group: TPM main specification version1.2, revision 116. `http://www.trustedcomputinggroup.org` (2011), (last accessed October 25, 2014)
28. Trusted Computing Group: Trusted platform module library, family 2.0. `http://www.trustedcomputinggroup.org` (2013), (last accessed March 10, 2015)
29. Xilinx: Zynq-7000 all programmable soc zc702 evaluation kit. `http://www.xilinx.com/products/boards-and-kits/EK-Z7-ZC702-G.htm`
30. Wachsmann, C., Chen, L., Dietrich, K., Löhr, H., Sadeghi, A.R., Winter, J.: Lightweight anonymous authentication with tls and daa for embedded mobile devices. In: Information Security, pp. 84–98. Springer (2011)
31. Wilson, P., Frey, A., Mihm, T., Kershaw, D., Alves, T.: Implementing embedded security on dual-virtual-cpu systems. IEEE Design & Test 24(6), 582–591 (2007)
32. Xi, L., Yang, K., Zhang, Z., Feng, D.: Daa-related apis in tpm 2.0 revisited. In: Trust and Trustworthy Computing, pp. 1–18. Springer (2014)
33. Yang, B., Feng, D., Qin, Y.: A lightweight anonymous mobile shopping scheme based on daa for trusted mobile platform. In: TrustCom, 2014 IEEE 13th International Conference on. pp. 9–17. IEEE (2014)
34. Zhang, Q., Zhao, S., Xi, L., Feng, W., Feng, D.: Mdaak: A flexible and efficient framework for direct anonymous attestation on mobile devices. In: Information and Communications Security. Springer (2014)
35. Zhao, S., Zhang, Q., Hu, G., Qin, Y., Feng, D.: Providing root of trust for arm trustzone using on-chip sram. In: Proceedings of the 4th International Workshop on Trustworthy Embedded Devices. pp. 25–36. ACM (2014)