Lightweight Anonymous Subscription with Efficient Revocation

Vireshwar Kumar
viresh@vt.edu
Virginia Tech

He Li
heli@vt.edu
Virginia Tech

Noah Luther
nluther@vt.edu
Virginia Tech

Pranav Asokan
pranav06@vt.edu
Virginia Tech

Jung-Min (Jerry) Park
jungmin@vt.edu
Virginia Tech

Kaigui Bian
bkg@pku.edu.cn
Peking University

Martin B. H. Weiss
mbw@pitt.edu
University of Pittsburgh

Taieb Znati
znati@cs.pitt.edu
University of Pittsburgh

Abstract

In an anonymous subscription system (ASS), a subscribed user (SU) is able to access the services of a service provider without having to reveal its true identity. For a SU computing platform that is compliant with the Trusted Platform Module (TPM) standard, direct anonymous attestation (DAA) is an appropriate cryptographic protocol for realizing ASS, since DAA enables privacy-preserving authentication of the SU platform. This approach takes advantage of a cryptographic key that is securely embedded in the platform's hardware. Although the computing industry and academia have made significant strides in developing secure and sound DAA schemes, these schemes share a common drawback that may act as a major obstacle to their widespread deployment. In all of the existing schemes, the SU suffers from significant computational and communication costs that increase proportionally to the size of the revocation list. This drawback renders the existing schemes to be impractical when the size of the revocation list grows beyond a relatively modest size. In this paper, we propose a novel scheme called *Lightweight Anonymous Subscription with Efficient Revocation* (LASER) that addresses this very problem. In LASER, the computational and communication costs of the SU's signature are multiple orders of magnitude lower than the prior art. LASER achieves this significant performance improvement by shifting most of the computational and communication costs from the DAA's online proce-

dure (i.e., signature generation) to its offline procedure (i.e., acquisition of keys/credentials). We have conducted a thorough analysis of LASER's performance-related features and compared the findings to the prior art. We have also conducted a comprehensive evaluation of LASER by implementing it on a laptop platform with an on-board TPM. To the best of our knowledge, the results presented in this paper represent the *first* implementation and analysis of a scheme using an actual TPM cryptoprocessor that is compliant with the most recent TPM specification version 2.0. We have thoroughly analyzed the security of LASER in the random oracle model.

1 Introduction

There has been a rapid growth in the online electronic subscription services where the subscribed users (SUs) access contents (e.g., video) and/or resources (e.g., software) made available by the service providers (SPs). In many subscription services, the SUs have significant concerns for remaining anonymous and preserving privacy so that the SPs cannot track their interests, usage patterns, geographical locations, and other personal details. The notion of anonymity in the subscription services has been further fueled by multiple data breaches in the recent years [1]. Hence, there is a major research thrust to develop an *Anonymous Subscription System* (ASS) in which the SPs provide access of their services to the subscribed and authenticated, but anonymous SUs [2, 3, 4, 5]. *Direct Anonymous Attestation* (DAA) is the most appropriate cryptographic protocol to realize the ASS as DAA enables privacy-preserving authentication of a SU's platform by utilizing the cryptographic key which is securely embedded into the platform's hardware [6].

A DAA scheme involves three entities—a platform, a verifier and an issuer. We consider the SU as the platform's user, and the SP as the verifier. The role of the issuer is to generate and issue keys/credentials to platforms, and revoke compromised or insecure platforms by updating and publishing revocation lists. A platform (SU) consists of a host and a *trusted platform module* (TPM). The TPM is a secure and dedicated cryptoprocessor which is designed to secure the platform by integrating its cryptographic key into its hardware [7]. The TPM generates the anonymous signature on the *login request* message sent by the platform to the verifier. The host utilizes the credentials obtained from the issuer to assist the TPM in the generation of the signature by performing most of the computationally expensive operations. The verifier (SP) verifies the validity of the signature received from the platform. As part of the verification process, the verifier also checks the revocation status of the platform. In this paper, the signature on the login request message is called a "login signature".

The Trusted Computing Group (TCG) has standardized the elliptic curve cryptography (ECC)-based DAA in the most recent TPM specification version 2.0 [7, 8, 9]. This TPM specification has also been published as the international standard ISO/IEC 11889:2015 [10]. With a major thrust to standardize DAA for the industry, Intel has incorporated 2.4 billion keys for Enhanced Privacy ID

(EPID) in the chipsets and processors shipped between the years 2008 and 2016 [11, 12]. EPID is a revised DAA scheme with the support for signature-based revocation (i.e., the ability of revoking a platform based on its malicious signature). Note that platforms with these chipsets also support other DAA schemes similar to EPID [13]. Intel's Software Guard Extensions (SGX) technology which is utilized for secure remote computation and digital rights management also provides support for DAA schemes [14].

Although the computing industry and academia have made noteworthy strides in improving the security and efficacy of DAA schemes in recent years, all the existing schemes in the literature still share a common critical drawback of inefficient revocation check procedures that may hinder their widespread adoption [8, 9, 13]. To support the revocation check procedure, the existing DAA schemes employ a signature-based revocation list [8, 13]. This list contains tuples retrieved from the malicious login signatures which are generated by the malicious platforms. In these schemes, for each revoked tuple included in the revocation list, a platform needs to generate a proof of non-revocation of its secret key with respect to the tuple, and include it as a part of its signature. Hence, two things increase linearly with the number of revoked platforms indicated in the revocation list [15]: (1) the computational complexity of generating the login signature; and (2) the communication overhead in terms of the length of the login signature. These shortcomings are common to the existing DAA schemes, and they pose a significant technical challenge in terms of the deployment of DAA in real-world applications—i.e., the computational complexity and the communication overhead become unacceptably high for most applications when the revocation list grows beyond a modest size (e.g., a few hundred). This challenge also implies that the existing schemes are not appropriate for the subscription systems that have stringent latency requirements. Note that several hundreds of platforms are revoked per day in a network with a large number (e.g., a million) of users [16, 17]. Also, in the case of a discovery of a severe vulnerability, several thousands of TPM secret keys may need to be revoked at once [18].

In this paper, we propose a novel ECC-based DAA scheme called *Lightweight Anonymous Subscription with Efficient Revocation* (LASER) which addresses the problem of revocation scalability in DAA. In LASER, the computational complexity of the login signature generation procedure and communication overhead of the login signature are multiple orders of magnitude lower than the prior art. LASER achieves this significant performance improvement by shifting most of the computational and communication costs (due to the revocation check procedure) from the DAA's *online* procedure (i.e., login signature generation) to its *offline* procedure (i.e., acquisition of keys/credentials). This strategy significantly improves the practicality of DAA in real-world applications, because the critical performance bottlenecks of those applications are determined by the performance of the online procedure. Unlike legacy DAA schemes, LASER is scalable, and can be deployed in the large subscription system which is expected to have a long revocation list.

Further, in the existing DAA schemes, the login signatures generated by a

platform satisfy the concept of *absolute unlinkability* (i.e., the signatures are unlinkable by the issuer and the verifier). Here, linking two signatures means that they are proved to be generated by the same platform. In addition to the concept of absolute unlinkability, our design of LASER leads to a novel concept of unlinkability called *conditional unlinkability* where the generated signatures remain unlinkable by the verifier, but they may or may not be linkable by the issuer. In this paper, we propose the notion of *adaptable unlinkability* which implies that the platform is able to adaptably select one of the two concepts of unlinkability of its login signatures. Our results exhibit that the notion of adaptable unlinkability enables LASER to provide the needed privacy in a flexible and practical manner.

The paper's main contributions are summarized below.

1. We propose a novel anonymous subscription scheme called LASER which significantly reduces the computational and communication costs of the login signature generation procedure compared to the prior art.

2. We design LASER using the existing TPM algorithms under the framework of the TPM specification version 2.0. Hence, LASER can be readily deployed in an existing network comprised of platforms with on-board TPM 2.0.

3. LASER and our strategy for designing it provide valuable insights into how to address the open problem of reducing the costs of revocation in DAA. Furthermore, we expect that the notion of adaptable unlinkability may pave the way for designing practical and scalable DAA schemes.

4. We have validated our analytical results by implementing LASER on a laptop platform with an on-board TPM. To the best of our knowledge, *the results presented in this paper represent the first implementation and analysis of an anonymous authentication scheme using an actual TPM cryptoprocessor that is compliant with the TPM specification version 2.0.*

5. We have analyzed the security of LASER in the random oracle model under the complexity assumptions which are widely utilized in analyzing the security of DAA schemes.

2 Related Work

The anonymous authentication schemes (e.g., group signatures [19]) can preserve the privacy of the SUs by decoupling their identities from their login signatures communicated to the SPs. However, in these schemes, a malicious SU can share its subscription keys with other SUs, and may facilitate authorized access to unsubscribed users. Blanton proposed the first practical scheme to realize the ASS [2]. In this scheme, a SP provides access of its services to an authenticated SU in each authentication epoch such that two login requests by the same SU in two different epochs remain unlinkable. The SP also ensures that there is only one login during the epoch using the credential allotted to the SU, and there is no sharing of credential. However, in this scheme, there

exists a trade-off between the SP's desire for a long epoch (to reduce SP-side computations) and SU's desire for a short epoch (so that it can "re-anonymize" its different login requests). Lee et al. mitigate this trade-off by having short epochs, and proposing an efficient re-authentication technique for the SUs who do not need unlinkability between two epochs [3]. However, this scheme is limited by the lack of the revocation algorithm and the requirement of tight time synchronization among all the entities in the ASS.

We assert that *Direct Anonymous Attestation* (DAA) is the most appropriate cryptographic protocol for realizing ASS [6]. DAA enables privacy-preserving authentication of a SU's platform by utilizing the cryptographic key which is securely embedded in the platform's hardware. The embedded key cannot be retrieved and distributed by the SU. The aforementioned trade-off related to the authentication epochs does not exist in DAA. Also, no tight time synchronization is required among the entities in DAA.

Brickell et al. proposed the first instantiation of RSA-based DAA for anonymous attestation of a computing platform [6]. This work has been followed by a number of enhancements to DAA. The most notable RSA-based DAA scheme is called the Enhanced Privacy ID (EPID) [12] which supplements DAA with a revocation check procedure. Compared to RSA-based DAA schemes, the ECC-based DAA schemes are significantly more efficient in terms of the computational complexity and communication overhead [8, 13, 20, 21]. However, in the schemes which support signature-based revocation, the revocation check procedures are very inefficient in terms of the computational and communication costs [22, 15, 23]. The computational complexity of the signature generation algorithm at the platform, and the length of the signature increase linearly with the number of revoked platforms. Hence, the existing DAA schemes are *not scalable*—i.e., they are impractical to use for applications in which the length of the revocation list is expected to grow beyond a modest size (e.g., a few hundred).

To mitigate the high overhead of the revocation check procedure, Brickell and Li [12], and Chen and Li [15] proposed a somewhat trivial solution—it requires the issuer to "reset" the group that it is managing (i.e., replace all credentials and keys of the platforms with new ones) when the number of tuples in the revocation list exceeds a pre-determined threshold value. Note that in order to reset a group, each platform needs to re-establish a communication link with the issuer, obtain the credentials, and execute corresponding computations. Hence, this approach is far from a panacea, and is impractical for a large network with a large number of platforms. In this paper, LASER along with the proposed notion of adaptable unlinkability addresses the open problem of reducing the cost of revocation in DAA making it more practical to be used for the ASS.

In terms of implementation results in the prior art, either the functionality of the TPM is only simulated in trusted execution environments, e.g., ARM TrustZone [24], or the results are obtained from the implementation on TPM version 1.2 [25]. Note that the simulation environment cannot correctly exhibit the limitations of a real embedded TPM chip. The results in this paper represent the first implementation and analysis using an actual TPM that is compliant

with the most recent TPM specification version 2.0.

3 Overview of LASER

We propose a novel DAA scheme called *Lightweight Anonymous Subscription with Efficient Revocation* (LASER) which mitigates the problem of revocation scalability plaguing the existing DAA schemes. We formally define the components of LASER as follows.

Definition 1. LASER is composed of the following protocols.

1. $(\mathtt{gpk}, \mathtt{isk}) \leftarrow \mathsf{Setup}(1^\lambda)$: This setup algorithm is run by the issuer. The input to this algorithm is a security parameter 1^λ, where $\lambda \in \mathbb{N}$. Here, $\mathbb{N}$ represents the set of natural numbers. This algorithm outputs an *issuer's secret key*, represented by $\mathtt{isk}$, and a corresponding *group public key*, represented by $\mathtt{gpk}$. The $\mathtt{isk}$ is known only to the issuer, and $\mathtt{gpk}$ is published.

2. $(\mathtt{tsk}, \mathtt{hdl}, \mathtt{tpk}, \mathtt{mcl}) \leftarrow \mathsf{MemCreGen}(\mathtt{gpk}, \mathtt{isk}, m_s)$: To join the group created by the issuer, the TPM and the host of the platform run this registration protocol with the issuer. The inputs to the issuer are $\mathtt{gpk}$ and $\mathtt{isk}$, the input to the TPM is $\mathtt{gpk}$, and the inputs to the host are $\mathtt{gpk}$ and $m_s \in \mathbb{N}$. Here, m_s represents the number of absolutely unlinkable credentials (discussed in Section 4.2) allotted to each platform, and its value is set by the host based on the unlinkability requirement of the platform. In this protocol, the TPM generates a *TPM's secret key*, represented by $\mathtt{tsk}$, a corresponding *TPM's public key*, represented by $\mathtt{tpk}$, and a *key handle*, represented by $\mathtt{hdl}$. The $\mathtt{hdl}$ specifies the location of the $\mathtt{tsk}$ in the secure memory of TPM. The $\mathtt{tsk}$ is known only to the TPM, and $\mathtt{tpk}$ and $\mathtt{hdl}$ are forwarded to the host. Further, the host acquires a *membership credential*, represented by $\mathtt{memCre}_\mathtt{j}$, for each $j \in [1, m_s]$, from the issuer. Finally, the host outputs a *membership credential list*, represented by $\mathtt{mcl} = (\mathtt{memCre}_1, \cdots, \mathtt{memCre}_{\mathtt{m_s}})$.

3. $(\mathtt{ctl}', \mathtt{logCre}_\mathtt{j}) \leftarrow \mathsf{LogCreGen}(\mathtt{gpk}, \mathtt{isk}, \mathtt{ctl}, \mathtt{tsk}, \mathtt{hdl}, \mathtt{tpk}, \mathtt{memCre}_\mathtt{j})$: The TPM and the host run this login credential acquisition protocol with the issuer. In this protocol, the inputs to the issuer are $\mathtt{gpk}$, $\mathtt{isk}$, and a *credential token list*, represented by $\mathtt{ctl}$, the inputs to the TPM are $\mathtt{gpk}$ and $\mathtt{tsk}$, and the inputs to the host are $\mathtt{gpk}$, $\mathtt{hdl}$, $\mathtt{tpk}$, and $\mathtt{memCre}_\mathtt{j}$. The $\mathtt{ctl}$ is securely stored and maintained by the issuer. In this protocol, the issuer outputs an updated list $\mathtt{ctl}'$, and the host acquires a *login credential*, represented by $\mathtt{logCre}_\mathtt{j}$, from the issuer.

4. $(\mathtt{logCre}_\mathtt{j}, \mathtt{cul}') \leftarrow \mathsf{SelectLogCre}(\mathtt{lcl}, \mathtt{cul}, \mathtt{csr})$: This credential selection algorithm is performed by the host. The inputs to the host are a *login credential list*, represented by $\mathtt{lcl}$, a *credential-usage list*, represented by $\mathtt{cul}$, and a *credential-selection rule*, represented by $\mathtt{csr}$. Before running this algorithm, the TPM and the host run the $\mathsf{LogCreGen}$ protocol with the issuer for all $j \in [1, m_s]$ to obtain the login credential list $\mathtt{lcl} = (\mathtt{logCre}_1, \cdots, \mathtt{logCre}_{\mathtt{m_s}})$.

Each instance of the LogCreGen protocol is initiated by the host after a randomly selected time interval. To keep track of the utilized credentials from $\texttt{lcl}$, the host maintains the credential-usage list $\texttt{cul}$. The host also employs an application-based rule assignment for $\texttt{csr}$ which takes one of the two values, i.e., $\texttt{csr} \in \{\texttt{absUnlink}, \texttt{conUnlink}\}$, corresponding to the application-based request to generate an absolutely unlinkable signature or a conditionally unlinkable signature, respectively. In this algorithm, the host selects the value of $j \in [1, m_s]$ based on the rule $\texttt{csr}$ and the current usage list $\texttt{cul}$. It outputs a login credential $\texttt{logCre}_\texttt{j}$ from $\texttt{lcl}$. It also outputs the updated list $\texttt{cul}'$.

5. $\sigma_s \leftarrow \mathsf{Sign}(\texttt{gpk}, \texttt{tsk}, \texttt{hdl}, \texttt{tpk}, \texttt{logCre}_\texttt{j}, M)$: This login signature generation protocol is performed between the TPM and the host. The inputs to the TPM are $\texttt{gpk}$ and $\texttt{tsk}$, and the inputs to the host are $\texttt{gpk}$, $\texttt{hdl}$, $\texttt{tpk}$, $\texttt{logCre}_\texttt{j}$ and a login request message, represented by $M \in \{0,1\}^*$. This protocol outputs a login signature σ_s.

6. $\texttt{valid/invalid} \leftarrow \mathsf{Verify}(\texttt{gpk}, \sigma_s, M, \texttt{tRL}, \texttt{kRL})$: This verification algorithm takes $\texttt{gpk}$, a purported login signature σ_s, a login request message M, a *token-based revocation list*, represented by $\texttt{tRL}$, and a *key-based revocation list*, represented by $\texttt{kRL}$, as inputs. The $\texttt{tRL}$ and $\texttt{kRL}$ are maintained and published by the issuer. This algorithm verifies: (1) whether the signature is honestly generated, (2) whether the login credential used to generate the signature is not revoked, and (3) whether the TPM's secret key is not revoked. If all of these three verification steps are successful, this algorithm outputs the value $\texttt{valid}$; otherwise, it outputs the value $\texttt{invalid}$.

7. Revoke: Using this algorithm, the issuer revokes the keys and credentials of a compromised platform. The revocation can be driven by the verifier, the issuer or the platform's user. Hence, this algorithm comprises of three sub-algorithms, and only one of these sub-algorithms is run based on the available inputs.

   (a) $\texttt{tRL}' \leftarrow \mathsf{RevokeSign}(\texttt{gpk}, \texttt{ctl}, \sigma_s, M, \texttt{tRL}, \texttt{kRL})$: This is the verifier-driven signature-based revocation sub-algorithm which is utilized when the verifier provides a malicious signature from a platform to the issuer. It takes $\texttt{gpk}$, $\texttt{ctl}$, a signature σ_s, a message M, $\texttt{tRL}$, and $\texttt{kRL}$, as inputs. It updates the $\texttt{tRL}$, and outputs the updated list, $\texttt{tRL}'$.

   (b) $\texttt{tRL}' \leftarrow \mathsf{RevokeTpk}(\texttt{gpk}, \texttt{ctl}, \texttt{cre}, \texttt{tpk}, \texttt{tRL})$: This is the platform's user-driven TPM's public key-based revocation sub-algorithm which is utilized when the platform cannot generate an honest signature on a message (e.g., when the platform is stolen). It takes $\texttt{gpk}$, $\texttt{ctl}$, a membership or login credential $\texttt{cre}$, $\texttt{tpk}$, and $\texttt{tRL}$, as inputs. It updates the $\texttt{tRL}$, and outputs the updated list, $\texttt{tRL}'$.

   (c) $\texttt{kRL}' \leftarrow \mathsf{RevokeTsk}(\texttt{gpk}, \texttt{cre}, \texttt{tsk}, \texttt{kRL})$: This is the issuer-driven TPM's secret key-based revocation sub-algorithm which is utilized when the

issuer finds that the TPM's secret key has been extracted from the TPM, and published along with a valid membership or login credential. It takes gpk, a membership/login credential cre, tsk, and kRL, as inputs. It updates the kRL, and outputs the updated list, kRL′.

8. true/false ← Identify(gpk, σ_s, M, $\mathtt{tsk}_*$): This signature tracing algorithm is required to characterize the security properties. It takes gpk, a signature σ_s, a message M, and a TPM's secret key $\mathtt{tsk}_*$ as inputs. It outputs the value true if σ_s is proved to be generated using $\mathtt{tsk}_*$; otherwise, it outputs the value false.

We paraphrase the above definition of LASER as follows. In LASER, the platform obtains two types of credentials—(1) membership credentials through the MemCreGen protocol, and (2) login credentials through the LogCreGen protocol. In the MemCreGen protocol, the platform registers with the issuer using the TPM's secret key, and obtains m_s membership credentials. In the LogCreGen protocol, the platform utilizes a membership credential, and acquires a corresponding login credential from the issuer. The platform executes the LogCreGen protocol for m_s number of times at random time intervals. This means that multiple instances of LogCreGen protocol are performed with the issuer by multiple platforms within a time interval, and hence the issuer cannot learn the relationship among the login credentials by the timing and cannot link the acquired login credentials to a particular platform. Through the Sign protocol, the platform utilizes a login credential to generate a signature on the login request message communicated to the verifier. Through the Verify algorithm, the verifier checks the validity of the signature, and the revocation status of the platform.

To support the revocation check procedure, the issuer publishes two revocation lists: (1) the token-based revocation list tRL which contains all the revoked tokens, and (2) the key-based revocation list kRL which contains all the revoked TPM's secret keys. Also, the issuer includes a revocation token, represented by y_j, in each login credential. In the Sign protocol, the platform includes the TPM's secret key tsk and the revocation token y_j as exponents over randomly selected bases. In the Verify algorithm, the verifier determines the revocation status of tsk corresponding to each revoked key included in kRL, and the revocation status of y_j corresponding to each revoked token included in tRL.

Recall that in the existing DAA schemes, for each revoked tuple included in the revocation list, the platform needs to generate a proof of non-revocation of its TPM's secret key with respect to the tuple, and include it as a part of its login signature [15]. Unlike the existing DAA schemes, in LASER, the platform does not need to generate any proof of knowledge of non-revocation of its TPM's secret key. However, in LASER, the platform needs to obtain m_s login credentials by running the LogCreGen protocol for m_s number of times. In this way, most of the burden of the revocation check procedure in LASER is shifted from the online login signature generation to the offline acquisition of login credentials. This unique feature of LASER brings about two practical

advantages. Firstly, during the login signature generation protocol, the TPM is *not* burdened with any computations related to the revocation check procedure, and this results in a significant reduction in the computational complexity of login signature generation. Secondly, the length of the login signature generated by the platform and communicated to the verifier is constant, and does not grow proportionally with the length of the revocation list. These two advantages are especially important when a DAA scheme needs to be deployed for the ASS with a large number of SUs.

4 Security Definitions

Most of the ECC-based DAA schemes are proved secure in the game-based security model presented by Brickell et al. [20]. Over the years, the game-based security model for DAA has been further developed [25, 26]. However, Camenisch et al. have identified specific shortcomings in the security models utilized in the existing DAA schemes, and have presented a simulation-based security model for DAA [13, 27]. In this paper, LASER utilizes the game-based security model which does not suffer from the shortcomings mentioned in [13]. In the following discussions, we present the definitions of the security properties of LASER.

4.1 Correctness

The correctness property implies that a signature generated using the login signature generation protocol by an unrevoked platform is correctly verified using the verification algorithm, and is correctly traced back to the platform, as defined below.

Definition 2. For all λ, m_s, $\mathtt{ctl}$, $\mathtt{cul}$, $\mathtt{csr}$, M, $\mathtt{tRL}$, and $\mathtt{kRL}$, LASER satisfies the correctness property if

$$
\begin{aligned}
(\mathtt{gpk}, \mathtt{isk}) &\leftarrow \mathsf{Setup}(1^{\lambda}),\\
(\mathtt{tsk}, \mathtt{hdl}, \mathtt{tpk}, \mathtt{mcl}) &\leftarrow \mathsf{MemCreGen}(\mathtt{gpk}, \mathtt{isk}, m_s),\\
(\mathtt{ctl}', \mathtt{logCre_j}) &\leftarrow \mathsf{LogCreGen}(\mathtt{gpk}, \mathtt{isk}, \mathtt{ctl}, \mathtt{tsk}, \mathtt{hdl}, \mathtt{tpk}, \mathtt{memCre_j}), \forall j \in [1, m_s],\\
(\mathtt{logCre_j}, \mathtt{cul}') &\leftarrow \mathsf{SelectLogCre}(\mathtt{lcl}, \mathtt{cul}, \mathtt{csr}), \text{and}\\
\sigma_s &\leftarrow \mathsf{Sign}(\mathtt{gpk}, \mathtt{tsk}, \mathtt{hdl}, \mathtt{tpk}, \mathtt{logCre_j}, M), \text{then}\\
\mathtt{valid} &\leftarrow \mathsf{Verify}(\mathtt{gpk}, \sigma_s, M, \mathtt{tRL}, \mathtt{kRL}), \text{and}\\
\mathtt{true} &\leftarrow \mathsf{Identify}(\mathtt{gpk}, \sigma_s, M, \mathtt{tsk}).
\end{aligned}
$$

4.2 Adaptable Anonymity

The notion of adaptable anonymity consists of the following two properties.

1. *Anonymity*: This property requires that no entity (including the issuer) is able to identify the platform which has generated a given signature.

2. *Adaptable unlinkability*: This notion requires that the platform is able to adaptably control whether or not any two signatures can be linked by a particular entity. This is a new notion of unlinkability in a DAA scheme introduced in this paper. For any two signatures, the platform may select *one* of the following two properties of adaptable unlinkability.

   (a) *Absolute unlinkability*: The two signatures cannot be linked either by the issuer or by the verifier.

   (b) *Conditional unlinkability*: The two signatures cannot be linked by the verifier, but they may or may not be linkable by the issuer.

The definition of adaptable anonymity follows the definitions of absolute and conditional unlinkability as presented below.

Definition 3. For an adversary $\mathcal{A}$ and a challenger $\mathcal{C}$, the *absolute unlinkability game* is defined as follows.

1. *Setup*: $\mathcal{C}$ simulates the Setup algorithm, and provides $\mathcal{A}$ with the resulting $\mathtt{isk}$ and $\mathtt{gpk}$. $\mathcal{C}$ also creates m_t platforms with identities $\mathcal{P}_i$, $\forall i \in [1, m_t]$. Further, $\mathcal{C}$ initializes a list of the corrupted platforms, $\mathtt{cpl} = \varnothing$, and a list of corrupted credentials, $\mathtt{ccl} = \varnothing$. Here, $\varnothing$ represents an empty set.

2. *MemCreGen*: $\mathcal{C}$ acts as the platform, and simulates the protocol $\mathsf{MemCreGen}$ with $\mathcal{A}$ which acts as the issuer. For all $i \in [1, m_t]$, $\mathcal{C}$ generates $\mathtt{tsk_i}$, and acquires $\mathtt{memCre_{ij}}$, $\forall j \in [1, m_s]$.

3. *LogCreGen*: $\mathcal{C}$ acts as the platform, and simulates the $\mathsf{LogCreGen}$ protocol with $\mathcal{A}$ which acts as the issuer. For all $i \in [1, m_t]$ and $j \in [1, m_s]$, $\mathcal{C}$ acquires $\mathtt{logCre_{ij}}$.

4. *Queries-Phase I*: $\mathcal{A}$ queries $\mathcal{C}$ about the following.

   (a) *Sign*: $\mathcal{A}$ requests $\mathcal{C}$ to generate a signature on a message M on behalf of $\mathcal{P}_i$. $\mathcal{C}$ runs the $\mathsf{SelectLogCre}$ with the credential-selection rule $\mathtt{conUnlink}$ or $\mathtt{absUnlink}$. If $\mathcal{C}$ utilizes the rule $\mathtt{absUnlink}$, it appends $\mathtt{logCre_{ij}}$ to $\mathtt{ccl}$. Further, $\mathcal{C}$ simulates the Sign protocol, and responds to $\mathcal{A}$ with the signature σ_s.

   (b) *TskCorrupt*: $\mathcal{A}$ requests $\mathcal{C}$ to provide the TPM's secret key of $\mathcal{P}_i$. $\mathcal{C}$ responds to $\mathcal{A}$ with $\mathtt{tsk_i}$, and appends $\mathcal{P}_i$ to $\mathtt{cpl}$.

   (c) *MemCreCorrupt*: $\mathcal{A}$ requests the j^{th} membership credential of platform $\mathcal{P}_i$. $\mathcal{C}$ responds to $\mathcal{A}$ with $\mathtt{memCre_{ij}}$, and appends the corresponding $\mathtt{logCre_{ij}}$ to $\mathtt{ccl}$.

   (d) *LogCreCorrupt*: $\mathcal{A}$ requests the j^{th} login credential of platform $\mathcal{P}_i$. $\mathcal{C}$ responds to $\mathcal{A}$ with $\mathtt{logCre_{ij}}$, and appends $\mathtt{logCre_{ij}}$ to $\mathtt{ccl}$.

5. *Challenge*: $\mathcal{A}$ submits to $\mathcal{C}$ a message M and two platforms $\mathcal{P}_{i_0}$ and $\mathcal{P}_{i_1}$ with the restriction that $\mathcal{P}_{i_0}$, $\mathcal{P}_{i_1} \notin \mathtt{cpl}$. $\mathcal{C}$ selects $\phi \leftarrow_{\$} \{0, 1\}$. Here, $\leftarrow_{\$}$

represents a random selection. Corresponding to $\mathcal{P}_{i_\phi}$, $\mathcal{C}$ selects $\texttt{logCre}_{\texttt{i}_\phi\texttt{j}_\phi}$ using the credential-selection rule `absUnlink` in the SelectLogCre algorithm such that $\texttt{logCre}_{\texttt{i}_\phi\texttt{j}_\phi} \notin \texttt{ccl}$. Further, $\mathcal{C}$ runs the Sign protocol, and responds with the signature σ_s on M.

6. *Queries-Phase II (Restricted Queries)*: After obtaining the challenge, $\mathcal{A}$ continues to probe $\mathcal{C}$ with the queries mentioned in *Queries-Phase I*, except for the *TskCorrupt* queries for $\texttt{tsk}_{\texttt{i}_0}$ and $\texttt{tsk}_{\texttt{i}_1}$, *MemCreCorrupt* queries for $\texttt{memCre}_{\texttt{i}_0\texttt{j}_0}$ and $\texttt{memCre}_{\texttt{i}_1\texttt{j}_1}$, and *LogCreCorrupt* queries for $\texttt{logCre}_{\texttt{i}_0\texttt{j}_0}$ and $\texttt{logCre}_{\texttt{i}_1\texttt{j}_1}$.

7. *Output*: $\mathcal{A}$ outputs a bit ϕ' indicating its guess of ϕ.

$\mathcal{A}$ wins the game if $\phi' = \phi$, and the advantage of $\mathcal{A}$ is defined as $Adv_{\mathcal{A}}^{abs} = |\Pr(\phi' = \phi) - 1/2|$, where Pr represents the probability of an event. LASER with the rule `absUnlink` satisfies the absolute unlinkability property if the advantage of any probabilistic polynomial time (PPT) adversary on winning the absolute unlinkability game is negligibly small.

Definition 4. For an adversary $\mathcal{A}$ and a challenger $\mathcal{C}$, the *conditional unlinkability game* is defined as follows.

1. *Setup*: $\mathcal{C}$ simulates the Setup algorithm, and provides $\mathcal{A}$ with the resulting `gpk`. $\mathcal{C}$ does not reveal `isk` to $\mathcal{A}$. $\mathcal{C}$ also creates m_t platforms with identities $\mathcal{P}_i$, $\forall i \in [1, m_t]$. Further, $\mathcal{C}$ initializes two lists `cpl` and `ccl`.

2. *MemCreGen*: $\mathcal{C}$ acts as the platform as well as the issuer, and simulates MemCreGen protocol to generate $\texttt{tsk}_\texttt{i}$, $\forall i \in [1, m_t]$, and $\texttt{memCre}_\texttt{ij}$, $\forall i \in [1, m_t], j \in [1, m_s]$.

3. *LogCreGen*: $\mathcal{C}$ acts as the platform as well as the issuer, and simulates LogCreGen protocol to generate $\texttt{logCre}_\texttt{ij}$, $\forall i \in [1, m_t]$ and $\forall j \in [1, m_s]$.

4. *Queries-Phase I*: $\mathcal{A}$ probes $\mathcal{C}$ with the queries defined in the *Queries-Phase I* in the absolute unlinkability game. $\mathcal{A}$ probes $\mathcal{C}$ with the following additional queries.

   (a) *Revoke*: $\mathcal{A}$ requests $\mathcal{C}$ to revoke the login credential utilized to generate a signature σ'_s on a message M'. $\mathcal{C}$ simulates the Revoke algorithm and responds with the updated $\texttt{tRL}'$.

5. *Challenge*: $\mathcal{A}$ submits to $\mathcal{C}$ a message M and two platforms $\mathcal{P}_{i_0}$ and $\mathcal{P}_{i_1}$ with the restriction that $\mathcal{P}_{i_0}, \mathcal{P}_{i_1} \notin \texttt{cpl}$. $\mathcal{C}$ selects $\phi \leftarrow_\$ \{0, 1\}$. Corresponding to $\mathcal{P}_{i_\phi}$, $\mathcal{C}$ selects $\texttt{logCre}_{\texttt{i}_\phi\texttt{j}_\phi}$ using the credential-selection rule `conUnlink` in the SelectLogCre algorithm such that $\texttt{logCre}_{\texttt{i}_\phi\texttt{j}_\phi} \notin \texttt{ccl}$. Further, $\mathcal{C}$ runs the Sign protocol, and responds with the signature σ_s on M.

6. *Queries-Phase II (Restricted Queries)*: $\mathcal{A}$ probes $\mathcal{C}$ with the restricted queries defined in the *Queries-Phase II* in the absolute unlinkability game.

7. *Output*: $\mathcal{A}$ outputs a bit ϕ' indicating its guess of ϕ.

$\mathcal{A}$ wins the game if $\phi' = \phi$, and the advantage of $\mathcal{A}$ is defined as $Adv_{\mathcal{A}}^{con} = |\Pr(\phi' = \phi) - 1/2|$. LASER with the rule `conUnlink` satisfies the conditional unlinkability property if the advantage of any PPT adversary on winning the conditional unlinkability game is negligibly small.

Definition 5. LASER satisfies the adaptable anonymity property if LASER satisfies the absolute unlinkability property for the signatures generated using the credential-selection rule `absUnlink`, and the conditional unlinkability property for the signatures generated using the credential-selection rule `conUnlink`.

4.3 Traceability

The traceability property implies that no colluding set of platforms can create a valid signature that can not be traced back to any platform, as defined below.

Definition 6. For an adversary $\mathcal{A}$ and a challenger $\mathcal{C}$, the *traceability game* is defined as follows.

1. The *Setup*, *MemCreGen*, *LogCreGen* and *Queries* phases in this game are defined in the same manner as the *Setup*, *MemCreGen*, *LogCreGen* and *Queries* phases in conditional unlinkability game, respectively.
2. *Output*: $\mathcal{A}$ outputs a message M_* and a signature σ_* for given $\mathtt{tRL}_*$ and $\mathtt{kRL}_*$.

$\mathcal{A}$ wins the above traceability game if:

1. $\mathtt{valid} \leftarrow \mathsf{Verify}(\mathtt{gpk}, \sigma_*, M_*, \mathtt{tRL}_*, \mathtt{kRL}_*)$;
2. $\mathcal{A}$ did not obtain σ_* by making a *Sign* query; and
3. $\forall i \in [1, m_t]$, $\mathtt{false} \leftarrow \mathsf{Identify}(\mathtt{gpk}, \sigma_*, M_*, \mathtt{tsk_i})$.

LASER satisfies the traceability property if for any PPT adversary, the probability on winning the traceability game is negligibly small.

4.4 Non-frameability

The non-frameability property implies that no colluding set of entities (including the issuer) can forge a valid signature that can be traced back to a non-colluding platform, as defined below.

Definition 7. For an adversary $\mathcal{A}$ and a challenger $\mathcal{C}$, the *non-frameability game* is defined as follows.

1. The *Setup*, *MemCreGen*, *LogCreGen* and *Queries* phases in this game are defined in the same manner as *Setup*, *MemCreGen*, *LogCreGen* and *Queries* phases in the absolute unlinkability game, respectively.

2. *Output*: $\mathcal{A}$ outputs a message M_* and a signature σ_* corresponding to a platform $\mathcal{P}_{i^*}$.

$\mathcal{A}$ wins the above non-frameability game if:

1. $\mathcal{A}$ did not obtain σ_* by making a *Sign* query;
2. $\mathcal{P}_{i^*} \notin \mathtt{cpl}$; and
3. $\mathtt{true} \leftarrow \mathsf{Identify}(\mathtt{gpk}, \sigma_*, M_*, \mathtt{tsk}_{\mathtt{i}^*})$, where $\mathtt{tsk}_{\mathtt{i}^*}$ is the TPM's secret key of the platform $\mathcal{P}_{i^*}$.

LASER satisfies the non-frameability property if for any PPT adversary, the probability on winning the non-frameability game is negligibly small.

Among the above properties, the adaptable unlinkability is an important notion to consider in evaluating LASER with respect to other DAA schemes. In the existing DAA schemes, the platform obtains a credential, and can generate signatures which are unlinkable by both the verifier and the issuer, and hence have absolute unlinkability. However, in LASER, we observe that the platform obtains m_s login credentials, and has the option to generate signatures in two categories. In the first category, if a signature is generated using a login credential which was not utilized earlier to generate any other signature, neither the issuer and nor the verifier is able to link this signature to any other signature generated by the same platform. Hence, in LASER, the platform can generate m_s signatures which satisfy the absolute unlinkability property. In the second category, if a signature is generated using a login credential which was also utilized earlier, the issuer can link this signature to all the signatures generated previously using the same login credential. But even in this second category, the verifier is not able to determine whether any two signatures are generated using the same login credential. Hence, the platform can generate a large number (which is not limited by the parameter m_s) of signatures which satisfy conditional unlinkability property. Note that in the second category, if two signatures are generated using two different login credentials, even the issuer is not able to determine whether the two signatures are generated by the same platform.

Hence, in terms of security properties, there are two attributes of LASER which distinguish it from all other DAA schemes. First, LASER enables the platform to generate signatures which satisfy conditional unlinkability property. This attribute makes LASER usable in applications which desire a trade-off between the absolute and no unlinkability properties. Second, LASER satisfies the absolute unlinkability property in a limited sense, i.e., the number of absolutely unlinkable signatures is limited by the number of the different login credentials. LASER exploits this attribute to significantly reduce the large computational complexity and communication overhead plaguing the existing DAA schemes.

5 Background and Assumptions

The schemes in the prior art [2, 3] and LASER are the cryptographic protocols utilized by the SU (platform) for the acquisition of the key/credential from

the issuer and generation of anonymous login signatures sent for verification to the SP (verifier). A comprehensive anonymous subscription system requires additional mechanisms (e.g., a network-anonymity service [28] and a private information retrieval scheme [29]) to anonymize other aspects of the SU's interaction with the SP. The discussion of these mechanisms is out of scope of this paper.

We assume that each TPM has a secret endorsement key S_T embedded into it, and there is an associated public key P_T. We also assume that the issuer has a secret/public key pair (S_I, P_I). These keys are utilized for the secure communication between the platform and the issuer. The discussions of the procedures of establishing this secure channel are out of scope of this paper.

Let $\mathbb{Z}_p^*$ represent the set of integers modulo p. Also, let there be a pair of multiplicative cyclic groups of prime order p, $(\mathbb{G}_1, \mathbb{G}_2)$, called a bilinear group pair, such that there exists a group $\mathbb{G}_T$ and a bilinear mapping function, $e : \mathbb{G}_1 \times \mathbb{G}_2 \rightarrow \mathbb{G}_T$. We utilize the Type 3 pairing which means that $\mathbb{G}_1 \neq \mathbb{G}_2$, and there does not exist any computable isomorphism from $\mathbb{G}_1$ to $\mathbb{G}_2$ [30]. Further, we assume that there exists a collision resistant hash function $\mathcal{H}_z : \{0,1\}^* \rightarrow \mathbb{Z}_p^*$, which is treated as a random oracle. LASER utilizes the BBS+ signatures for providing anonymous authentication [31, 32, 33]. We provide the definition of BBS+ signature in Appendix A. The definitions of the group mapping function $\mathcal{H}_g$, and the algorithms TPM2_CreatePrimary, TPM2_Commit and TPM2_Mod_Sign are presented in Appendix B. These concise definitions have been extracted from the elaborate definitions included in the TPM specification version 2.0 [7, 25].

The security of LASER is proved in the random oracle model based on the discrete logarithm (DL) assumption, the decisional Diffie-Hellman (DDH) assumption [34], and the q-strong Diffie-Hellman (q-SDH) assumption [35] which are defined as follows.

Assumption 1 ($\mathbb{G}_1$-DDH Assumption)**.** *Given* $(P, P^a, P^b, P^c) \in \mathbb{G}_1^4$*, where* $a, b \in \mathbb{Z}_p^*$*, as input for each PPT algorithm* $\mathcal{A}$*, the probability with which* $\mathcal{A}$ *is able to differentiate whether* $c = a \cdot b$*, or* $c \leftarrow_{\$} \mathbb{Z}_p^*$*, is negligibly small.*

Assumption 2 (q-SDH Assumption)**.** *Given a* $(q+3)$*-tuple* $\left(g_1, g_1^{\gamma}, \cdots, g_1^{\gamma^q}, g_2, g_2^{\gamma}\right)$*, where* $g_1 \leftarrow_{\$} \mathbb{G}_1$*,* $g_2 \leftarrow_{\$} \mathbb{G}_2$*, and* $\gamma \leftarrow_{\$} \mathbb{Z}_p^*$*, as input for each PPT algorithm* $\mathcal{A}$*, the probability that* $\mathcal{A}$ *outputs a pair* $\left(g_1^{1/(\gamma+z)}, z\right)$*, where* $z \in \mathbb{Z}_p^*$*, is negligibly small.*

6 Details of LASER

In this section, we present the details of the algorithms and the protocols of LASER. We provide the theorems corresponding to each of the four security properties of LASER, and the comprehensive proofs of those theorems in Appendix C.

6.1 Setup

We assume that there exists an asymmetric bilinear group pair ($\mathbb{G}_1$,$\mathbb{G}_2$) of prime order p, a bilinear mapping function $e : \mathbb{G}_1 \times \mathbb{G}_2 \to \mathbb{G}_T$, and a hash function $\mathcal{H}_z : \{0,1\}^* \to \mathbb{Z}_p^*$. The parameters for $(\mathbb{G}_1, \mathbb{G}_2, e, \mathcal{H}_z)$ are published.

$(\mathtt{gpk}, \mathtt{isk}) \leftarrow \mathsf{Setup}(1^\lambda)$

This algorithm is run by the issuer. With the input 1^λ, it outputs `isk` and `gpk`. It proceeds as follows.

1. Select $g_1 \leftarrow_\$ \mathbb{G}_1$ and $g_2 \leftarrow_\$ \mathbb{G}_2$ such that g_1 and g_2 are the generators of $\mathbb{G}_1$ and $\mathbb{G}_2$, respectively.
2. Select $h_1, h_2, h_3 \leftarrow_\$ \mathbb{G}_1$, and $\gamma \leftarrow_\$ \mathbb{Z}_p^*$.
3. Compute $\omega = g_2^\gamma$; and set $\mathtt{isk} = \gamma$, and $\mathtt{gpk} = (g_1, h_1, h_2, h_3, g_2, \omega)$.
4. Output $(\mathtt{gpk}, \mathtt{isk})$.

6.2 Registration

$(\mathtt{tsk}, \mathtt{hdl}, \mathtt{tpk}, \mathtt{mcl}) \leftarrow \mathsf{MemCreGen}(\mathtt{gpk}, \mathtt{isk}, m_s)$

This registration protocol is performed among the TPM, the host and the issuer. The inputs to the issuer are `gpk` and `isk`, the input to the TPM is `gpk`, and the inputs to the host are `gpk` and m_s. In this protocol, the TPM generates `tsk`, stores it in its secure memory, and outputs `tpk` and `hdl`. The host outputs `mcl`. It proceeds as follows.

1. Upon the request of the host, the TPM runs the algorithm TPM2_CreatePrimary (discussed in Appendix B.2), generates $\mathtt{tsk} = f$, and forwards the outputs `hdl` and $\mathtt{tpk} = I = h_1^f$, to the host.
2. The TPM and the host generate a registration-request signature σ_m on a nonce $n_m \leftarrow_\$ \{0,1\}^\lambda$ to present the signature proof of knowledge (SPK) of `tsk` along with m_s parameters $u'_1, u'_2, \cdots, u'_{m_s} \leftarrow_\$ \mathbb{Z}_p^*$. The σ_m is given as

$$\sigma_m \leftarrow SPK\left\{(f, u'_1, u'_2, \cdots, u'_{m_s}) : U_1 = h_1^f \cdot h_2^{u'_1}, \right. \\ \left. U_2 = h_1^f \cdot h_2^{u'_2}, \cdots, U_{m_s} = h_1^f \cdot h_2^{u'_{m_s}}\right\}(n_m). \quad (1)$$

3. The host sends (n_m, σ_m) to the issuer.
4. The issuer verifies the validity of signature σ_m. If the verification fails, the issuer aborts; otherwise, the issuer proceeds as follows.
   (a) For each $j \in [1, m_s]$, select $u''_j, v_j \leftarrow_\$ \mathbb{Z}_p^*$, and compute $J_j = \left(g_1 \cdot U_j \cdot h_2^{u''_j}\right)^{\frac{1}{\gamma + v_j}}$.
   (b) Send $(J_1, u''_1, v_1, \cdots, J_{m_s}, u''_{m_s}, v_{m_s})$ to the host.

5. The host performs the following.

   (a) For each $j \in [1, m_s]$, compute $u_j = u'_j + u''_j$, and set a membership credential $\mathtt{memCre_j} = (J_j, u_j, v_j)$. Note that (J_j, u_j, v_j) is a BBS+ signature on f.

   (b) Output $\mathtt{mcl} = (\mathtt{memCre}_1, \cdots, \mathtt{memCre}_{m_s})$.

6.3 Acquisition of Login Credential

We assume that the credential token list is represented as $\mathtt{ctl} = \{(K_i, y_i) : K_i \in \mathbb{G}_1, y_i \in \mathbb{Z}_p^*, \forall i \in [1, m_t m_s]\}$, where m_t represents the number of platforms in the network.

$(\mathtt{ctl}', \mathtt{logCre_j}) \leftarrow \mathsf{LogCreGen}(\mathtt{gpk}, \mathtt{isk}, \mathtt{ctl}, \mathtt{tsk}, \mathtt{hdl}, \mathtt{tpk}, \mathtt{memCre_j})$

This protocol is performed among the TPM, the host and the issuer. The inputs to the issuer are $\mathtt{gpk}$, $\mathtt{isk}$ and $\mathtt{ctl}$, the inputs to the TPM are $\mathtt{gpk}$ and $\mathtt{tsk}$, and the inputs to the host are $\mathtt{gpk}$, $\mathtt{hdl}$, $\mathtt{tpk}$ and $\mathtt{memCre_j}$. In this protocol, the issuer outputs the updated list $\mathtt{ctl}'$, and the host outputs a login credential $\mathtt{logCre_j}$. This protocol proceeds as follows.

1. The host and the TPM generate a login credential-request signature σ_g on a nonce $n_g \leftarrow_{\$} \{0,1\}^{\lambda}$ to present the SPK of $\mathtt{tsk}$, u_j, a parameter $x_j \leftarrow_{\$} \mathbb{Z}_p^*$, and a BBS+ signature (J_j, u_j, v_j) on f. The σ_g is given as

$$\sigma_g \leftarrow SPK\ \{(f, u_j, v_j, x_j) : C_g = B_g^f,\ K_j = g_1^{u_j},\ L_j = h_1^f \cdot h_2^{x_j},\ e(J_j, \omega \cdot g_2^{v_j}) = e(g_1 \cdot h_1^f \cdot h_2^{u_j}, g_2)\}(n_g). \quad (2)$$

2. The host sends (n_g, σ_g) to the issuer.

3. The issuer verifies: (1) whether the signature is honestly generated, and (2) whether the membership credential has not been utilized previously to acquire a login credential. If both the verification steps (as shown below) are successful, the issuer proceeds; otherwise it aborts.

   (a) Verify the validity of σ_g.

   (b) For the entries corresponding to K_i in the list $\mathtt{ctl}$, verify that $K_j \notin \mathtt{ctl}$ by utilizing a conventional binary search algorithm.

4. The issuer proceeds as follows.

   (a) Select $y_j, z_j \leftarrow_{\$} \mathbb{Z}_p^*$; and compute $A_j = \left(g_1 \cdot L_j \cdot h_3^{y_j}\right)^{\frac{1}{\gamma + z_j}}$.

   (b) Append an entry of the tuple (K_j, y_j) to the list $\mathtt{ctl}$, and output the updated list $\mathtt{ctl}'$.

   (c) Send (A_j, y_j, z_j) to the host.

5. The host outputs the login credential, $\mathtt{logCre_j} = (A_j, x_j, y_j, z_j)$. Note that (A_j, y_j, z_j) is a BBS+ signature on (f, x_j).

6.4 Selection of Login Credential

After running the LogCreGen protocol for m_s number of times, the host sets the login credential list $\mathtt{lcl} = (\mathtt{logCre}_1, \cdots, \mathtt{logCre}_{\mathtt{m_s}})$. Each entry in the credential-usage list cul takes one of the three values, i.e., $\mathtt{cul_j} \in \{\mathtt{unUsed}, \mathtt{absUsed}, \mathtt{conUsed}\}$, corresponding to the cases where $\mathtt{logCre_j}$ has never been utilized, has already been utilized to generate an absolutely unlinkable signature, or has already been utilized to generate one or more conditionally unlinkable signatures, respectively. The credential-selection rule is given as $\mathtt{csr} \in \{\mathtt{absUnlink}, \mathtt{conUnlink}\}$.

$(\mathtt{logCre_j}, \mathtt{cul}') \leftarrow \mathsf{SelectLogCre}(\mathtt{lcl}, \mathtt{cul}, \mathtt{csr})$

This selection algorithm is performed by the host. The inputs to this algorithm are lcl, cul, and csr. This algorithm outputs a login credential $\mathtt{logCre_j}$ and an updated list cul′ as follows.

1. If $\mathtt{csr} = \mathtt{absUnlink}$, select $j \in [1, m_s]$ such that $\mathtt{cul_j} = \mathtt{unUsed}$, and set $\mathtt{cul_j} = \mathtt{absUsed}$; otherwise, if $\mathtt{csr} = \mathtt{conUnlink}$, select $j \in [1, m_s]$ such that $\mathtt{cul_j} = \mathtt{unUsed}$ or $\mathtt{cul_j} = \mathtt{conUsed}$, and set $\mathtt{cul_j} = \mathtt{conUsed}$.
2. Select the login credential $\mathtt{logCre_j}$ from lcl.
3. Output $\mathtt{logCre_j}$, and the updated list cul′.

6.5 Login Signature Generation

$\sigma_s \leftarrow \mathsf{Sign}(\mathtt{gpk}, \mathtt{tsk}, \mathtt{hdl}, \mathtt{tpk}, \mathtt{logCre_j}, M)$

This login signature generation protocol is performed between the TPM and the host. The inputs to the TPM are gpk and tsk, and the inputs to the host are gpk, hdl, tpk, $\mathtt{logCre_j}$ and a login request message M. This protocol outputs a login signature σ_s which presents the SPK of tsk, a valid revocation token y_j, and a BBS+ signature (A_j, y_j, z_j) on (f, x_j). The σ_s is given as

$$\sigma_s \leftarrow SPK\ \{(f, x_j, y_j, z_j) : C_s = B_s^f,\ E_s = D_s^{y_j},\ e(A_j, \omega \cdot g_2^{z_j}) = e(g_1 \cdot h_1^f \cdot h_2^{x_j} \cdot h_3^{y_j}, g_2)\}(M). \quad (3)$$

6.6 Login Signature Verification

We assume that the token-based revocation list tRL is represented as $\mathtt{tRL} = \{y_i : y_i \in \mathbb{Z}_p^*, \forall i \in [1, m_r]\}$, where m_r is the number of revoked login credentials. The key-based revocation list kRL is represented as $\mathtt{kRL} = \{f_i : f_i \in \mathbb{Z}_p^*, \forall i \in [1, m_k]\}$, where m_k is the number of revoked TPM's secret keys.

$\mathtt{valid/invalid} \leftarrow \mathsf{Verify}(\mathtt{gpk}, \sigma_s, M, \mathtt{tRL}, \mathtt{kRL})$

This verification algorithm takes gpk, a purported login signature σ_s, a login request message M, tRL, and kRL as inputs. This algorithm verifies: (1) whether the signature is honestly generated, (2) whether the login credential used to generate the signature is not revoked, and (3) whether the TPM's secret key

is not revoked. If all of these three verification steps (as shown below) are successful, this algorithm outputs the value `valid`; otherwise it outputs the value `invalid`.

1. Verify the validity of the signature σ_s.
2. For each entry of y_i in `tRL`, compute $E_i = D_s^{y_i}$, and verify that $E_i \neq E_s$.
3. For each entry of f_i in `kRL`, compute $C_i = B_s^{f_i}$, and verify that $C_i \neq C_s$.

6.7 Revocation

The revocation algorithm Revoke comprises of following three sub-algorithms, and only one of these sub-algorithms is run based on the available inputs.

$\texttt{tRL}' \leftarrow \mathsf{RevokeSign}(\texttt{gpk}, \texttt{ctl}, \sigma_s, M, \texttt{tRL}, \texttt{kRL})$

The inputs to this signature-based revocation sub-algorithm are `gpk`, `ctl`, a signature σ_s, a message M, `tRL` and `kRL`. This sub-algorithm outputs the updated revocation list, $\texttt{tRL}'$, using the following steps.

1. Verify that σ_s is an honestly generated signature, i.e., $\texttt{valid} \leftarrow \mathsf{Verify}(\texttt{gpk}, \sigma_s, M, \texttt{tRL}, \texttt{kRL})$. If the verification fails, abort.
2. For each entry of y_i in the list `ctl`, compute $E_i = D_s^{y_i}$, and find the index i such that $E_i = E_s$.
3. Append y_i to `tRL`, and output the updated $\texttt{tRL}'$. This revokes the login credential utilized to generate the signature σ_s.

$\texttt{tRL}' \leftarrow \mathsf{RevokeTpk}(\texttt{gpk}, \texttt{ctl}, \texttt{tpk}, \texttt{cre}, \texttt{tRL})$

The inputs to this TPM's public key-based revocation sub-algorithm are `gpk`, `ctl`, `tpk`, a membership or login credential `cre`, and `tRL`. This sub-algorithm outputs the updated revocation list, $\texttt{tRL}'$, as follows.

1. If `cre` is a membership credential,
   (a) Verify that $e(J_j, \omega \cdot g_2^{v_j}) = e(g_1 \cdot h_1^f \cdot h_2^{u_j}, g_2)$. If the verification fails, abort.
   (b) Compute $K_j = g_1^{u_j}$, and search for the index i in the entries corresponding to K_i in the `ctl` for which $K_i = K_j$ using conventional binary search algorithm.
2. If `cre` is a login credential,
   (a) Verify that $e(A_j, \omega \cdot g_2^{z_j}) = e(g_1 \cdot h_1^f \cdot h_2^{x_j} \cdot h_3^{y_j}, g_2)$. If the verification fails, abort.
   (b) Search for the index i in the entries corresponding to y_i in the `ctl` for which $y_i = y_j$ using conventional binary search algorithm.

3. Append the corresponding y_i from the list `ctl` to the `tRL`. This revokes the corresponding login credential.

$\texttt{kRL}' \leftarrow \mathsf{RevokeTsk}(\texttt{gpk}, \texttt{cre}, \texttt{tsk}, \texttt{kRL})$

The inputs to this TPM's secret key-based revocation sub-algorithm are `gpk`, a membership or login credential `cre`, `tsk`, and `kRL`. This sub-algorithm outputs the updated revocation list, $\texttt{kRL}'$, as follows.

1. If `cre` is a membership credential, verify that $e(J_j, \omega \cdot g_2^{v_j}) = e(g_1 \cdot h_1^f \cdot h_2^{u_j}, g_2)$. If the verification fails, abort.
2. If `cre` is a login credential, verify that $e(A_j, \omega \cdot g_2^{z_j}) = e(g_1 \cdot h_1^f \cdot h_2^{x_j} \cdot h_3^{y_j}, g_2)$. If the verification fails, abort.
3. Append f to the `kRL`, and output the updated $\texttt{kRL}'$. This revokes the `tsk`.

Note that LASER enables practical and flexible revocation which is a desirable characteristic in the context of the subscription system. The three sub-algorithms mentioned above correspond to the following three cases. Firstly, if only one of platform's login credentials is compromised, then the issuer should specifically revoke only the compromised login credential, and should not need to revoke the platform. Hence, in LASER, using the RevokeSign sub-algorithm, the issuer revokes only the login credential utilized to generate the signature σ_s. Since the platform has m_s login credentials, the platform is able to generate valid signatures using other unrevoked login credentials. Secondly, if the host is compromised, the issuer should revoke all the login credentials without knowing the `tsk`. Hence, in LASER, using the RevokeTpk sub-algorithm, the issuer revokes the compromised login credentials which are provided to the issuer by the platform. In this case, if the SU needs to revoke the platform's login credentials after it is stolen, he/she needs to store `tpk` and `mcl` or `lcl` at a device other than the platform, and provide them to the issuer [23]. Thirdly, if the platform is completely compromised and the TPM's secret key is published, the issuer should revoke the platform such that the platform is not able to generate valid signatures using any of the acquired login credentials. Hence, using the RevokeTsk sub-algorithm, the issuer revokes `tsk`. Since the compromise of a login credential is significantly more likely than that of an embedded TPM's secret key, we assume that the length of the token-based revocation list `tRL` is significantly larger than that of the key-based revocation list `kRL`, i.e., $m_r >> m_k$.

Also, note that after the revocation of a login credential, all of the previous signatures associated with the revoked login credential can be linked together by the verifier. This drawback of *backward-linkability* is shared by all the anonymous authentication schemes which support verifier-local revocation [19]. This drawback can be mitigated by utilizing either time-stamped parameters [36] or accumulators [22]. However, discussions on such a countermeasure are beyond the scope of this paper.

6.8 Signature Tracing

$\texttt{true/false} \leftarrow \mathsf{Identify}(\texttt{gpk}, \sigma_s, M, \texttt{tsk}_*)$

This algorithm takes gpk, a signature σ_s, a message M, and a TPM's secret key $\texttt{tsk}_*$ as inputs. Through the following steps, this algorithm outputs the value true if σ_s is proved to have been generated using $\texttt{tsk}_* = f_*$; otherwise it outputs the value false.

1. Verify that σ_s is an honestly generated signature, i.e., $\texttt{valid} \leftarrow \mathsf{Verify}(\texttt{gpk}, \sigma_s, M, \varnothing, \varnothing)$. If the verification fails, output the value false.

2. Compute $C_* = B_s^{f_*}$, and verify that $C_s = C_*$. If the verification succeeds, output the value true; otherwise output the value false.

7 Analytical Evaluation

In this section, we analytically evaluate the computational complexity and communication overhead of LASER, and compare LASER's performance with the ECC-based DAA scheme proposed by Camenisch, Drijvers and Lehmann (CDL-EPID) [13]. We utilize the same technique presented in [13] to generate the SPKs presented in equations (1), (2) and (3). We select CDL-EPID with no attributes as the benchmark scheme because similar to LASER, CDL-EPID: (1) employs the notion of signature-based revocation, (2) splits the workload at the platform between the TPM and the host, and (3) is traceable under the q-SDH assumption. We do not consider the data storage requirements since the storage capacity is assumed to be abundantly available at the host, the verifier and the issuer, and the storage requirement is assumed to be the same at the TPM for both the DAA schemes.

In the following discussion, we consider only the computationally expensive operations—i.e., exponentiation in $\mathbb{G}_1$ and bilinear mapping. The time taken to perform all other operations, e.g., multiplication, addition, inverse, binary search, etc., are significantly smaller than the time taken to compute an exponentiation in $\mathbb{G}_1$, and hence are ignored. We represent the number of exponentiations in $\mathbb{G}_1$ and bilinear mappings by E_{G_1} and B_M, respectively. Also, let the number of elements in $\mathbb{G}_1$ and $\mathbb{Z}_p^*$ communicated between the entities be represented by L_{G_1} and L_{Zp}, respectively.

7.1 Computational Complexity

In a DAA scheme, we divide the operations at the TPM, the host, the verifier, and the issuer into two classes—(1) offline, and (2) online. All the operations which can be pre-computed or stored, and do not need to be generated in real time are classified as offline operations. The offline operations include the computations at the TPM, the host and the issuer for establishing the platform's

Table 1: Comparison of the number of the offline and online computational operations in the DAA schemes.

		CDL-EPID		LASER	
		E_{G_1}	B_M	E_{G_1}	B_M
offline	TPM	2	0	$2 + 3m_s$	0
	Host	0	0	$16m_s$	0
	Issuer	4	0	$21m_s$	$2m_s$
online	TPM	$3 + 3m_r$	0	3	0
	Host	$8 + 5m_r$	0	14	0
	Verifier	$10 + 5m_r$	2	$12 + m_r$	2

Table 2: Comparison of the number of elements in the offline and online communication in the DAA schemes.

		CDL-EPID		LASER	
		L_{G_1}	L_{Z_p}	L_{G_1}	L_{Z_p}
offline	*p-i-sig*	1	3	$8m_s$	$3 + 10m_s$
	i-p-cre	1	2	$2m_s$	$4m_s$
	i-p-rev	$2m_r$	0	0	0
	i-v-rev	$2m_r$	0	0	m_r
online	*p-v-sig*	$5 + m_r$	$7 + 4m_r$	7	8

membership and/or login credentials. The operations which need to be performed in real time are classified as online operations. The online operations include the computations at the TPM and the host for generating the login signature, and the computations at the verifier for verifying the signature.

Table 1 presents the number of computationally expensive offline and online operations performed by each entity in the two DAA schemes. In LASER, the total offline computational complexities are computed by summing the computational complexities in the MemCreGen and the LogCreGen protocols. In Table 1, we observe that the computational complexities of the offline operations in CDL-EPID and LASER are $\mathcal{O}(1)$ and $\mathcal{O}(m_s)$, respectively. Most importantly, in Table 1, we observe that the computational complexities of the platform's online operations are $\mathcal{O}(m_r)$ in CDL-EPID as compared to $\mathcal{O}(1)$ in LASER. Note that the computational complexity of verifier's online operations are $\mathcal{O}(m_r)$ in CDL-EPID as well as LASER. However, in the context of the subscriptions system, we assume that the verifiers (SPs) have access to servers with large computational resources, and hence they are able to handle the large computational cost of the revocation check procedure.

7.2 Communication Overhead

In a DAA scheme, we divide the communications at the platform, the verifier and the issuer into two classes—(1) offline, and (2) online. All the communications which can be pre-shared and stored, and do not need to be performed

in real time are classified as offline communications. The offline communication overhead includes the communication from the platform to the issuer for sending the signatures with requests for the membership and/or login credentials (represented by *p-i-sig*). It also includes the communication from the issuer to the platform for sending the membership and/or login credentials (represented by *i-p-cre*), and the revocation list (represented by *i-p-rev*). Further, it includes the communication from the issuer to the verifier for sending the revocation list (represented by *i-v-rev*). The communications which need to be performed in real time are classified as online communications. The online communication overhead includes the communication between the platform and the verifier for sending and receiving the login signature (represented by *p-v-sig*).

Table 2 presents the number of offline and online elements communicated between the entities in the two DAA schemes. In LASER, the offline communication overheads are computed by summing the communication overheads in the MemCreGen and the LogCreGen protocols. In Table 2, we observe that in CDL-EPID, the sum of the offline communication overheads at the platform and the verifier increase by $\mathcal{O}(m_r)$. In LASER, the sum of the offline communication overheads increases by $\mathcal{O}(m_s)$ at the platform and $\mathcal{O}(m_r)$ at the verifier. Most importantly, in Table 2, we observe that the online communication overhead increases by $\mathcal{O}(m_r)$ in CDL-EPID as compared to $\mathcal{O}(1)$ in LASER.

8 Implementation Results

We present an illustrative application scenario where LASER is employed to realize the ASS. We obtain the results by implementing the application scenario on a laptop platform with an on-board TPM. To the best of our knowledge, this is the *first* implementation and analysis of an anonymous authentication scheme using an actual TPM cryptoprocessor that is compliant with the TPM specification version 2.0. Note that the design of LASER under the framework of the most recent specification also simplifies its adaptation and evaluation in existing networks.

8.1 Illustrative Application Scenario

We assume that one million SUs are subscribed at the issuer in an online subscription service. The SUs are required to renew their subscription every month which consists of 30 days. Each SU (platform) sends login requests to ten SPs (verifiers) per day, and generates one login signature corresponding to the request to each SP per day. This means that the total number of login signatures generated per day by a SU is ten. Moreover, we assume that 0.2 percent of the laptop platforms belonging to the SUs are revoked every month because they are lost or stolen [17]. This means that over the period of a month, the number of revoked platforms increases from 0 to 2000. In this illustrative scenario, we consider and analyze the following four deployment cases of DAA.

1. CDL-EPID: During subscription, each SU obtains the membership credential using the offline procedure. Then, each SU generates absolutely unlinkable signatures for the ten SPs per day through the online procedure.

2. LASER with absolute unlinkability (LASER-A): During subscription with the issuer, through offline procedure, each SU obtains 300 membership credentials and corresponding login credentials (i.e., $m_s = 300$). Using these login credentials, each SU generates absolutely unlinkable signatures for the ten SPs per day through the online procedure.

3. LASER with conditional unlinkability where the issuer can link some login signatures (LASER-B): Through the offline procedure, the SU obtains 30 membership credentials and corresponding login credentials (i.e., $m_s = 30$). Each SU generates conditionally unlinkable signatures for the ten SPs per day using a login credential through the online procedure. Any two signatures generated on the same day remain linkable by the issuer, but any two signatures generated on two different days remain unlinkable by the issuer.

4. LASER with conditional unlinkability where the issuer can link all the login signatures (LASER-C): During subscription, each SU obtains only one membership credential and corresponding login credential (i.e., $m_s = 1$). Each SU generates conditionally unlinkable signatures through the online procedure. Hence, the signatures generated throughout the month are unlinkable by the SPs, but linkable by the issuer.

8.2 Details of the Prototype

We obtain the computational overheads in the above deployment cases by implementing them on a Lenovo laptop with 2.6 GHz Intel i7 6600U CPU. We leverage OpenSSL, the pairing-based cryptography (PBC) library [37], and IBM Trusted Software Stack (TSS) for TPM 2.0 [38] to prototype CDL-EPID and LASER in *C*. The prototypes of the TPM and the host in LASER comprise of 600 and 1800 lines of code, respectively. The total development time of the prototypes is around 1000 man-hours. We utilize the Barreto-Naehrig (BN) curve which is standardized for DAA by the TCG [39]. Specifically, we utilize the "Type F" internal described in PBC library which is constructed on the curve of the form $y^2 = x^3 + 3$ with embedding degree 12 where the lengths of an element in $\mathbb{Z}_p^*$ and $\mathbb{G}_1$ are 256 bits and 512 bits, respectively. With this curve, the DAA provides 128 bits of security which is approximately the same level of security as a symmetric key encryption with a key size of 128 bits, or an RSA signature with a modulus size of 3072 bits.

Note that the TPM2_Sign algorithm in the TPM specification version 2.0 can be utilized as a Diffie-Hellman (DH) oracle which significantly reduces the achievable level of security with the TPM [40]. Recently, multiple proposals have been made to alleviate this vulnerability, but none of these has been accepted by the TCG yet [41, 42]. LASER utilizes the algorithm TPM2_Mod_Sign (discussed in Appendix B.4) to prevent the TPM interface from being used as

```
primaryHandle = primaryHandle;
inSensitive.sensitive.data.t.size = 0;
inSensitive.sensitive.userAuth.t.size = 0;
inPublic.publicArea.type = TPM_ALG_ECC;
inPublic.publicArea.nameAlg = halg;
inPublic.publicArea.objectAttributes.val = 0;
inPublic.publicArea.objectAttributes.val |= TPMA_OBJECT_FIXEDTPM;
inPublic.publicArea.objectAttributes.val |= TPMA_OBJECT_FIXEDPARENT;
inPublic.publicArea.objectAttributes.val |= TPMA_OBJECT_SENSITIVEDATAORIGIN;
inPublic.publicArea.objectAttributes.val |= TPMA_OBJECT_USERWITHAUTH;
inPublic.publicArea.objectAttributes.val &= ~TPMA_OBJECT_ADMINWITHPOLICY;
inPublic.publicArea.objectAttributes.val |= TPMA_OBJECT_NODA;
inPublic.publicArea.objectAttributes.val &= ~TPMA_OBJECT_RESTRICTED;
inPublic.publicArea.objectAttributes.val &= ~TPMA_OBJECT_DECRYPT;
inPublic.publicArea.objectAttributes.val |= TPMA_OBJECT_SIGN;
inPublic.publicArea.authPolicy.t.size = 0;
inPublic.publicArea.parameters.eccDetail.symmetric.algorithm = TPM_ALG_NULL;
inPublic.publicArea.parameters.eccDetail.scheme.scheme = TPM_ALG_ECDAA;
inPublic.publicArea.parameters.eccDetail.curveID = TPM_ECC_BN_P256;
inPublic.publicArea.parameters.eccDetail.kdf.scheme = TPM_ALG_NULL;
inPublic.publicArea.parameters.eccDetail.kdf.details.mgf1.hashAlg = halg;
inPublic.publicArea.parameters.eccDetail.scheme.details.ecdaa.hashAlg = halg;
inPublic.publicArea.unique.ecc.x.t.size = 0;
inPublic.publicArea.unique.ecc.y.t.size = 0;
outsideInfo.t.size = 0;
creationPCR.count = 0;
```

Figure 1: Configuration parameters for TPM2_CreatePrimary.

a static DH oracle. This algorithm is presented in [41], and utilized in the most recent DAA scheme [13]. However, an on-board TPM chip cannot perform this algorithm as it is not part of the specifications and there is no available interface to modify the existing algorithms. Hence, we are forced to utilize the TPM2_Sign algorithm given in the specifications for obtaining the implementation results. This means that although the construction of LASER provides 128 bits of security, its prototype implementation in this paper offers only 85 bits of security [42].

It is quite challenging to work with the TPM algorithms by jumping directly into the TPM standard [43]. Much of this complexity is abstracted away by utilizing the IBM TSS. The TSS is a software package that can be linked to the application to serve as a communication layer between the application and the TPM. This functionality is extremely helpful and is utilized for writing applications that do not require lower-level access to read and write TPM structures. The individual TPM algorithms can be run using the commands from the TSS after setting the appropriate configuration parameters. However, in our experience, the most challenging task of prototyping LASER using TSS was to determine the configuration parameters which were needed for a given TPM command. Due to the absence of any open-source illustration, we needed to delve into the TPM specification version 2.0 to tackle this challenge. Figure 1 presents the configuration parameters utilized for the TPM2_CreatePrimary algorithm which is discussed in Appendix B.2.

Note that running early-development phase codes directly on the TPM may lock the platform or delete important information already stored in the TPM.

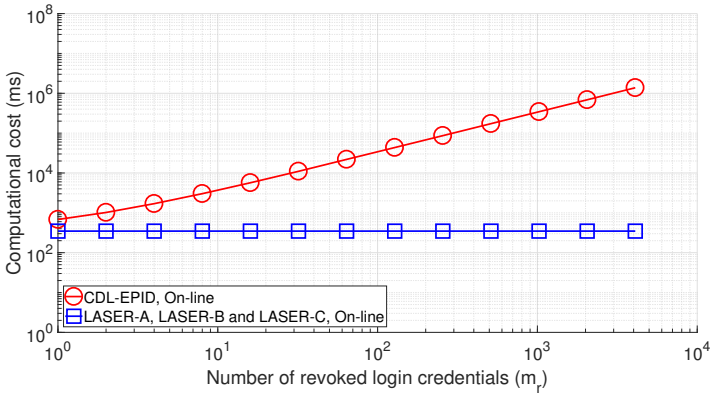

Figure 2: Comparison of the platform's online computational costs in the DAA schemes.

Hence, we utilized TPM 2.0 emulator [44], and developed the initial code with the TSS to correctly interface with the TPM emulator. We optimized the code using the conventional techniques, such as, computing multi-exponentiation in $\mathbb{G}_1$. Further, we needed to modify the BIOS settings to enable the security chip on the laptop platform. We installed and loaded the TPM device driver, and verified that the hardware TPM was functional. Finally, we ported the emulator implementation code to work with the hardware TPM.

8.3 Results

By averaging over 100 iterations, we obtain the running time in milliseconds (ms) for different protocols in LASER and CDL-EPID. Figure 2 presents the curves for the platform's online computational costs in CDL-EPID, LASER-A, LASER-B, and LASER-C vs. m_r. We observe that the online computational cost in CDL-EPID increases linearly with the increase in m_r. However, the online computational costs in LASER-A, LASER-B and LASER-C do not change with m_r. With $m_r = 1000$, the time taken to generate a signature is only 348 ms in LASER-A as compared to 342,112 ms in CDL-EPID. Further, in Table 3, we present the *monthly* computational costs of the offline and online operations in the four cases when the number of revoked platforms increases from 0 to 2000 over the month. In Table 3, we observe that the offline computational costs in LASER-A and LASER-B are significantly higher than that in CDL-EPID, while the offline computational cost in LASER-C is only slightly higher than that in CDL-EPID. From Figure 2 and Table 3, we note that significantly lower online computational cost in LASER-A, LASER-B and LASER-C as compared to CDL-EPID is achieved at the cost of higher offline computational cost. This trade-off between offline and online computational costs is very advantageous because the online operations occur significantly more often than the offline

Table 3: Comparison of the running time (in milliseconds) of the operations in the DAA schemes.

		CDL-EPID	LASER-A	LASER-B	LASER-C
Offline	TPM	749	94,765	10,176	1,091
	Host	11	13,811	1,391	57
	Issuer	24	29,429	3,012	175
Online	TPM	93,637,299	94,008	94,008	94,008
	Host	8,996,412	10,362	10,362	10,362
	Verifier	1,236,690	327,909	327,909	327,909

Table 4: Comparison of the communication overhead (in bits) between the entities in the DAA schemes.

		CDL-EPID	LASER-A	LASER-B	LASER-C
Offline	*p-i-sig*	1,280	1,997,568	200,448	7,414
	i-p-cre	1,024	614,400	61,440	2,048
	i-p-rev	2,048,000	0	0	0
	i-v-rev	2,048,000	512,000	512,000	512,000
Online	*p-v-sig*	462,105,600	1,689,600	1,689,600	1,689,600

operations.

Table 4 presents the monthly online and offline communication overheads in the four cases in the aforementioned scenario. We observe that LASER-B and LASER-C result in the same online communication cost, but significantly lower offline communication costs when compared to LASER-A. We also observe that LASER-A is more than two orders of magnitude more efficient than CDL-EPID in terms of the online communication overhead.

From the above results, we observe that LASER-A incurs significantly lower online overhead - in terms of both computation and communication - compared to CDL-DAA at the cost of higher offline overhead. As the online procedure require significantly lower latency than the offline procedure, LASER-A is more practical than CDL-EPID when m_r is large.

Another noteworthy attribute of LASER is its realization of the novel notion that we refer to as *adaptable unlinkability*. LASER is capable of increasing both the computational and communication efficiency of the underlying DAA protocol by relaxing the notion of absolute unlinkability (which is provided by LASER-A) to realize conditional unlinkability (which is provided by LASER-B and LASER-C). Note that CDL-EPID cannot employ this notion of adaptable unlinkability. The above illustrative scenario also demonstrates that absolute unlinkability (adopted by the prior art) of all of the signatures is not necessary, and the concept of adaptable unlinkability enables LASER to provide the needed privacy attributes in a flexible and practical manner.

9 Conclusion

In this paper, we propose a novel DAA scheme called Lightweight Anonymous Subscription with Efficient Revocation (LASER) which can be utilized to realize the anonymous subscription system. We show that the revocation is the primary performance bottleneck of modern DAA schemes and that existing schemes do not scale well to large networks because of the high computational and communication costs corresponding to their revocation check procedures. By using the novel concept of adaptable unlinkability, LASER manages to significantly reduce the computational and communication burden of the SU platform's online protocol at the cost of increased burden of the SU platform's offline protocol.

References

[1] Ponemon Institute, "Cost of data breach study: Global overview," 2017.

[2] M. Blanton, "Online subscriptions with anonymous access," in *Proceedings of the ACM Symposium on Information, Computer and Communications Security (ASIACCS)*, pp. 217–227, 2008.

[3] M. Z. Lee, A. M. Dunn, B. Waters, E. Witchel, and J. Katz, "Anon-pass: Practical anonymous subscriptions," in *Proceedings of the IEEE Symposium on Security and Privacy (S&P)*, pp. 319–333, 2013.

[4] N. Komninos and A. K. Junejo, "Privacy preserving attribute based encryption for multiple cloud collaborative environment," in *IEEE/ACM 8th International Conference on Utility and Cloud Computing (UCC)*, pp. 595–600, 2015.

[5] T. Gupta, N. Crooks, W. Mulhern, S. Setty, L. Alvisi, and M. Walfish, "Scalable and private media consumption with popcorn," in *Proceedings of the 13th Usenix Conference on Networked Systems Design and Implementation (NSDI)*, pp. 91–107, 2016.

[6] E. Brickell, J. Camenisch, and L. Chen, "Direct anonymous attestation," in *Proceedings of the 11th ACM Conference on Computer and Communications Security (CCS)*, pp. 132–145, 2004.

[7] Trusted Computing Group, "TPM library specification," 2014.

[8] E. Brickell and J. Li, "Enhanced privacy ID from bilinear pairing for hardware authentication and attestation," in *IEEE Second International Conference on Social Computing (SocialCom)*, pp. 768–775, 2010.

[9] L. Chen, D. Page, and N. P. Smart, "On the design and implementation of an efficient DAA scheme," in *International Conference on Smart Card Research and Advanced Application*, pp. 223–237, 2010.

[10] ISO/IEC, "ISO/IEC 11889:2015, parts 1-4," 2015.

[11] G. AlLee, "EPID for IoT security," 2016.

[12] E. Brickell and J. Li, "Enhanced privacy ID: A direct anonymous attestation scheme with enhanced revocation capabilities," *IEEE Transactions on Dependable and Secure Computing*, vol. 9, no. 3, pp. 345–360, 2012.

[13] J. Camenisch, M. Drijvers, and A. Lehmann, "Anonymous attestation using the strong Diffie Hellman assumption revisited." Cryptology ePrint Archive, Report 2016/663, 2016.

[14] V. Costan and S. Devadas, "Intel SGX explained,"

[15] L. Chen and J. Li, "Revocation of direct anonymous attestation," in *International Conference on Trusted Systems*, pp. 128–147, 2010.

[16] Y. Liu, W. Tome, L. Zhang, D. Choffnes, D. Levin, B. Maggs, A. Mislove, A. Schulman, and C. Wilson, "An end-to-end measurement of certificate revocation in the web's PKI," in *Proceedings of the Internet Measurement Conference (IMC)*, pp. 183–196, 2015.

[17] Ponemon Institute, "The billion dollar lost laptop problem: Benchmark study of U.S. organizations," 2010.

[18] M. Nemec, M. Sys, P. Svenda, D. Klinec, and V. Matyas, "The return of coppersmith's attack: Practical factorization of widely used RSA moduli," in *Proceedings of the ACM SIGSAC Conference on Computer and Communications Security (CCS)*, pp. 1631–1648, 2017.

[19] D. Boneh and H. Shacham, "Group signatures with verifier-local revocation," in *Proceedings of the 11th ACM Conference on Computer and Communications Security (CCS)*, pp. 168–177, 2004.

[20] E. Brickell and J. Li, "A pairing-based DAA scheme further reducing TPM resources," in *International Conference on Trust and Trustworthy Computing (TRUST)*, pp. 181–195, 2010.

[21] L. Chen and R. Urian, "DAA-A: Direct anonymous attestation with attributes," in *International Conference on Trust and Trustworthy Computing (TRUST)*, pp. 228–245, 2015.

[22] M. H. Au and A. Kapadia, "PERM: Practical reputation-based blacklisting without TTPs," in *Proceedings of the ACM Conference on Computer and Communications Security (CCS)*, pp. 929–940, 2012.

[23] W. Lueks, G. Alpar, J.-H. Hoepman, and P. Vullers, "Fast revocation of attribute-based credentials for both users and verifiers," *Computers & Security*, vol. 67, pp. 308–323, 2017.

[24] L. Xi, D. Feng, Y. Qin, F. Wei, J. Shao, and B. Yang, "Direct anonymous attestation in practice: Implementation and efficient revocation," in *Twelfth Annual International Conference on Privacy, Security and Trust (PST)*, pp. 67–74, 2014.

[25] L. Chen and J. Li, "Flexible and scalable digital signatures in TPM 2.0," in *Proceedings of the ACM SIGSAC Conference on Computer and Communications Security (CCS)*, pp. 37–48, 2013.

[26] D. Bernhard, G. Fuchsbauer, E. Ghadafi, N. P. Smart, and B. Warinschi, "Anonymous attestation with user-controlled linkability," *International Journal of Information Security*, vol. 12, no. 3, pp. 219–249, 2013.

[27] J. Camenisch, M. Drijvers, and A. Lehmann, "Universally composable direct anonymous attestation," in *IACR International Workshop on Public Key Cryptography*, pp. 234–264, 2016.

[28] R. Dingledine, N. Mathewson, and P. Syverson, "Tor: The second-generation onion router," in *Proceedings of the 13th Conference on USENIX Security Symposium - Volume 13*, pp. 21–21, 2004.

[29] B. Chor, O. Goldreich, E. Kushilevitz, and M. Sudan, "Private information retrieval," in *Proceedings of IEEE 36th Annual Foundations of Computer Science*, pp. 41–50, 1995.

[30] S. D. Galbraith, K. G. Paterson, and N. P. Smart, "Pairings for cryptographers," *Discrete Applied Mathematics*, vol. 156, no. 16, pp. 3113–3121, 2008.

[31] D. Boneh, X. Boyen, and H. Shacham, "Short group signatures," in *Advances in Cryptology - CRYPTO*, pp. 41–55, 2004.

[32] J. Camenisch and A. Lysyanskaya, "Signature schemes and anonymous credentials from bilinear maps," in *Advances in Cryptology - CRYPTO*, pp. 56–72, 2004.

[33] M. H. Au, W. Susilo, and Y. Mu, "Constant-size dynamic k-TAA," in *Proceedings of the 5th International Conference on Security and Cryptography for Networks (SCN)*, pp. 111–125, 2006.

[34] D. Boneh, "The decision Diffie-Hellman problem," in *Proceedings of the Third International Symposium on Algorithmic Number Theory*, pp. 48–63, 1998.

[35] D. Boneh and X. Boyen, "Short signatures without random oracles and the SDH assumption in bilinear groups," *Journal of Cryptology*, vol. 21, no. 2, pp. 149–177, 2008.

[36] T. Nakanishi and N. Funabiki, "A short verifier-local revocation group signature scheme with backward unlinkability," in *Advances in Information and Computer Security*, vol. 4266, pp. 17–32, 2006.

[37] B. Lynn, "PBC: Pairing-based cryptography," 2017.

[38] K. Goldman, "IBM TPM 2.0 TSS," 2017.

[39] P. S. L. M. Barreto and M. Naehrig, "Pairing-friendly elliptic curves of prime order," in *International Workshop on Selected Areas in Cryptography*, pp. 319–331, 2005.

[40] T. Acar, L. Nguyen, and G. Zaverucha, "A TPM diffie-hellman oracle," *IACR Cryptology ePrint Archive*, p. 667, 2013.

[41] L. Xi, K. Yang, Z. Zhang, and D. Feng, "DAA-related APIs in TPM 2.0 revisited," in *International Conference on Trust and Trustworthy Computing (TRUST)*, pp. 1–18, 2014.

[42] J. Camenisch, L. Chen, M. Drijvers, A. Lehmann, D. Novick, and R. Urian, "One TPM to bind them all: Fixing TPM 2.0 for provably secure anonymous attestation," in *Proceedings of 38th IEEE Symposium on Security and Privacy (S&P)*, 2017. to appear.

[43] W. Arthur and D. Challener, *A Practical Guide to TPM 2.0: Using the Trusted Platform Module in the New Age of Security*. Apress, 1st ed., 2015.

[44] K. Goldman, "IBM TPM 2.0 emulator," 2017.

A BBS+ Signature

The BBS+ signature is defined as follows [31, 32, 33].

1. $(\mathtt{pk}, \mathtt{sk}) \leftarrow \mathsf{BBS\_Setup}(1^\lambda)$: The input to this algorithm is the security parameter 1^λ. This algorithm selects $g_1, h_1, h_2, h_3 \leftarrow_\$ \mathbb{G}_1$, $g_2 \leftarrow_\$ \mathbb{G}_2$, $\gamma \leftarrow_\$ \mathbb{Z}_p^*$; and computes $\omega = g_2^\gamma$. This algorithm sets the public key $\mathtt{pk} = (g_1, h_1, h_2, h_3, g_2, \omega)$, and the secret key $\mathtt{sk} = \gamma$. It outputs $(\mathtt{pk}, \mathtt{sk})$.

2. $\sigma_b \leftarrow \mathsf{BBS\_Sign}(\mathtt{pk}, \mathtt{sk}, f, x)$: The inputs to this algorithm are $\mathtt{pk}$, $\mathtt{sk}$, and two messages $f, x \in \mathbb{Z}_p^*$. This algorithm selects $y, z \leftarrow_\$ \mathbb{Z}_p^*$, and computes $A = (g_1 \cdot h_1^f \cdot h_2^x \cdot h_3^y)^{1/(\gamma+z)}$. It sets the signature $\sigma_b = (A, y, z)$, and outputs σ_b.

3. $\mathtt{valid}/\mathtt{invalid} \leftarrow \mathsf{BBS\_Verify}(\mathtt{pk}, \sigma_b, f, x)$: The inputs to this algorithm are $\mathtt{pk}$, a signature σ_b, and two messages f and x. This algorithm verifies that $e(A, \omega \cdot g_2^z) = e(g_1 \cdot h_1^f \cdot h_2^x \cdot h_3^y, g_2)$. If the verification succeeds, this algorithm outputs the value $\mathtt{valid}$; otherwise, it outputs the value $\mathtt{invalid}$.

B TPM: Functions and Algorithms

We present the functions and algorithms which are related to DAA and are included in the TPM specification version 2.0 [25].

B.1 Group Mapping

With the hash function $\mathcal{H}_z$, a group mapping function $\mathcal{H}_g : \mathbb{Z}_p^* \rightarrow \mathbb{Z}_p^* \times \mathbb{G}_1$, is constructed as follows. We assume that the group $\mathbb{G}_1$ is constructed using the elliptic curve $y^2 = x^3 + a_c x + b_c$, where a_c and b_c are constant parameters, and the tuple (x, y) represents an element in $\mathbb{G}_1$. Here, $x, y, a_c, b_c \in \mathbb{Z}_p^*$. With an input $a_0 \in \mathbb{Z}_p^*$, the function $\mathcal{H}_g$ outputs a tuple (a_1, b_1, b_2), where $a_1 \in \mathbb{Z}_p^*$, and $(b_1, b_2) \in \mathbb{G}_1$, as follows.

1. Set $i = 0$, where i is an integer.
2. Compute $a_1 = \mathcal{H}_z(a_0 \parallel i)$, $x = \mathcal{H}_z(a_1)$, and $z = x^3 + a_c x + b_c$. Here, $\parallel$ represents concatenation of two strings of bits.
3. Compute $y = \sqrt{z}$. If y does not exist, set $i = i + 1$, and start back from Step (2).
4. Set $b_1 = x$ and $b_2 = y$, and output (a_1, b_1, b_2).

B.2 Key Generation

We assume that the TPM has an internal secret value `DAAseed`, and an internal counter value `cnt`.

$(\texttt{hdl}, \texttt{tpk}) \leftarrow \mathsf{TPM2\_CreatePrimary}(P_I)$: Given $P_I \in \{0,1\}^*$, this algorithm performs the following steps.

1. Compute $f = \mathcal{H}_z(\texttt{DAAseed} \parallel \texttt{cnt} \parallel P_I)$.
2. Set the TPM's secret key $\texttt{tsk} = f$, store `tsk`, and generate a key handle `hdl` which specifies its location in the secure memory of the TPM.
3. Compute $I = h_1^f$, set $\texttt{tpk} = I$, and output $(\texttt{hdl}, \texttt{tpk})$.

B.3 Commitment Generation

$(\texttt{ctr}, C, S_1, S_2) \leftarrow \mathsf{TPM2\_Commit}(\texttt{hdl}, a_1, b_2, h_1)$: With the key handle `hdl`, $a_1, b_2 \in \mathbb{Z}_p^*$, and $h_1 \in \mathbb{G}_1$, as inputs, this algorithm performs the following.

1. Retrieve $\texttt{tsk} = f$ using the key handle `hdl`.
2. Select $r_f \leftarrow_{\$} \mathbb{Z}_p^*$.
3. If $a_1 = \varnothing$, set $C = \varnothing$ and $S_1 = \varnothing$; otherwise, perform the following.
   (a) Compute $b_1 = \mathcal{H}_z(a_1)$, and set $B = (b_1, b_2)$.
   (b) Compute $C = B^f$, and $S_1 = B^{r_f}$.
4. If $h_1 = \varnothing$, set $S_2 = \varnothing$; otherwise, compute $S_2 = h_1^{r_f}$.
5. Store r_f, and generate a counter value `ctr` which specifies its location in the secure memory of the TPM.
6. Output $(\texttt{ctr}, C, S_1, S_2)$.

B.4 Signature Generation

$(n_t, c_t, s_f) \leftarrow \mathsf{TPM2\_Mod\_Sign}(\mathtt{hdl}, \mathtt{ctr}, c_h, M)$: With the key handle $\mathtt{hdl}$, the counter value $\mathtt{ctr}$, $c_h \in \mathbb{Z}_p^*$, and a message $M \in \{0,1\}^*$ as inputs, this algorithm performs the following.

1. Retrieve the TPM's secret key $\mathtt{tsk} = f$ using the key handle $\mathtt{hdl}$, and r_f using the counter value $\mathtt{ctr}$.
2. Select a nonce $n_t \leftarrow_{\$} \{0,1\}^{\lambda}$, compute $c_t = \mathcal{H}_z(c_h \parallel n_t \parallel M)$ and $s_f = r_f + c_t \cdot f$, and output (n_t, c_t, s_f).

C Detailed Proofs of Security

In this section, we provide the theorems corresponding to the security properties discussed in Section 4, and provide comprehensive proofs for them.

Theorem 1. *LASER satisfies the correctness property.*

Proof. This follows from the specifications of the scheme. □

Lemma 2. *In the random oracle model, LASER with the credential-selection rule* `absUnlink` *is absolutely unlinkable under the* $\mathbb{G}_1$*-DDH assumption.*

Proof. Let there be an adversary $\mathcal{A}$ which succeeds to break the absolute unlinkability property of LASER with the credential-selection rule `absUnlink` with a non-negligible probability. Given the $\mathbb{G}_1$-DDH instance, a simulator $\mathcal{S}$ intends to find out whether $c = a \cdot b$, or $c \leftarrow_{\$} \mathbb{Z}_p^*$. To achieve this, $\mathcal{S}$ sets up the following absolute unlinkability game to interact with $\mathcal{A}$.

1. *Setup*: $\mathcal{S}$ simulates the Setup algorithm, sets $h_1 = P$, and sends $\mathtt{isk}$ and $\mathtt{gpk}$ to $\mathcal{A}$. $\mathcal{S}$ creates m_t platforms with identities $\mathcal{P}_i$, $\forall i \in [1, m_t]$, and selects a specific $\mathcal{P}_{i^*}$. Further, $\mathcal{S}$ initializes a list of the corrupted platforms, $\mathtt{cpl} = \varnothing$, and a list of corrupted credentials, $\mathtt{ccl} = \varnothing$.
2. *MemCreGen*: $\mathcal{S}$ which acts as a platform, simulates $\mathsf{MemCreGen}$ protocol with $\mathcal{A}$ which acts as the issuer. For each $i \in [1, m_t]$, $\mathcal{S}$ acquires the membership credentials $\mathtt{memCre}_{\mathtt{ij}}$, $\forall j \in [1, m_s]$ by generating the σ_m differently for the following two cases.
   (a) If $i \neq i^*$, $\mathcal{S}$ generates the secret key $\mathtt{tsk}_{\mathtt{i}}$, and the corresponding signature σ_m.
   (b) If $i = i^*$, $\mathcal{S}$ sets $\mathtt{tpk}_* = I_* = P^a$, and simulates the signature σ_m. $\mathcal{S}$ backpatches the hash oracle to preserve consistency.
3. *LogCreGen*: $\mathcal{S}$ acts as the platform, and simulates the $\mathsf{LogCreGen}$ protocol with $\mathcal{A}$ which acts as the issuer. For each $i \in [1, m_t]$, and $j \in [1, m_s]$, $\mathcal{S}$ acquires a login credential $\mathtt{logCre}_{\mathtt{ij}}$ by generating the signature σ_g differently for the following two cases.

(a) If $i \neq i^*$, $\mathcal{S}$ utilizes $\mathtt{tsk_i}$ and generates signature σ_g.

(b) If $i = i^*$, $\mathcal{S}$ selects $\theta_g \leftarrow_\$ \mathbb{Z}_p^*$, computes $B_g = P^{\theta_g}$ and $C_g = (P^a)^{\theta_g}$, and simulates all other elements of σ_g.

4. *Queries-Phase I*: $\mathcal{A}$ queries $\mathcal{S}$ as follows.

   (a) *Sign*: $\mathcal{A}$ requests $\mathcal{S}$ to generate a signature σ_s on a message M for $\mathcal{P}_i$, and send σ_s. To respond to this query, $\mathcal{S}$ runs the SelectLogCre algorithm with a random credential-selection rule. If $\mathcal{S}$ utilizes `absUnlink`, it appends $\mathtt{logCre_{ij}}$ to `ccl`. Further, $\mathcal{S}$ simulates the Sign protocol differently for the following two cases.

      i. If $i \neq i^*$, $\mathcal{S}$ utilizes $\mathtt{tsk_i}$ to generate signature σ_s.

      ii. If $i = i^*$, $\mathcal{S}$ selects $\theta_s \leftarrow_\$ \mathbb{Z}_p^*$, computes $B_s = P^{\theta_s}$ and $C_s = (P^a)^{\theta_s}$, and simulates all other elements of σ_s.

   (b) *TskCorrupt*: $\mathcal{A}$ queries $\mathcal{S}$ for the TPM's secret key of $\mathcal{P}_i$. If $i \neq i^*$, $\mathcal{S}$ responds to $\mathcal{A}$ with $\mathtt{tsk_i}$, and appends $\mathcal{P}_i$ to `cpl`. Otherwise, if $i = i^*$, $\mathcal{S}$ aborts the game.

   (c) *MemCreCorrupt*: $\mathcal{A}$ requests the j^{th} membership credential of platform $\mathcal{P}_i$. $\mathcal{S}$ responds to $\mathcal{A}$ with $\mathtt{memCre_{ij}}$, and appends the corresponding $\mathtt{logCre_{ij}}$ to `ccl`.

   (d) *LogCreCorrupt*: $\mathcal{A}$ requests the j^{th} login credential of platform $\mathcal{P}_i$. $\mathcal{S}$ responds to $\mathcal{A}$ with $\mathtt{logCre_{ij}}$, and appends $\mathtt{logCre_{ij}}$ to `ccl`.

5. *Challenge*: $\mathcal{A}$ submits a message M, and two platforms $\mathcal{P}_{i_0}$ and $\mathcal{P}_{i_1}$ to $\mathcal{S}$ with the restriction that $\mathcal{P}_{i_0}, \mathcal{P}_{i_1} \notin$ `cpl`. If $\mathcal{P}_{i^*} \notin \{\mathcal{P}_{i_0}, \mathcal{P}_{i_1}\}$, then $\mathcal{S}$ aborts the game. Otherwise, $\mathcal{S}$ picks $\phi \in \{0, 1\}$ such that $\mathcal{P}_{i^*} = \mathcal{P}_{i_\phi}$. $\mathcal{S}$ selects $\mathtt{logCre_{i_\phi j_\phi}}$ using the SelectLogCre algorithm with the rule `absUnlink` such that $\mathtt{logCre_{i_\phi j_\phi}} \notin$ `ccl`. Further, in the Sign protocol, $\mathcal{S}$ selects $\theta_s \leftarrow_\$ \mathbb{Z}_p^*$, computes $B_s = (P^b)^{\theta_s}$ and $C_s = (P^c)^{\theta_s}$, and simulates all other elements of σ_s. Finally, $\mathcal{S}$ sends the generated σ_s to $\mathcal{A}$.

6. *Queries-Phase II (Restricted Queries)*: After obtaining the challenge, $\mathcal{A}$ continues to probe $\mathcal{S}$ with the queries mentioned in *Queries-Phase I*, except for the *TskCorrupt* queries for $\mathtt{tsk_{i_0}}$ and $\mathtt{tsk_{i_1}}$, *MemCreCorrupt* queries for $\mathtt{memCre_{i_0 j_0}}$ and $\mathtt{memCre_{i_1 j_1}}$, and *LogCreCorrupt* queries for $\mathtt{logCre_{i_0 j_0}}$ and $\mathtt{logCre_{i_1 j_1}}$.

7. *Output*: $\mathcal{A}$ outputs $\phi' \in \{0, 1\}$ as the guess for ϕ, or aborts. If $\phi = \phi'$, then $\mathcal{S}$ outputs `true`, which means that $c = a \cdot b$; otherwise, $\mathcal{S}$ outputs `false`, which means that $c \leftarrow_\$ \mathbb{Z}_p^*$.

In the above framework, $\mathcal{S}$ does not abort the game with a non-negligible probability which is represented by $\epsilon_{\mathcal{S}}^{abs}$. Let $Adv_{\mathcal{A}}^{abs}$ be the non-negligible advantage that $\mathcal{A}$ succeeds in breaking the absolute unlinkability property of LASER with the rule `absUnlink`, given that $\mathcal{S}$ does not abort during the above

game. On one hand, $\mathcal{S}$ generates the signatures in the *MemCreGen*, *LogCreGen* and *Queries* phases by assuming the TPM's secret key of the platform $\mathcal{P}_{i^*}$ as $\mathtt{tsk}_* = a$. On the other hand, $\mathcal{S}$ generates the signatures in the *Challenge* phase by assuming the TPM's secret key of the platform $\mathcal{P}_{i^*}$ as $\mathtt{tsk}_* = c/b$. Hence, if $c = a \cdot b$, then $\mathcal{S}$ simulates the game successfully, and $\mathcal{A}$ succeeds in breaking the absolute unlinkability property of LASER with advantage $Adv_{\mathcal{A}}^{abs}$. In the other case, if $c \leftarrow_{\$} \mathbb{Z}_p^*$, then $\mathcal{S}$ cannot simulate the game successfully, and $\mathcal{A}$ either aborts the game or does not win with any non-negligible advantage. Therefore, $\mathcal{S}$ has a non-negligible probability of at least $\epsilon_{\mathcal{S}}^{abs} \cdot Adv_{\mathcal{A}}^{abs}/2$ in breaking the $\mathbb{G}_1$-DDH assumption. □

Lemma 3. *In the random oracle model, under $\mathbb{G}_1$-DDH assumption, LASER with the credential-selection rule* `conUnlink` *is conditionally unlinkable against any adversary which does not know* `isk`.

Proof. Let there be an adversary $\mathcal{A}$ which does not know `isk`, but succeeds to break the conditional unlinkability property of LASER with the rule `conUnlink` with a non-negligible probability. Given the $\mathbb{G}_1$-DDH instance, a simulator $\mathcal{S}$ intends to find out whether $c = a \cdot b$, or $c \leftarrow_{\$} \mathbb{Z}_p^*$. To achieve this, $\mathcal{S}$ sets up the following conditional unlinkability game to interact with $\mathcal{A}$.

1. *Setup*: $\mathcal{S}$ simulates the Setup algorithm, sets $h_1 = P$, and sends `gpk` to $\mathcal{A}$. $\mathcal{S}$ creates platforms $\mathcal{P}_i$, $\forall i \in [1, m_t]$, selects a platform $\mathcal{P}_{i^*}$, and initializes `cpl` and `ccl`.

2. *MemCreGen* and *LogCreGen*: $\mathcal{S}$ runs these phases as discussed in *MemCreGen* and *LogCreGen* phases in the proof of Lemma 2, except $\mathcal{S}$ acts as the platform as well as the issuer.

3. *Queries-Phase I*: $\mathcal{A}$ probes $\mathcal{S}$ with the queries discussed in the *Queries-Phase I* in the proof of Lemma 2. $\mathcal{A}$ also probes $\mathcal{S}$ with the following additional queries.

   (a) *Revoke*: $\mathcal{A}$ requests $\mathcal{S}$ to revoke the credential utilized to generate a signature σ' on a message M'. $\mathcal{S}$ simulates the Revoke algorithm, and responds with $\mathtt{tRL}'$.

4. *Challenge*: $\mathcal{A}$ submits the challenge, and $\mathcal{S}$ picks $\phi \in \{0, 1\}$ and responds with a signature as discussed in the *Challenge* phase in the proof of Lemma 2, except $\mathcal{S}$ selects $\mathtt{logCre}_{\mathtt{i}_\phi \mathtt{j}_\phi}$ using the SelectLogCre algorithm with the credential-selection rule `conUnlink`.

5. *Queries-Phase II (Restricted Queries)*: After obtaining the challenge, $\mathcal{A}$ continues to probe $\mathcal{S}$ with the queries mentioned in *Queries-Phase I* with the restrictions as discussed in *Queries-Phase II* in the proof of Lemma 2.

6. *Output*: $\mathcal{A}$ outputs $\phi' \in \{0, 1\}$ as the guess for ϕ, or aborts. If $\phi = \phi'$, then $\mathcal{S}$ outputs `true`, which means that $c = a \cdot b$; otherwise, $\mathcal{S}$ outputs `false`, which means that $c \leftarrow_{\$} \mathbb{Z}_p^*$.

In the above framework, $\mathcal{S}$ does not abort the game with a non-negligible probability which is represented by $\epsilon_{\mathcal{S}}^{con}$. Let $Adv_{\mathcal{A}}^{con}$ be the non-negligible advantage that $\mathcal{A}$ succeeds in breaking the conditional unlinkability property of LASER with the rule `conUnlink`, given that $\mathcal{S}$ does not abort during the above game. Hence, $\mathcal{S}$ has the non-negligible probability of at least $\epsilon_{\mathcal{S}}^{con} \cdot Adv_{\mathcal{A}}^{con}/2$ in breaking the $\mathbb{G}_1$-DDH assumption. □

Theorem 4. *In the random oracle model, LASER is adaptably anonymous under the $\mathbb{G}_1$-DDH assumption.*

Proof. This follows from the proofs of Lemma 2 and 3. □

Theorem 5. *In the random oracle model, LASER is traceable under the q-SDH assumption.*

Proof. Let there be an adversary $\mathcal{A}$ that succeeds to break the traceability property of LASER with a non-negligible probability. Also, let $(g_1, \rho_1, \cdots, \rho_q, g_2, \omega)$ be the instance of the q-SDH assumption, where $\rho_l = g_1^{\gamma^l}$, $\forall l \in [1, q]$, such that $q = 2 \cdot m_t \cdot m_s$, and $\omega = g_2^{\gamma}$. Given this instance, a simulator $\mathcal{S}$ intends to break the q-SDH assumption by generating a tuple $\left(g_1^{1/(\gamma+z)}, z\right)$, where $z \leftarrow_{\$} \mathbb{Z}_p^*$, with a non-negligible probability. To achieve this, $\mathcal{S}$ sets up the following traceability game to interact with $\mathcal{A}$.

1. *Setup*: $\mathcal{S}$ simulates the Setup algorithm as follows.
   (a) Create m_t platforms with identities $\mathcal{P}_i$, $\forall i \in [1, m_t]$.
   (b) Select $v_{\hat{i}\hat{j}}, z_{\hat{i}\hat{j}} \leftarrow_{\$} \mathbb{Z}_p^*$, $\forall \hat{i} \in [1, m_t], \forall \hat{j} \in [1, m_s]$.
   (c) Define a polynomial F with variable γ such that

$$F(\gamma) = \prod_{\hat{i}=1}^{m_t} \prod_{\hat{j}=1}^{m_s} \left(\gamma + v_{\hat{i}\hat{j}}\right)\left(\gamma + z_{\hat{i}\hat{j}}\right) = \sum_{l=0}^{q} \alpha_l \cdot \gamma^l,$$

   where $\alpha_l \in \mathbb{Z}_p^*$, $\forall l \in [0, q]$, are the coefficients of the polynomial $F(\gamma)$. Note that these coefficients can be computed without the knowledge of γ.
   (d) Compute $\widetilde{g}_1 = g_1^{F(\gamma)} = \prod_{l=0}^{q} \rho_l^{\alpha_l}$. Note that we denote $\rho_0 = g_1$.
   (e) Select $r_1, r_2, r_3 \leftarrow_{\$} \mathbb{Z}_p^*$; and compute $h_1 = \widetilde{g}_1^{r_1}$, $h_2 = \widetilde{g}_1^{r_2}$, and $h_3 = \widetilde{g}_1^{r_3}$.
   (f) Set the group public key $\mathtt{gpk} = (\widetilde{g}_1, h_1, h_2, h_3, g_2, \omega)$.
   (g) Send $\mathtt{gpk}$ to $\mathcal{A}$.
2. *MemCreGen*: For each $i \in [1, m_t]$ and $j \in [1, m_s]$, $\mathcal{S}$ simulates the $\mathsf{MemCreGen}$ protocol as follows.
   (a) Select $f_i \leftarrow_{\$} \mathbb{Z}_p^*$, and set the secret key $\mathsf{tsk_i} = f_i$.

(b) Define a polynomial $F_m(\gamma) = \frac{F(\gamma)}{(\gamma+v_{ij})} = \sum_{l=0}^{q-1} \beta_l \cdot \gamma^l$, where $\beta_l \in \mathbb{Z}_p^*$, $\forall l \in [0, q-1]$, are the coefficients of the polynomial $F_m(\gamma)$.

(c) Select $u_{ij} \leftarrow_{\$} \mathbb{Z}_p^*$, and compute

$$J_{ij} = \left(\widetilde{g}_1 \cdot h_1^{f_i} \cdot h_2^{u_{ij}}\right)^{\frac{1}{\gamma+v_{ij}}} = g_1^{F_m(\gamma)\cdot(1+r_1\cdot f_i+r_2\cdot u_{ij})} = \prod_{l=0}^{q-1} \rho_l^{\beta_l\cdot(1+r_1\cdot f_i+r_2\cdot u_{ij})}.$$

(d) Set $\texttt{memCre}_{\texttt{ij}} = (J_{ij}, u_{ij}, v_{ij})$.

3. *LogCreGen*: For each $i \in [1, m_t]$ and $j \in [1, m_s]$, $\mathcal{S}$ simulates the LogCreGen protocol as follows.

(a) Define a polynomial $F_g(\gamma) = \frac{F(\gamma)}{\gamma+z_{ij}} = \sum_{l=0}^{q-1} \delta_l \cdot \gamma^l$, where $\delta_l \in \mathbb{Z}_p^*$, $\forall l \in [0, q-1]$ are the coefficients of the polynomial $F_g(\gamma)$.

(b) Select $x_{ij}, y_{ij} \leftarrow_{\$} \mathbb{Z}_p^*$, and compute

$$\begin{aligned} A_{ij} &= (\widetilde{g}_1 \cdot h_1^{f_i} \cdot h_2^{x_{ij}} \cdot h_3^{y_{ij}})^{\frac{1}{\gamma+z_{ij}}} = g_1^{F_g(\gamma)\cdot(1+r_1\cdot f_i+r_2\cdot x_{ij}+r_3\cdot y_{ij})} \\ &= \prod_{l=0}^{q-1} \rho_l^{\delta_l\cdot(1+r_1\cdot f_i+r_2\cdot x_{ij}+r_3\cdot y_{ij})}. \end{aligned}$$

(c) Set $\texttt{logCre}_{\texttt{ij}} = (A_{ij}, x_{ij}, y_{ij}, z_{ij})$.

4. *Queries*: $\mathcal{A}$ queries $\mathcal{S}$ about the following.

(a) *Sign*: $\mathcal{A}$ requests $\mathcal{S}$ to generate a signature σ_s on a message M for $\mathcal{P}_i$, and send σ_s. $\mathcal{S}$ selects a login credential $\texttt{logCre}_{\texttt{ij}}$, simulates the Sign protocol, and responds with the generated σ_s.

(b) *TskCorrupt*: $\mathcal{A}$ queries $\mathcal{S}$ for the TPM's secret key of $\mathcal{P}_i$. $\mathcal{S}$ responds to $\mathcal{A}$ with corresponding $\texttt{tsk}_{\texttt{i}}$.

(c) *MemCreCorrupt*: $\mathcal{A}$ requests $\mathcal{S}$ to send j^{th} membership credential of the platform $\mathcal{P}_i$. $\mathcal{S}$ sends $\texttt{memCre}_{\texttt{ij}}$ to $\mathcal{A}$.

(d) *LogCreCorrupt*: $\mathcal{A}$ requests $\mathcal{S}$ to send j^{th} login credential of the platform $\mathcal{P}_i$. $\mathcal{S}$ sends $\texttt{logCre}_{\texttt{ij}}$ to $\mathcal{A}$.

5. *Output*: If $\mathcal{A}$ succeeds to break the traceability property, it outputs a message M_* and a forged signature σ_*.

In the above framework, $\mathcal{S}$ succeeds whenever $\mathcal{A}$ succeeds. Hence, $\mathcal{S}$ obtains a successful forgery with a non-negligible probability. Further, $\mathcal{S}$ rewinds the framework to obtain two forged signatures on the same message, where the commitments are the same, but the challenges and responses are different. $\mathcal{S}$ then utilizes the knowledge extractor of the zero-knowledge proof to extract $(A_*, x_*, y_*, z_*, f_*)$, where

$$A_* = \left(\widetilde{g}_1 \cdot h_1^{f_*} \cdot h_2^{x_*} \cdot h_3^{y_*}\right)^{\frac{1}{\gamma+z_*}} = g_1^{\frac{F(\gamma)\cdot(1+r_1\cdot f_*+r_2\cdot x_*+r_3\cdot y_*)}{\gamma+z_*}}.$$

$\mathcal{S}$ sets $F(\gamma) \cdot (1 + r_1 \cdot f_* + r_2 \cdot x_* + r_3 \cdot y_*) = (\gamma + z_*) \cdot F_d(\gamma) + d_*$, for some polynomial $F_d(\gamma) = \sum_{l=0}^{q-1} d_l \cdot \gamma^l$, and constant $d_* \in \mathbb{Z}_p^*$. It implies that

$$A_* = g_1^{F_d(\gamma) + \frac{d_*}{\gamma + z_*}} \implies g_1^{\frac{1}{\gamma + z_*}} = \left(A_* \cdot g_1^{-F_d(\gamma)}\right)^{\frac{1}{d_*}},$$

where $g_1^{F_d(\gamma)} = \prod_{l=0}^{q-1} \rho_l^{d_l}$. In this way, $\mathcal{S}$ generates $\left(g_1^{\frac{1}{\gamma + z_*}}, z_*\right)$ with a non-negligible probability, and hence breaks the q-SDH assumption. □

Theorem 6. *In the random oracle model, LASER is non-frameable under the* $\mathbb{G}_1$*-DL assumption.*

Proof. Let there be an adversary $\mathcal{A}$ that succeeds to break the non-frameability property of LASER with a non-negligible probability. Let $(P, P^a) \in \mathbb{G}_1^2$, where $a \in \mathbb{Z}_p^*$, be the instance of the $\mathbb{G}_1$-DL assumption. Given this instance, a simulator $\mathcal{S}$ intends to break the $\mathbb{G}_1$-DL assumption by computing a with a non-negligible probability. To achieve this, $\mathcal{S}$ sets up the following non-frameability game to interact with $\mathcal{A}$.

1. The *Setup*, *MemCreGen*, *LogCreGen* and *Queries* phases in this game follow the same steps as discussed in the *Setup*, *MemCreGen*, *LogCreGen* and *Queries* phases in the proof of Lemma 2, respectively. Basically, $\mathcal{S}$ generates all the signatures corresponding to a specific platform $\mathcal{P}_{i^*}$ by assuming that its TPM's secret key $\mathtt{tsk}_* = a$.

2. *Output*: $\mathcal{A}$ outputs a signature σ_* on a message M_* for a platform $\mathcal{P}_{\hat{i}}$. If $\hat{i} \neq i^*$, $\mathcal{S}$ aborts the game.

In the above framework, $\mathcal{S}$ does not abort the game with a non-negligible probability. This means that $\mathcal{S}$ obtains a successful forgery with a non-negligible probability. Further, $\mathcal{S}$ rewinds the above framework to obtain two forged signatures on the same message, where the commitments are the same, but the challenges and responses are different. Using the knowledge extractor of the zero-knowledge proof, $\mathcal{S}$ extracts $(A_*, x_*, y_*, z_*, f_*)$ utilized to generate the forged signatures. Since the forged signatures belong to the platform $\mathcal{P}_{i^*}$, $\mathcal{S}$ determines a to be equal to f_*. Hence, $\mathcal{S}$ breaks the $\mathbb{G}_1$-DL assumption with a non-negligible probability. □