Direct Anonymous Attestation with Efficient Verifier-Local Revocation for Subscription System

Vireshwar Kumar
Virginia Tech
viresh@vt.edu

He Li
Virginia Tech
heli@vt.edu

Noah Luther
Virginia Tech
nluther@vt.edu

Pranav Asokan
Virginia Tech
pranav06@vt.edu

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

Kaigui Bian
Peking University
bkg@pku.edu.cn

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

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

ABSTRACT

For a computing platform that is compliant with the Trusted Platform Module (TPM) standard, direct anonymous attestation (DAA) is an appropriate cryptographic protocol for realizing an anonymous subscription system. This approach takes advantage of a cryptographic key that is securely embedded in the platform's hardware, and enables privacy-preserving authentication of the platform. In all of the existing DAA schemes, the platform 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 platform'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 procedure (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. We have implemented LASER on a laptop with an on-board TPM. To the best of our knowledge, this is the first implementation of a DAA scheme on an actual TPM cryptoprocessor that is compliant with the most recent TPM specification, viz., TPM 2.0.

KEYWORDS

Direct anonymous attestation; trusted platform module (TPM).

ACM Reference Format:
Vireshwar Kumar, He Li, Noah Luther, Pranav Asokan, Jung-Min (Jerry) Park, Kaigui Bian, Martin B. H. Weiss, and Taieb Znati. 2018. Direct Anonymous Attestation with Efficient Verifier-Local Revocation for Subscription System. In *ASIA CCS '18: 2018 ACM Asia Conference on Computer and Communications Security, June 4–8, 2018, Incheon, Republic of Korea.* ACM, New York, NY, USA, 8 pages. https://doi.org/10.1145/3196494.3196497

ASIA CCS '18, June 4–8, 2018, Incheon, Republic of Korea

ACM ISBN 978-1-4503-5576-6/18/06…$15.00
https://doi.org/10.1145/3196494.3196497

1 INTRODUCTION

There has been a rapid growth in the online electronic subscription services where the subscribed users access contents (e.g., video) and/or resources (e.g., software) made available by the service providers. In many subscription services, the users have significant concerns for remaining anonymous and preserving privacy so that the providers cannot track their interests, usage patterns, geographical locations, and other personal details. Hence, there is a major research thrust to develop an *Anonymous Subscription System* (ASS) in which the providers allow access of their services to the subscribed and authenticated, but anonymous users [4, 5, 20, 22, 23]. *Direct Anonymous Attestation* (DAA) is the most appropriate cryptographic protocol to realize the ASS as DAA enables privacy-preserving authentication of a user's platform by utilizing the cryptographic key which is securely embedded into the platform's hardware [7].

A DAA scheme involves three entities—a platform, a verifier and an issuer. We consider the subscribed user as the platform, and the service provider 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 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 [28]. 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 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 [8, 14, 28]. This specification has also been published as the international standard ISO/IEC 11889:2015 [21]. 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 [7–9, 11, 14, 15]. The existing DAA schemes employ a signature-based revocation check procedure which enables the issuer the ability of revoking a platform based on its malicious

signature [8, 11]. In these schemes, the revocation list contains tuples retrieved from the malicious login signatures. 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 [12, 25]: (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 of DAA 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 hundreds of platforms are revoked per day in a network with a large number (e.g., a million) of users [24, 27].

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 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 enabling controlled verifier-local revocation [6], and by shifting most of the computational and communication costs (to enable signature-based revocation) 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 subscription systems, because the critical performance bottlenecks of those applications are determined by the performance of the online procedure at the platform. Unlike legacy DAA schemes, LASER is scalable, and can be deployed in the subscription system which is expected to have a long revocation list.

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 [29], or the results are obtained from the implementation on TPM version 1.2 [13]. In this paper, we have validated our analytical results by implementing LASER 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 most recent TPM specification version 2.0.

2 PROPOSED SCHEME: LASER

2.1 Definition and Overview of LASER

Definition 2.1. 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. This algorithm outputs an *issuer's secret key*, represented by isk, and a corresponding *group public key*, represented by gpk. The isk is known only to the issuer, and 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 gpk and isk, the input to the TPM is gpk, and the inputs to the host are gpk and m_s. Here, m_s represents the number of absolutely unlinkable credentials allotted to each platform, and its value is set based on the unlinkability requirement of the platform (see Appendix A). In this protocol, the TPM generates a *TPM's secret key*, represented by tsk, a corresponding *TPM's public key*, represented by tpk, and a *key handle*, represented by hdl. The hdl specifies the location of the tsk in the secure memory of TPM. The tsk is known only to the TPM, and tpk and hdl are forwarded to the host. Further, the host acquires a *membership credential*, represented by $\mathtt{memCre}_j$, for each $j \in [1, m_s]$, from the issuer. Finally, the host outputs a *membership credential list*, represented by $\mathtt{mcl} = (\mathtt{memCre}_1, \cdots, \mathtt{memCre}_{m_s})$.

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

(4) $(\mathtt{logCre}_j, \mathtt{cul}') \leftarrow \mathsf{SelectLogCre}(\mathtt{lcl}, \mathtt{cul}, \mathtt{csr})$: This credential selection algorithm is performed by the host. The inputs to the host are a *login credential list*, represented by lcl, a *credential-usage list*, represented by cul, and a *credential-selection rule*, represented by csr. Before running this algorithm, the TPM and the host run the LogCreGen protocol with the issuer for all $j \in [1, m_s]$ to obtain the login credential list $\mathtt{lcl} = (\mathtt{logCre}_1, \cdots, \mathtt{logCre}_{m_s})$, where each instance of the LogCreGen protocol is initiated by the host after a randomly selected time interval. The host also employs an application-based rule assignment for csr which takes one of the two values, i.e., $\mathtt{csr} \in \{\mathtt{absUnlink}, \mathtt{conUnlink}\}$, 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 csr and the current usage list cul, and outputs the login credential $\mathtt{logCre}_j$ from lcl. It also outputs the updated list cul′.

(5) $\sigma_s \leftarrow \mathsf{Sign}(\mathtt{gpk}, \mathtt{tsk}, \mathtt{hdl}, \mathtt{tpk}, \mathtt{logCre}_j, M)$: 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, represented by $M \in \{0,1\}^*$. This protocol outputs a login signature σ_s.

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

(7) $\mathtt{tRL}' \leftarrow \mathsf{Revoke}(\mathtt{gpk}, \mathtt{ctl}, \sigma_s, M, \mathtt{tRL})$: This is the signature-based revocation algorithm which is utilized by the issuer to revoke the login credential of the compromised platform. This

algorithm takes gpk, ctl, a signature σ_s, a message M, and tRL, as inputs. It updates the tRL, and outputs the updated list, tRL′.

(8) true/false ← Identify(gpk, σ_s, M, 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 tsk$_*$ as inputs. It outputs the value true if σ_s is proved to be generated using tsk$_*$; otherwise, it outputs the value false.

Here, we paraphrase the technical details of LASER and highlight our contributions. 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. For each $j \in [1, m_s]$, the platform performs the LogCreGen protocol where it utilizes a membership credential, and acquires a corresponding login credential from the issuer. To support the signature-based verifier-local revocation, the issuer includes a revocation token, represented by y_j, in each login credential. The issuer also publishes the token-based revocation list tRL which contains all the revoked tokens. In the Sign protocol, the platform utilizes a login credential to generate a signature on the login request message. In the login signature, the platform includes a parameter called "token response" which is computed by the exponentiation of the token y_j over a randomly selected "base". Through the Verify algorithm, the verifier checks the validity of the signature. The verifier also determines the revocation status of the login credential utilized to generate the signature by computing the exponentiation of each of the revoked tokens in tRL over the base, and matching the result with the token response.

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 [12]. This results in a linear increase of the computational complexity and the communication overhead at the platform with the increase in the number of revoked credentials. 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. The inclusion of the revocation token in the login signature enables the controlled verifier-local revocation of the login credentials. 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, in LASER, most of the burden of the revocation check procedure at the platform 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 advantages are especially important when a DAA scheme needs to be deployed for the ASS with a large number of users.

Among the security properties (discussed in Appendix A), the unlinkability is an important notion to consider in evaluating LASER with respect to other DAA schemes. In the existing DAA schemes, the login signatures generated by the platform satisfy the concept of *absolute unlinkability* (i.e., the signatures are unlinkable by the issuer and the verifier) [3]. Here, linking two signatures means that they are proved to be generated by the same platform. In LASER, the platform obtains each of the m_s login credentials using zero-knowledge proofs of knowledge in the LogCreGen protocol, and hence any two login credentials cannot be linked by the issuer. This means that two signatures generated using two different login credentials cannot be linked by the issuer. However, since a login credential contains the revocation token, all the signatures generated using one login credential can be linked by the issuer. In fact, it is this linkability which enables verifier-local revocation. Note that the verifier cannot link any two signatures which are either generated using different login credentials or the same login credential. Hence, in addition to the concept of absolute unlinkability, our design of LASER leads to the concept of unlinkability called *conditional unlinkability* where the generated signatures remain unlinkable by the verifier, but they may be linkable (when generated using the same credential) or may not be linkable (when generated using different login credentials) by the issuer [4]. In this paper, we present 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 using the SelectLogCre algorithm (see Appendix A). Our results exhibit that the notion of adaptable unlinkability enables LASER to provide the needed privacy in a flexible and practical manner.

2.2 Details of LASER

We assume that $\mathbb{Z}_p^*$ represents 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 $\mathbb{G}_2$ [18]. We also assume a hash function $\mathcal{H}_z : \{0,1\}^* \rightarrow \mathbb{Z}_p^*$ which is considered as a random oracle. The parameters for $(\mathbb{G}_1, \mathbb{G}_2, e, \mathcal{H}_z)$ are published.

(gpk, isk) ← **Setup**(1^λ): This setup algorithm is run by the issuer 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 isk = γ, and gpk = $(g_1, h_1, h_2, h_3, g_2, \omega)$.
(4) Output (gpk, isk).

(tsk, hdl, tpk, mcl) ← **MemCreGen**(gpk, isk, m_s): This protocol is performed among the TPM, the host and the issuer as follows.

(1) Upon the request of the host, the TPM generates tsk = f, stores it in its secure memory, and forwards the outputs hdl and 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

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

(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 $\texttt{memCre}_\texttt{j} = (J_j, u_j, v_j)$. Note that (J_j, u_j, v_j) is a BBS+ signature on f (see [1]).
   (b) Output $\texttt{mcl} = (\texttt{memCre}_1, \cdots, \texttt{memCre}_{\texttt{m}_\texttt{s}})$.

$(\texttt{ctl}', \texttt{logCre}_\texttt{j}) \leftarrow$ **LogCreGen**(gpk, isk, ctl, tsk, hdl, tpk, $\texttt{memCre}_\texttt{j}$): This protocol is performed among the TPM, the host and the issuer. Here, we assume that the credential token list is represented as $\texttt{ctl} = \{(K_i, y_i) : K_i \in \mathbb{G}_1, y_i \in \mathbb{Z}_p^*, \forall i \in [1, m_t m_s]\}$, where m_t represents the number of platforms in the network. 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 tsk (by using parameters B_g and C_g), 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 ctl, verify that $K_j \notin \texttt{ctl}$ by utilizing a conventional 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 ctl, and output the updated list ctl′.
   (c) Send (A_j, y_j, z_j) to the host.
(5) The host outputs the login credential, $\texttt{logCre}_\texttt{j} = (A_j, x_j, y_j, z_j)$. Note that (A_j, y_j, z_j) is a BBS+ signature on (f, x_j).

$(\texttt{logCre}_\texttt{j}, \texttt{cul}')$ ← **SelectLogCre**(lcl, cul, csr): This login credential selection algorithm is performed by the host. To keep track of the utilized credentials from lcl, each entry in the cul takes one of the three values, i.e., $\texttt{cul}_\texttt{j} \in \{\texttt{unUsed}, \texttt{absUsed}, \texttt{conUsed}\}$, corresponding to the cases where $\texttt{logCre}_\texttt{j}$ 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. It proceeds as follows.

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

$\sigma_s \leftarrow$ **Sign**(gpk, tsk, hdl, tpk, $\texttt{logCre}_\texttt{j}$, M): This login signature generation protocol is performed between the TPM and the host. This protocol outputs a login signature σ_s which presents the SPK of tsk (by using parameters B_s and C_s), a valid revocation token y_j (by using parameters D_s and E_s), and a BBS+ signature (A_j, y_j, z_j) on (f, x_j). The σ_s is given as

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

valid/invalid ← **Verify**(gpk, σ_s, M, tRL): This verification algorithm checks the validity of the signature, and the revocation status of the login credential. If the following verification steps are successful, this algorithm outputs the value valid; otherwise it outputs the value invalid. Here, we assume that the token-based revocation list tRL is represented as $\texttt{tRL} = \{y_i : y_i \in \mathbb{Z}_p^*, \forall i \in [1, m_r]\}$, where m_r is the number of revoked login credentials.

(1) Verify the validity of the signature σ_s.
(2) For each $y_i \in \texttt{tRL}$, compute $E_i = D_s^{y_i}$, and verify that $E_i \neq E_s$.

tRL′ ← **Revoke**(gpk, ctl, σ_s, M, tRL): This signature-based revocation algorithm is run by the issuer as follows.

(1) Verify that σ_s is an honestly generated signature, i.e., valid ← Verify(gpk, σ_s, M, tRL). If the verification fails, abort.
(2) For each y_i in the list ctl, compute $E_i = D_s^{y_i}$, and find the index i such that $E_i = E_s$, where D_s and E_s are obtained from σ_s.
(3) Append y_i to tRL, and output the updated tRL′. This revokes the login credential utilized to generate the signature σ_s.

true/false ← **Identify**(gpk, σ_s, M, $\texttt{tsk}_*$): This algorithm verifies if σ_s is generated using $\texttt{tsk}_* = f_*$. This algorithm outputs the value true if both the following verification steps are successful; otherwise this algorithm outputs the value false.

(1) Verify that valid ← Verify(gpk, σ_s, M, $\varnothing$). Here, $\varnothing$ represents an empty set.
(2) Compute $C_* = B_s^{f_*}$, and verify that $C_s = C_*$.

3 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) [11]. We consider only the computationally expensive operations—i.e., exponentiation in $\mathbb{G}_1$ and bilinear mapping. We represent the number of exponentiations in $\mathbb{G}_1$ and bilinear mappings by E_{G_1} and B_M, respectively. 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.

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_{Zp}	L_{G_1}	L_{Zp}
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

To analyze the computational complexity in a DAA scheme, we divide the operations 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 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 $O(1)$ and $O(m_s)$, respectively. Most importantly, in Table 1, we observe that the computational complexities of the platform's online operations are $O(m_r)$ in CDL-EPID as compared to $O(1)$ in LASER. Although the computational complexity of verifier's online operations are $O(m_r)$ in CDL-EPID as well as LASER, we note that the verifiers (service providers) have access to servers with large computational resources, and hence they are able to handle the online operations.

To analyze the communication overhead in a DAA scheme, we divide the communications 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 communicating 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 the online communication overhead increases by $O(m_r)$ in CDL-EPID as compared to $O(1)$ in LASER.

4 IMPLEMENTATION RESULTS

We assume an illustrative application where one million users are subscribed at the issuer in an online subscription service. The users are required to renew their subscription every month which consists of 30 days. Each user (platform) sends login requests to ten providers (verifiers) per day, and generates one login signature corresponding to the request to each provider per day. This means that the total number of login signatures generated per day by a user is ten. Moreover, we assume that 0.2 percent of the laptop platforms belonging to the users are revoked every month because they are lost or stolen [27]. 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—(1) CDL-EPID, (2) *LASER-A*: LASER with absolute unlinkability, i.e., $m_s = 300$, (3) *LASER-B*: LASER with conditional unlinkability where the issuer can link some login signatures, i.e., $m_s = 30$, and (4) *LASER-C*: LASER with conditional unlinkability where the issuer can link all the login signatures, i.e., $m_s = 1$.

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 [26], and IBM Trusted Software Stack (TSS) for TPM 2.0 [19] to prototype CDL-EPID and LASER in *C*. The prototypes of the TPM and the host in LASER (available at [17]) comprise of 600 and 1800 lines of code, respectively. The total development time of the prototypes was around 1000 man-hours. We utilize the Barreto-Naehrig (BN) curve which is standardized for DAA by the TCG [2]. 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 implemented with TPM 2.0 specification provides 85 bits of security [10].

By averaging over 100 iterations, we obtain the running time in milliseconds (ms) for different protocols in LASER and CDL-EPID. Our results indicate that with $m_r = 1000$, the time taken to generate a signature is only 348 ms in LASER-A, LASER-B and LASER-C 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

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

that in CDL-EPID. From 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. 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, LASER-B and LASER-C are 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 all the three deployment scenarios of LASER incur significantly lower online overhead - in terms of both computation and communication - compared to CDL-DAA at the cost of higher offline overhead. This trade-off between offline and online computational and communication costs is very advantageous because the online computational and communication occur significantly more often than the offline ones. Also, as the online procedure require significantly lower latency than the offline procedure, LASER is more practical than CDL-EPID when m_r is large. Another noteworthy attribute of LASER is its realization of the security notion that we refer to as *adaptable unlinkability* (see Apendix A). 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).

5 CONCLUSION

In this paper, we proposed a novel DAA scheme called LASER which supports verifier-local revocation, and can be utilized to realize the anonymous subscription system. We have shown that LASER achieves a significant performance improvement over the prior art by shifting most of the computational and communication overhead at the platform from its online protocol to its offline protocol. We have evaluated LASER through analytical analysis as well as through an actual implementation on a TPM cryptoprocessor.

REFERENCES

[1] M. H. Au, W. Susilo, Y. Mu, and S. S. M. Chow. 2013. Constant-size dynamic k-times anonymous authentication. *IEEE Systems Journal* 7, 2 (2013), 249–261.

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

[3] D. Bernhard, G. Fuchsbauer, E. Ghadafi, N. P. Smart, and B. Warinschi. 2013. Anonymous attestation with user-controlled linkability. *International Journal of Information Security* 12, 3 (2013), 219–249.

[4] Alberto Blanco-Justicia and Josep Domingo-Ferrer. 2016. Privacy-aware loyalty programs. *Computer Communications* 82 (2016), 83 – 94.

[5] M. Blanton. 2008. Online Subscriptions with Anonymous Access. In *Proceedings of the ACM Symposium on Information, Computer and Communications Security (ASIACCS)*. 217–227.

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

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

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

[9] E. Brickell and J. Li. 2012. Enhanced privacy ID: A direct anonymous attestation scheme with enhanced revocation capabilities. *IEEE Transactions on Dependable and Secure Computing* 9, 3 (2012), 345–360.

[10] J. Camenisch, L. Chen, M. Drijvers, A. Lehmann, D. Novick, and R. Urian. 2017. One TPM to bind them all: Fixing TPM 2.0 for provably secure anonymous attestation. In *IEEE Symposium on Security and Privacy (S&P)*. 901–920.

[11] J. Camenisch, M. Drijvers, and A. Lehmann. 2016. Anonymous attestation using the strong Diffie Hellman assumption revisited. In *Proceedings of the 9th International Conference on Trust and Trustworthy Computing (TRUST)*. 1–20.

[12] L. Chen and J. Li. 2010. Revocation of direct anonymous attestation. In *International Conference on Trusted Systems*. 128–147.

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

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

[15] L. Chen and R. Urian. 2015. DAA-A: Direct anonymous attestation with attributes. In *International Conference on Trust and Trustworthy Computing*. 228–245.

[16] V. Kumar et al. 2018. Direct anonymous attestation with efficient verifier-local revocation for subscription system. Cryptology ePrint Archive, 2018/290. (2018).

[17] V. Kumar et al. 2018. Prototype of LASER. (2018). Retrieved Nov. 1, 2017 from https://github.com/luthern/laser/

[18] S. D. Galbraith, K. G. Paterson, and N. P. Smart. 2008. Pairings for cryptographers. *Discrete Applied Mathematics* 156, 16 (2008), 3113–3121.

[19] K. Goldman. 2017. IBM TPM 2.0 TSS. (2017). Retrieved Nov. 1, 2017 from https://sourceforge.net/projects/ibmtpm20tss/

[20] T. Gupta, N. Crooks, W. Mulhern, S. Setty, L. Alvisi, and M. Walfish. 2016. Scalable and Private Media Consumption with Popcorn. In *Proceedings of the 13th Usenix Conference on Networked Systems Design and Implementation (NSDI)*. 91–107.

[21] ISO/IEC. 2015. ISO/IEC 11889:2015, Parts 1-4. (2015). Retrieved Nov. 1, 2017 from http://www.iso.org/iso/home/store/catalogue_tc/catalogue_detail.htm?csnumber=66510

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

[23] M. Z. Lee, A. M. Dunn, B. Waters, E. Witchel, and J. Katz. 2013. Anon-Pass: Practical Anonymous Subscriptions. In *Proceedings of the IEEE Symposium on Security and Privacy (S&P)*. 319–333.

[24] Y. Liu, W. Tome, L. Zhang, D. Choffnes, D. Levin, B. Maggs, A. Mislove, A. Schulman, and C. Wilson. 2015. An End-to-End Measurement of Certificate Revocation in the Web's PKI. In *Proceedings of the Internet Measurement Conference (IMC)*. 183–196.

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

[26] B. Lynn. 2017. PBC: Pairing-based cryptography. (2017). Retrieved Nov. 1, 2017 from https://crypto.stanford.edu/pbc/

[27] Ponemon Institute. 2010. The billion dollar lost laptop problem: Benchmark study of U.S. organizations. (2010).

[28] Trusted Computing Group. 2014. TPM library specification. (2014). Retrieved Nov. 1, 2017 from http://www.trustedcomputinggroup.org/tpm-library-specification/

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

A SECURITY DEFINITIONS

In the following discussions, we present the definitions of the security properties of LASER. We provide the theorems corresponding to each of the security properties of LASER, and the comprehensive proofs of those theorems in the full length version of the paper which is available at [16].

A.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 A.1. For all λ, m_s, $\mathtt{ctl}$, $\mathtt{cul}$, $\mathtt{csr}$, M, and $\mathtt{tRL}$, LASER satisfies the correctness property if

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

A.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. 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 A.2. For an adversary $\mathcal{A}$ and a challenger C, the *absolute unlinkability game* is defined as follows.

(1) *Setup*: C simulates the Setup algorithm, and provides $\mathcal{A}$ with the resulting $\mathtt{isk}$ and gpk. C also creates m_t platforms with identities $\mathcal{P}_i$, $\forall i \in [1, m_t]$. Further, C initializes a list of the corrupted platforms, $\mathtt{cpl} = \emptyset$, and a list of corrupted credentials, $\mathtt{ccl} = \emptyset$.
(2) *MemCreGen*: C acts as the platform, and simulates the protocol MemCreGen with $\mathcal{A}$ which acts as the issuer. For all $i \in [1, m_t]$, C generates $\mathtt{tsk_i}$, and acquires $\mathtt{memCre_{ij}}$, $\forall j \in [1, m_s]$.
(3) *LogCreGen*: C acts as the platform, and simulates the LogCreGen protocol with $\mathcal{A}$ which acts as the issuer. For all $i \in [1, m_t]$ and $j \in [1, m_s]$, C acquires $\mathtt{logCre_{ij}}$.
(4) *Queries-Phase I*: $\mathcal{A}$ queries C about the following.
   (a) *Sign*: $\mathcal{A}$ requests C to generate a signature on a message M on behalf of $\mathcal{P}_i$. C runs the SelectLogCre with the credential-selection rule $\mathtt{conUnlink}$ or $\mathtt{absUnlink}$. If C utilizes the rule $\mathtt{absUnlink}$, it appends $\mathtt{logCre_{ij}}$ to $\mathtt{ccl}$. Further, C simulates the Sign protocol, and responds to $\mathcal{A}$ with the signature σ_s.
   (b) *TskCorrupt*: $\mathcal{A}$ requests C to provide the TPM's secret key of $\mathcal{P}_i$. 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 $\mathcal{P}_i$. 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 $\mathcal{P}_i$. C responds to $\mathcal{A}$ with $\mathtt{logCre_{ij}}$, and appends $\mathtt{logCre_{ij}}$ to $\mathtt{ccl}$.
(5) *Challenge*: $\mathcal{A}$ submits to C a message M and two platforms $\mathcal{P}_{i_0}$ and $\mathcal{P}_{i_1}$ with the restriction that $\mathcal{P}_{i_0}, \mathcal{P}_{i_1} \notin \mathtt{cpl}$. C selects $\phi \leftarrow\!\!\$\, \{0, 1\}$. Here, $\leftarrow\!\!\$$ represents a random selection. Corresponding to $\mathcal{P}_{i_\phi}$, C selects $\mathtt{logCre_{i_\phi j_\phi}}$ using the credential-selection rule $\mathtt{absUnlink}$ in the SelectLogCre algorithm such that $\mathtt{logCre_{i_\phi j_\phi}} \notin \mathtt{ccl}$. Further, 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 C 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 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 $Adv^{abs}_{\mathcal{A}} = |\Pr(\phi' = \phi) - 1/2|$, where Pr represents the probability of an event. LASER with the rule $\mathtt{absUnlink}$ 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 A.3. For an adversary $\mathcal{A}$ and a challenger C, the *conditional unlinkability game* is defined as follows.

(1) *Setup*: C simulates the Setup algorithm, and provides $\mathcal{A}$ with the resulting gpk. C does not reveal $\mathtt{isk}$ to $\mathcal{A}$. C also creates m_t platforms with identities $\mathcal{P}_i$, $\forall i \in [1, m_t]$. Further, C initializes two lists $\mathtt{cpl}$ and $\mathtt{ccl}$.
(2) *MemCreGen*: C acts as the platform as well as the issuer, and simulates MemCreGen protocol to generate $\mathtt{tsk_i}$, $\forall i \in [1, m_t]$, and $\mathtt{memCre_{ij}}$, $\forall i \in [1, m_t], j \in [1, m_s]$.
(3) *LogCreGen*: C acts as the platform as well as the issuer, and simulates LogCreGen protocol to generate $\mathtt{logCre_{ij}}$, $\forall i \in [1, m_t]$ and $\forall j \in [1, m_s]$.
(4) *Queries-Phase I*: $\mathcal{A}$ probes C with the queries defined in the *Queries-Phase I* in the absolute unlinkability game. $\mathcal{A}$ probes C with the following additional queries.
   (a) *Revoke*: $\mathcal{A}$ requests C to revoke the login credential utilized to generate a signature σ'_s on a message M'. C simulates the Revoke algorithm and responds with the updated $\mathtt{tRL}'$.

(5) *Challenge*: $\mathcal{A}$ submits to C a message M and two platforms $\mathcal{P}_{i_0}$ and $\mathcal{P}_{i_1}$ with the restriction that $\mathcal{P}_{i_0}, \mathcal{P}_{i_1} \notin \mathtt{cpl}$. C selects $\phi \leftarrow_{\$} \{0,1\}$. Corresponding to $\mathcal{P}_{i_\phi}$, C selects $\mathtt{logCre}_{\mathtt{i}_\phi \mathtt{j}_\phi}$ using the credential-selection rule conUnlink in the SelectLogCre algorithm such that $\mathtt{logCre}_{\mathtt{i}_\phi \mathtt{j}_\phi} \notin \mathtt{ccl}$. Further, C runs the Sign protocol, and responds with the signature σ_s on M.
(6) *Queries-Phase II (Restricted Queries)*: $\mathcal{A}$ probes 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 $Adv^{con}_{\mathcal{A}} = |\Pr(\phi' = \phi) - 1/2|$. 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 A.4. 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.

A.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 A.5. For an adversary $\mathcal{A}$ and a challenger 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 $\mathtt{tRL}_*$.

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

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

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

A.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 A.6. For an adversary $\mathcal{A}$ and a challenger 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) $\mathtt{true} \leftarrow \mathsf{Identify}(\mathsf{gpk}, \sigma_*, M_*, \mathtt{tsk}_{\mathtt{i}^*})$, 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.

ACKNOWLEDGMENTS

This work was supported in part by National Science Foundation through grants 1314598, 1265886, and 1642928, and by the industry affiliates of the Broadband Wireless Access and Applications Center.