IEEE International Conference on Social Computing / IEEE International Conference on Privacy, Security, Risk and Trust

Enhanced Privacy ID from Bilinear Pairing for Hardware Authentication and Attestation

Ernie Brickell
Intel Corporation
Hillsboro, USA
Email: ernie.brickell@intel.com

Jiangtao Li
Intel Labs
Hillsboro, USA
Email: jiangtao.li@intel.com

***Abstract*—Enhanced Privacy ID (EPID) is a cryptographic scheme that enables the remote authentication and attestation of a hardware device while preserving the privacy of the device. EPID can be seen as a direct anonymous attestation scheme with enhanced revocation capabilities. In EPID, a device can be revoked if the private key embedded in the hardware device has been extracted and published widely so that the revocation manager finds the corrupted private key. In addition, the revocation manager can revoke a device based on the signatures the device has created, if the private key of the device is not known. In this paper, we introduce a new security notion of EPID including the formal definitions of anonymity and unforgeability. We also give a construction of an EPID scheme from bilinear pairing. Our EPID scheme is efficient and provably secure in the random oracle model under the strong Diffie-Hellman assumption and the decisional Diffie-Hellman assumption.**

***Index Terms*—hardware authentication; trusted computing; privacy; anonymity; direct anonymous attestation; cryptographic protocol**

I. Introduction

Consider the following authentication problem: a hardware device (e.g., a graphics chip, a trusted platform module, a mobile device, a smart phone, or a processor) wants to authenticate to a service provider that it is a genuine hardware device instead of a software simulator, so that the service provider can send a protected resource (e.g., one-time password or high definition media) to the device. One possible solution is that the hardware manufacturer assigns each device a unique device certificate. The device can authenticate to the service provider by showing the device certificate. However, such solution raises a serious privacy concern as the device certificate can uniquely identify the device.

Brickell, Camenisch, and Chen [10] introduced a cryptographic scheme called Direct Anonymous Attestation (DAA) that can solve the above problem. The original usage of DAA is for anonymous authentication of a special hardware device called the Trusted Platform Module (TPM). The DAA scheme was adopted by the Trusted Computing Group (TCG) [38] and standardized in the TCG TPM Specification 1.2 [37].

In a DAA scheme, a hardware device can be revoked only if the private key embedded in the hardware device has been extracted and published widely so that the revocation manager finds the corrupted private key. However, if an attacker corrupts a hardware device and obtains the device's private key, but he never publishes it, then there is no way to revoke the key in DAA. If the named base option in DAA is used, it can allow revocation based on signatures for all uses of the same named base, but it has the unfortunate property of removing the anonymity for all uses with the same named base. To get around the problem of the limited revocation properties of DAA, Brickell and Li [12] introduced the notion of Enhanced Privacy ID (EPID). In EPID, the revocation manager can revoke a hardware device based on the signatures that were signed by the private key of the device, without reducing the anonymity properties. The EPID scheme will have broader applicability beyond attestation and the TCG application. More motivations about EPID can be found in [13].

In an EPID scheme, there are four types of entities: an issuer, a revocation manager, platforms, and verifiers. The issuer could be the same entity as the revocation manager. The issuer is in charge of issuing membership to platforms, i.e., each platform obtains a unique private key from the issuer through a join process. A platform can prove membership to a verifier by creating a signature using its private key. The verifier can verify membership of the platform by verifying the signature, but he cannot learn the identity of the platform. One important feature of EPID is that nobody besides the platform knows the platform's private key and nobody can trace the signatures created by the platform if the platform has not been tampered. Yet an EPID scheme has to be able to revoke a platform if the platform's private key has been corrupted. There are two types of revocations in EPID: (1) private-key based revocation in which the revocation manager revokes a platform based on the platform's private key, and (2) signature based revocation in which the revocation manager revokes a platform based on the signatures created by the platform. A formal specification of EPID is given in Section II. In this paper, we provide two contributions:

1) We give a new security notion of EPID. This new security model is intended to address the same concept of EPID introduced in [12]. We formally model the two revocation methods into the security model, and we give a formal definition of anonymity and unforgeability with notions of revocation embedded.
2) We develop a concrete EPID scheme from bilinear maps. Our EPID scheme builds on top of Boneh, Boyen, and Shacham's group signature scheme [6] and Furukawa

and Imai group signature scheme [28]. Our construction of EPID is efficient and provably secure in the random oracle model under the strong Diffie-Hellman assumption and the decisional Diffie-Hellman assumption. Our new EPID scheme requires a much shorter key length and signature size than the original RSA based EPID scheme [12] and yet achieves a higher level of security.

A. Motivation for Signature Based Revocation

We now present a concrete example for motivating signature based revocation. Suppose each platform has a unique EPID private key. Consider there is a provisioning server for provisioning content protection keys to each platform. Only platform with valid EPID private key can obtain a unique content protection key from the provisioning server. If an attacker breaks one EPID private key, he could use the corrupted EPID private key to obtain content protection keys. If the attacker publishes the EPID private key over the Internet, we can revoke the key. However, in practice, the attacker may embed the obtained content protection key in a media ripper software without publishing the EPID private key. Once we find the ripper software on the Internet and extract the content protection key in it, we can back trace to the EPID signature that was used to obtain the content protection key. We then revoke the platform based on the signature without knowing the corrupted EPID private key.

A possible alternative to handle revocation is to add traceability to the EPID scheme, as most group signature schemes do. That is, we give the revocation manager the ability to open a signature and identify the actual signer. To revoke a platform based on its signature, the revocation manager first finds out the platforms private key or its identity, then put the private key into the revocation list. As in DAA schemes [10], [11], EPID scheme chooses not to have traceability from the revocation manager in order to provide *maximum privacy* for the unrevoked platforms. Traceability provides the capability that a revocation manager can determine which platform generates which signatures, for any signatures at any time, without any acknowledgement from the user that is being traced. This is not desirable from a privacy perspective. In EPID, if a platforms private key has been revoked, i.e., placed in a revocation list, the user of the platform will know that he is revoked or being traced. If the user finds that his platform is not in the revocation list, then he is assured that nobody can trace him, including the issuer and the revocation manager. Observe that, if the revocation manager does not have traceability and the signature cannot be opened, revocation based on signature is a much more challenging problem.

B. Revocation of Hardware Devices

The EPID scheme is mainly designed for hardware authentication and attestation. Consider that a hardware manufacturer issues a unique EPID private key for each hardware device. Assume the hardware device has a secure storage to store the private key and a trusted execution environment to use the private key for creating the signatures. We expect the revocation of hardware devices to be a rare event for the following reasons:

1) To use the EPID scheme for hardware authentication, we revoke an EPID key only if it has been extracted out of the hardware device. EPID enables platform authentication, but not for user authentication. If a platform changes ownership or is stolen, there is no need to revoke the EPID key inside as it is a valid platform.
2) In general, hardware has a good protection on its key materials than software based solutions. If the secure storage and trusted execution environment of the platform are implemented in hardware, then the only way to extract the EPID key is to launch a physical attack.

Although our revocation method in the EPID scheme seems expensive, it is more capable, privacy friendly, and practical given our estimation that revocation is a rare event.

C. Related Work

The EPID scheme can be seen as a special group signature scheme [1], [6], [21], [24] without the feature of opening a group signature and identifying the signer of the signature. There have been several revocation methods proposed for group signatures, such as [9], [36], [18], [2], [7]. The unique property that EPID has that none of the above have, is the capability to revoke a private key that generated a signature, without being able to open the signature. The EPID scheme in this paper also shares some properties with identity escrow [30], anonymous credential systems [17], [22], pseudonym system of Brands [8], and blacklistable anonymous credentials scheme [39].

As mentioned earlier, EPID can be seen as a DAA scheme with additional revocation capabilities. We remove some features of DAA from the design of EPID, such as the name based option and the outsourcing capability, since those features are more TPM specific. We could easily add those features back to EPID if needed. After DAA was first introduced by Brickell, Camenisch, and Chen [10], it has drawn a lot of attention from both industry and cryptographic community (e.g., [16], [32], [3], to list a few). Brickell, Chen, and Li recently constructed the first pairing based DAA scheme [11]. Later Chen, Morrissey, and Smart [26] showed that the DAA scheme can be further optimized by transferring the underlying pairing groups from the symmetric to the asymmetric settings. Recently, Chen [25] and Brickell and Li [15] proposed DAA schemes built on top of this paper. Both DAA schemes [25], [15] focus on the TPM implementation and outsourcing capability, but they do not have the additional revocation capability as in the EPID schemes.

D. Organization of This Paper

Rest of this paper is organized as follows. We give a formal specification of EPID and present the corresponding security model in Section II. We then define our notations and present security assumptions in Section III. We present our EPID scheme in Section IV. In Section V, we recommend two choices of elliptic curves and security parameters, analyze the efficiency of our scheme, and in the end compare our scheme with the previous EPID scheme and the related schemes. Finally, we conclude the paper in Section VI.

II. Specification and Security Model of EPID

In the rest of this paper, we use the following notations. Let S be a finite set, $x \leftarrow S$ denotes that x is chosen uniformly at random from S. Let $b \leftarrow A(a)$ denote an algorithm A that is given input a and outputs b. Let $\langle c, d\rangle \leftarrow P_{A,B}\langle a, b\rangle$ denote an interactive protocol between A and B, where A inputs a and B inputs b; in the end, A obtains c and B obtains d.

A. Specification of EPID

In an EPID scheme, there are four types of entities: an issuer $\mathcal{I}$, a revocation manager $\mathcal{R}$, platforms $\mathcal{P}$, and verifiers $\mathcal{V}$. There are two revocation lists managed by $\mathcal{R}$: a private-key based revocation list, denoted as pRL, and a signature based revocation list, denoted as sRL. An EPID scheme has the following four algorithms Setup, Sign, Verify, and Revoke, and one interactive protocol Join.

Setup This setup algorithm for the issuer $\mathcal{I}$ takes a security parameter 1^k as input and outputs a group public key gpk and an issuing private key isk.

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

Join This join protocol is an interactive protocol between the issuer $\mathcal{I}$ and a platform $\mathcal{P}$. $\mathcal{I}$ is given the group public key gpk and the issuing private key isk. $\mathcal{P}$ is given gpk. In the end, $\mathcal{P}$ outputs a private key sk, while $\mathcal{I}$ outputs nothing.

$$\langle \perp, \mathtt{sk}\rangle \leftarrow \mathsf{Join}_{\mathcal{I},\mathcal{P}}\langle(\mathtt{gpk}, \mathtt{isk}), \mathtt{gpk}\rangle$$

Sign On input of the group public key gpk, a private key sk, a message m, and a signature based revocation list sRL, this sign algorithm outputs $\perp$ if sk has been revoked in sRL, or outputs a signature σ otherwise.

$$\perp/\sigma \leftarrow \mathsf{Sign}(\mathtt{gpk}, \mathtt{sk}, m, \mathtt{sRL})$$

The sRL is used by the platform to prove that it has not been revoked in sRL, i.e., it has not created any of the signatures in sRL.

Verify On input of the group public key gpk, a message m, a private-key based revocation list pRL, a signature based revocation list sRL, and a signature σ, this verify algorithm outputs valid, invalid, or revoked.

$$\mathtt{valid}/\mathtt{invalid}/\mathtt{revoked} \leftarrow \mathsf{Verify}(\mathtt{gpk}, m, \mathtt{pRL}, \mathtt{sRL}, \sigma)$$

Note that the verifier needs to use the same sRL that is used in the signature creation to verify the signature, otherwise the verification would fail. It is not an issue in practice as the verifier will send the latest sRL to the platform as a challenge. Given a signature already created, the verifier cannot later verify it against a newer version of sRL, however, he can verify it against any versions of pRL.

Revoke There are two types of revocations.

Private-key based revocation: Given the group public key gpk and a private key sk, $\mathcal{R}$ updates pRL by adding sk to pRL.

$$\mathtt{pRL} \leftarrow \mathsf{Revoke}(\mathtt{gpk}, \mathtt{pRL}, \mathtt{sk})$$

Signature based revocation: Given the group public key gpk, a message m, and a signature σ on m, $\mathcal{R}$ updates sRL by placing σ to sRL after verifying σ.

$$\mathtt{sRL} \leftarrow \mathsf{Revoke}(\mathtt{gpk}, \mathtt{pRL}, \mathtt{sRL}, m, \sigma)$$

In the usage model, the private-key based revocation is used when a platform has been corrupted by the adversary, i.e., a private key gets extracted from the secure storage of the platform and published widely. The signature based revocation is used when $\mathcal{R}$ identifies that a platform $\mathcal{P}$ has been corrupted, but has not obtained $\mathcal{P}$'s private key.

For private-key based revocation, the revocation list is not sent to the platform. This revocation method is known in the literature as verifier-local revocation [7] and has also been used in the DAA schemes [10], [11]. One implication of this revocation method is that, given a signature σ on a message m and a private key sk, a verifier can easily determine whether σ was generated using sk by putting sk into pRL. Another implication is that, once a platform has been revoked in pRL, it loses its privacy and anonymity as all its previous signatures can be traced. We intentionally design this feature to penalize the revoked platform. Recall that, the issuer or the revocation manager cannot revoke a platform in pRL unless the platform's private key has been extracted and revealed by the attacker. In other words, the issuer cannot arbitrarily invade a platform's privacy by revoking the platform into pRL as he does not know the platform's private key.

B. Security Definition of EPID

An EPID scheme is secure if it satisfies the following three requirements: correctness, anonymity for unrevoked platforms, and unforgeability.

1) Correctness: Loosely speaking, the correctness requirement states that, every signature generated by a platform can be verified as valid, except when the platform is revoked. Formally speaking, let Σ_i be the set of all signatures generated by the platform $\mathcal{P}_i$, we have

$$\begin{aligned}&\sigma \leftarrow \mathsf{Sign}(\mathtt{gpk}, \mathtt{sk}_i, m, \mathtt{sRL}),\\ &\mathsf{Verify}(\mathtt{gpk}, m, \mathtt{pRL}, \mathtt{sRL}, \sigma) = \mathtt{valid}\\ &\iff (\mathtt{sk}_i \notin \mathtt{pRL}) \wedge (\Sigma_i \cap \mathtt{sRL} = \emptyset)\end{aligned}$$

2) Anonymity for Unrevoked Platforms: An EPID scheme satisfies the anonymity property for unrevoked platforms if no adversary can win the following anonymity game. In the anonymity game, the goal of the adversary is to determine which one of two private keys was used in generating a signature. As mentioned earlier, given a signature and a private key, the adversary could determine whether the signature was generated using the private key, thus the adversary should not be given access to either key. The anonymity game between a challenger and an adversary $\mathcal{A}$ is defined as follows.

1) Setup. The adversary $\mathcal{A}$ runs $(\mathtt{gpk}, \mathtt{isk}) \leftarrow \mathsf{Setup}(1^k)$ and sends $\mathtt{gpk}$ to the challenger.
2) Queries. The adversary $\mathcal{A}$ can make the following queries to the challenger.
   a) Join. $\mathcal{A}$ requests for creating a new platform $\mathcal{P}_i$. The challenger makes sure that i has not been requested before and then runs the join protocol as $\mathcal{P}_i$ with $\mathcal{A}$ as the issuer. In the end, the challenger obtains $\mathtt{sk}_i$.
   b) Sign. $\mathcal{A}$ chooses a subset of the signatures obtained from the challenger as $\mathtt{sRL}$[1]. $\mathcal{A}$ may add signatures that he computes to $\mathtt{sRL}$. $\mathcal{A}$ requests a signature on a message m with $\mathtt{sRL}$ for platform $\mathcal{P}_i$. The challenger makes sure $\mathcal{P}_i$ has been created, computes $\sigma \leftarrow \mathsf{Sign}(\mathtt{gpk}, \mathtt{sk}_i, m, \mathtt{sRL})$, and returns σ to $\mathcal{A}$.
   c) Corrupt. $\mathcal{A}$ requests the private key of $\mathcal{P}_i$. The challenger makes sure $\mathcal{P}_i$ has been created and then responds with $\mathtt{sk}_i$.
3) Challenge. $\mathcal{A}$ outputs a message m, a subset of the signatures obtained from the challenger as $\mathtt{sRL}$, and two indices i_0 and i_1. $\mathcal{A}$ must have not made a corruption query on either index and $\mathtt{sRL}$ cannot include signatures from either $\mathcal{P}_{i_0}$ or $\mathcal{P}_{i_1}$. The challenger chooses a random bit $b \leftarrow \{0,1\}$, computes a signature $\sigma^* \leftarrow \mathsf{Sign}(\mathtt{gpk}, \mathtt{sk}_{i_b}, m, \mathtt{sRL})$, and sends σ^* to $\mathcal{A}$.
4) Restricted Queries. After the challenge phase, $\mathcal{A}$ can make additional queries to the challenger, restricted as follows.
   a) Join. $\mathcal{A}$ can make join queries as before.
   b) Sign. As before, except that $\mathcal{A}$ cannot include σ^* in $\mathtt{sRL}$.
   c) Corrupt. As before, but $\mathcal{A}$ cannot make corrupt queries at i_0 and i_1.
5) Output. Finally, $\mathcal{A}$ outputs a bit b'. The adversary wins if $b' = b$.

We define $\mathcal{A}$'s advantage in winning the anonymity game as $|\Pr[b = b'] - 1/2|$. The probability is taken over the coin tosses of $\mathcal{A}$, of the randomized setup, join, and sign algorithms, and over the choice of b.

Definition 1: An EPID scheme is anonymous for unrevoked platforms, if for every probabilistic polynomial-time adversary $\mathcal{A}$, the advantage in winning the anonymity game is negligible.

3) Unforgeability: We say that an EPID scheme is unforgeable if no adversary can win the following unforgeability game. In the unforgeability game, the adversary's goal is to forge a valid signature, given that all private keys known to the adversary have been revoked. The traceability game between a challenger and an adversary $\mathcal{A}$ is defined as follows.

[1] $\mathcal{A}$ may choose a different $\mathtt{sRL}$ for each sign query.

1) Setup. The challenger runs $(\mathtt{gpk}, \mathtt{isk}) \leftarrow \mathsf{Setup}(1^k)$ and sends $\mathtt{gpk}$ to the adversary $\mathcal{A}$. The challenger sets $U := \emptyset$, the set of platforms controlled by the adversary.
2) Queries. The adversary $\mathcal{A}$ can make the following queries to the challenger.
   a) Join. $\mathcal{A}$ requests for creating a new platform $\mathcal{P}_i$. The challenger makes sure that i has not been join requested before. There are two cases as follows: (1) the challenger and $\mathcal{A}$ run $\langle \perp, \mathtt{sk}_i \rangle \leftarrow \mathsf{Join}_{\mathcal{I},\mathcal{A}}\langle(\mathtt{gpk}, \mathtt{isk}), \mathtt{gpk}\rangle$, where $\mathcal{A}$ gets $\mathtt{sk}_i$ from the join protocol. $\mathcal{A}$ sends $\mathtt{sk}_i$ to the challenger who appends i to U, or (2) the challenger runs locally the join protocol and generates $\mathtt{sk}_i$. Note that $\mathcal{A}$ does not learn $\mathtt{sk}_i$.
   b) Sign. $\mathcal{A}$ chooses a subset of the signatures obtained from the challenger as $\mathtt{sRL}$. $\mathcal{A}$ requests a signature on a message m with $\mathtt{sRL}$ for platform $\mathcal{P}_i$. The challenger makes sure $\mathcal{P}_i$ has been created, computes $\sigma \leftarrow \mathsf{Sign}(\mathtt{gpk}, \mathtt{sk}_i, m, \mathtt{sRL})$, and returns σ to $\mathcal{A}$.
   c) Corrupt. $\mathcal{A}$ requests the private key of $\mathcal{P}_i$. The challenger makes sure $\mathcal{P}_i$ has been created before, responds with $\mathtt{sk}_i$, and appends i to U.
3) Response. Finally, $\mathcal{A}$ outputs a message m^*, a private key based revocation list $\mathtt{pRL}^*$, a signature based revocation list $\mathtt{sRL}^*$, and a signature σ^*.

The adversary wins the game if: (1) $\mathsf{Verify}(\mathtt{gpk}, \mathtt{pRL}^*, \mathtt{sRL}^*, \sigma^*, m^*) = \mathtt{valid}$; (2) for every $i \in U$, either $\mathtt{sk}_i \in \mathtt{pRL}^*$ or one of the signatures created by $\mathcal{P}_i$ is placed in $\mathtt{sRL}^*$; and (3) $\mathcal{A}$ did not obtain σ^* by making a sign query on m^*. In other words, the adversary wins if he can forge a valid group signature that he has not queried the signature before, and all the private keys he knows have been revoked.

Definition 2: An EPID scheme is unforgeable, if for every probabilistic polynomial-time adversary $\mathcal{A}$, the probability in winning the unforgeability game is negligible.

Note that a signature scheme that satisfies the EPID security model above is unforgeable under chosen message attacks. This follows immediately from the unforgeability game.

III. Background and Building Blocks

A. Background on Bilinear Maps

We follow the notation of Boneh, Boyen, and Shacham [6] to review some background on bilinear maps. Let G_1 and G_2 to two multiplicative cyclic groups of prime order p. Let g_1 be a generator of G_1 and g_2 be a generator of G_2. We say $e : G_1 \times G_2 \to G_T$ is an admissible bilinear map, if it satisfies the following properties:

1) Bilinear. For all $u \in G_1, v \in G_2$, and for all $a, b \in \mathbb{Z}$, $e(u^a, v^b) = e(u, v)^{ab}$.
2) Non-degenerate. $e(g_1, g_2) \neq 1$ and is a generator of G_T.
3) Computable. There exists an efficient algorithm for computing $e(u, v)$ for any $u \in G_1, v \in G_2$.

We sometimes call the two groups (G_1, G_2) in the above a bilinear group pair. In the rest of this paper, we consider bilinear maps $e : G_1 \times G_2 \to G_T$ where G_1, G_2, and G_T are multiplicative groups of prime order p. We could set $G_1 = G_2$. However, we allow for the more general case where $G_1 \neq G_2$ so that we are not limited to choose supersingular elliptic curves, this allows us to take advantage of certain families of elliptic curves in order to obtain the shortest possible private keys and group signatures.

B. Cryptographic Assumptions

1) Strong Diffie-Hellman Assumption: Let G_1 and G_2 be two cyclic groups of prime order p, respectively, generated by g_1 and g_2. The q-Strong Diffie-Hellman (q-SDH) problem in (G_1, G_2) is defined as follows: Given a $(q+3)$-tuple of elements $(g_1, g_1^{\gamma}, \ldots, g_1^{(\gamma^q)}, g_2, g_2^{\gamma})$ as input, output a pair $(g_1^{1/(\gamma+x)}, x)$ where $x \in \mathbb{Z}_p^*$. An algorithm $\mathcal{A}$ has advantage ϵ in solving q-SDH problem in (G_1, G_2) if

$$\Pr\left[\mathcal{A}(g_1, g_1^{\gamma}, \ldots, g_1^{(\gamma^q)}, g_2, g_2^{\gamma}) = (g_1^{1/(\gamma+x)}, x)\right] \geq \epsilon$$

where the probability is over the random choice of γ and the random bits of $\mathcal{A}$.

Definition 3: We say that the (q, t, ϵ)-SDH assumption holds in (G_1, G_2) if no t-time algorithm has advantage at least ϵ in solving the q-SDH problem.

The q-SDH assumption was used by Boneh and Boyen [5] to construct a short signature scheme without random oracles and was shown in the same paper that q-SDH assumption holds in the generic group in the sense of Shoup [35]. The q-SDH assumption was later used in [6] for constructing a short group signature scheme. The security of the SDH problem was studied by Cheon [27].

2) Decisional Diffie-Hellman Assumption: Let G, generated by g, be a cyclic group of prime order p. The Decisional Diffie-Hellman (DDH) problem in G is defined as follows: Given a tuple of elements (g, g^a, g^b, g^c) as input, output 1 if $c = ab$ and 0 otherwise. An algorithm $\mathcal{A}$ has advantage ϵ in solving DDH problem in G if

$$|\Pr\left[g \leftarrow G, a, b \leftarrow \mathbb{Z}_p : \mathcal{A}(g, g^a, g^b, g^{ab}) = 1\right] - \\ \Pr\left[g \leftarrow G, a, b, c \leftarrow \mathbb{Z}_p : \mathcal{A}(g, g^a, g^b, g^c) = 1\right]| \geq \epsilon$$

where the probability is over the uniform random choice of the parameters to $\mathcal{A}$ and over the random bits of $\mathcal{A}$.

Definition 4: We say that the (t, ϵ)-DDH assumption holds in G if no t-time algorithm has advantage at least ϵ in solving the DDH problem in G.

IV. Our EPID Scheme

There are four types of entities in our construction of EPID: an issuer $\mathcal{I}$, a revocation manager $\mathcal{R}$, platforms $\mathcal{P}$, and verifiers $\mathcal{V}$. Our EPID scheme has the following algorithms Setup, Sign, Verify, and Revoke and one interactive protocol Join which are defined as follows.

Setup: Given 1^k, this algorithm chooses a bilinear group pair (G_1, G_2) of prime order p and a bilinear map function $e : G_1 \times G_2 \to G_T$. It also chooses a cyclic group G_3 of order p in which the DDH problem is hard. Let g_1, g_2, g_3 be the generators of G_1, G_2, and G_3 respectively. It chooses $h_1, h_2 \leftarrow G_1$, $\gamma \leftarrow \mathbb{Z}_p^*$, and computes $w := g_2^{\gamma}$. This algorithm outputs

$$(\texttt{gpk}, \texttt{isk}) := ((e, p, G_1, G_2, G_3, g_1, g_2, g_3, h_1, h_2, w), \gamma)$$

Let $H : \{0,1\}^* \to \mathbb{Z}_p$ be a collision resistant hash function. We treat H as a random oracle in the proof of security. In the rest of this paper, we use T_1, T_2, T_3, T_4 to denote $e(g_1, g_2)$, $e(h_1, g_2)$, $e(h_2, g_2)$, and $e(h_2, w)$, respectively. T_1, T_2, T_3, T_4 can be pre-computed by the platforms and verifiers.

Join: The join protocol is performed by a platform $\mathcal{P}$ and the issuer $\mathcal{I}$. $\mathcal{P}$ takes $\texttt{gpk}$ as input and $\mathcal{I}$ has $\texttt{gpk}$ and $\texttt{isk}$. The protocol has the following steps:

1) $\mathcal{P}$ chooses at random $f, y' \leftarrow \mathbb{Z}_p$ and computes $C := h_1^f \cdot h_2^{y'}$.
2) $\mathcal{P}$ sends C to $\mathcal{I}$, and performs the following proof of knowledge to $\mathcal{I}$

$$PK\{(f, y') \,:\, h_1^f \cdot h_2^{y'} = C\}$$

We can use the standard zero-knowledge proof protocols such as [23], [34]. Because the security proof requires rewinding to extract f from an adversarial platform, this step can only be run sequentially. To support concurrent join, we could use verifiable encryption [20] of the f value or use the concurrent join technique described in [29] with some loss of efficiency.
3) $\mathcal{I}$ chooses at random $x \leftarrow \mathbb{Z}_p$ and $y'' \leftarrow \mathbb{Z}_p$, and computes

$$A := (g_1 \cdot C \cdot h_2^{y''})^{1/(x+\gamma)}.$$

4) $\mathcal{I}$ sends (A, x, y'') to the platform.
5) $\mathcal{P}$ computes $y := y' + y'' \pmod{p}$ and verifies that $e(A, wg_2^x) = e(g_1 h_1^f h_2^y, g_2)$.
6) $\mathcal{P}$ outputs $\texttt{sk} := (A, x, y, f)$ where (A, x, y) is a membership certificate on f.

Sign: On input of $\texttt{gpk}$, $\texttt{sk} = (A, x, y, f)$, a message $m \in \{0,1\}^*$, and a signature based revocation list $\texttt{sRL}$, this sign algorithm has the following steps:

1) It chooses $B \leftarrow G_3$ and computes $K := B^f$.
2) It chooses $a \leftarrow \mathbb{Z}_p$, computes that $b := y + ax \bmod p$ and $T := A \cdot h_2^a$.
3) It runs the following signature of knowledge protocol

$$SPK\{(x, f, a, b) \,:\, B^f = K \wedge \\ e(T, g_2)^{-x} \cdot T_2^f \cdot T_3^b \cdot T_4^a = e(T, w)/T_1\}(m).$$

a) It randomly picks

$$r_x \leftarrow \mathbb{Z}_p, \quad r_f \leftarrow \mathbb{Z}_p, \quad r_a \leftarrow \mathbb{Z}_p, \quad r_b \leftarrow \mathbb{Z}_p.$$

b) It computes

$$\begin{aligned} R_1 &:= B^{r_f}, \\ R_2 &:= e(T,g_2)^{-r_x} \cdot T_2^{r_f} \cdot T_3^{r_b} \cdot T_4^{r_a}, \\ &:= e(A,g_2)^{-r_x} \cdot T_2^{r_f} \cdot T_3^{r_b - ar_x} \cdot T_4^{r_a}. \end{aligned}$$

c) It then computes

$$c := H(\mathtt{gpk}, B, K, T, R_1, R_2, m).$$

d) It computes in $\mathbb{Z}_p$

$$\begin{aligned} s_x &:= r_x + cx, & s_f &:= r_f + cf, \\ s_a &:= r_a + ca, & s_b &:= r_b + cb. \end{aligned}$$

4) It sets $\sigma_0 := (B, K, T, c, s_x, s_f, s_a, s_b)$.

5) Let $\mathtt{sRL} = \{(B_1, K_1), \ldots, (B_{n_2}, K_{n_2})\}$. For $i = 1, \ldots, n_2$, it proves in zero-knowledge that $\mathtt{sk}$ has not been revoked in $\mathtt{sRL}$, i.e., computes

$$\sigma_i := \mathit{SPK}\{(f) \ : \ K = B^f \ \wedge \ K_i \neq B_i^f\}(m).$$

We can use the zero-knowledge proof protocol from Camenisch and Shoup [20]. Note that we could batch all the n_2 zero-knowledge proofs together as follows using the techniques in [12], [39] to achieve additional efficiency, i.e., computes

$$\begin{aligned} \mathit{SPK}\{(f) \ : \ & K = B^f \ \wedge \\ & K_1 \neq B_1^f \ \wedge \ \ldots \ \wedge \ K_{n_2} \neq B_{n_2}^f\}(m). \end{aligned}$$

For the simplicity, we assume the sign algorithm computes n_2 individual zero-knowledge proofs.

6) If any of the zero-knowledge proofs in the previous step fails, it outputs $\sigma := \perp$.

7) It outputs the signature $\sigma := (\sigma_0, \sigma_1, \ldots, \sigma_{n_2})$.

Verify: On input of $\mathtt{gpk}$, a message m, a private-key based revocation list $\mathtt{pRL}$, a signature based revocation list $\mathtt{sRL}$, and a signature σ, the verify algorithm has the following steps:

1) Let $\sigma = (\sigma_0, \sigma_1, \ldots, \sigma_{n_2})$, where $\sigma_0 = (B, K, T, c, s_x, s_f, s_a, s_b)$.

2) It first verifies that

$$B, K \overset{?}{\in} G_3, \qquad T \overset{?}{\in} G_1, \qquad s_x, s_f, s_a, s_b \overset{?}{\in} \mathbb{Z}_p.$$

3) It computes

$$\begin{aligned} \hat{R}_1 &:= B^{s_f} \cdot K^{-c}, \\ \hat{R}_2 &:= e(T,g_2)^{-s_x} \cdot T_2^{s_f} \cdot T_3^{s_b} \cdot T_4^{s_a} \cdot (T_1/e(T,w))^c, \\ &:= e(T, g_2^{-s_x} w^{-c}) \cdot T_2^{s_f} \cdot T_3^{s_b} \cdot T_4^{s_a} \cdot T_1^c. \end{aligned}$$

4) It verifies that

$$c \overset{?}{=} H(\mathtt{gpk}, B, K, T, \hat{R}_1, \hat{R}_2, m).$$

5) If any of the above verification fails, it outputs $\mathtt{invalid}$ and quits.

6) Let $\mathtt{pRL} = \{f_1, \ldots, f_{n_1}\}$. For $i = 1, \ldots, n_1$, it verifies that $K \neq B^{f_i}$.

7) Let $\mathtt{sRL} = \{(B_1, K_1), \ldots, (B_{n_2}, K_{n_2})\}$. For $i = 1, \ldots, n_2$, it verifies that σ_i is indeed a valid zero-knowledge proof

$$\mathit{SPK}\{(f) \ : \ K = B^f \ \wedge \ K_i \neq B_i^f\}(m).$$

8) If any of the above two steps fails, it outputs $\mathtt{revoked}$, otherwise, outputs $\mathtt{valid}$.

Revoke: Initially, both revocation lists are empty, i.e., $\mathtt{pRL} := \emptyset$ and $\mathtt{sRL} := \emptyset$.

1) *Private-key based revocation*: Given $\mathtt{gpk}$, $\mathtt{pRL}$, and a private key $\mathtt{sk} = (A, x, y, f)$ to be revoked, $\mathcal{R}$ updates $\mathtt{pRL}$ as follows: $\mathcal{R}$ verifies the correctness of $\mathtt{sk}$ by checking whether the equation $e(A, g_2^x w) = e(g_1 h_1^f h_2^y, g_2)$ holds, then appends f to $\mathtt{pRL}$.

2) *Signature based revocation*: Given $\mathtt{gpk}$, $\mathtt{pRL}$, $\mathtt{sRL}$, a message m, and corresponding signature σ, $\mathcal{R}$ updates $\mathtt{sRL}$ as follows: Let $\sigma = (\sigma_0, \sigma_1, \ldots, \sigma_{n_2})$. $\mathcal{R}$ first verifies that σ is a valid signature on m, i.e., checks $\mathsf{Verify}(\mathtt{gpk}, m, \mathtt{pRL}, \emptyset, \sigma_0) = \mathtt{valid}$, then appends (B, K) in σ_0 to $\mathtt{sRL}$.

Observe that steps 2-4 of the join algorithm and steps 2-4 of the verify algorithm indeed form a signature of knowledge

$$\mathit{PK}\{(A, x, y, f) \ : \ e(A, g_2^x w) = e(g_1 h_1^f h_2^y, g_2) \ \wedge \ K = B^f\}.$$

We first show the correctness of the signature of knowledge protocol. Let (A, x, y, f) be a private key that satisfies the equations $e(A, g_2^x w) = e(g_1 h_1^f h_2^y, g_2)$ and $B^f = K$. Step 3 of the sign algorithm is a standard way of proving the following two equations hold:

$$B^f = K, \qquad e(T, g_2)^{-x} \cdot T_2^f \cdot T_3^b \cdot T_4^a = e(T, w)/T_1.$$

The second equation holds because

$$\begin{aligned} & e(T, w) \cdot e(T, g_2)^x \\ &= e(T, g_2^x w) = e(A h_2^a, g_2^x w) = e(A, g_2^x w) \cdot e(h_2^a, g_2^x w) \\ &= e(g_1 h_1^f h_2^y, g_2) \cdot e(h_2^a, w) \cdot e(h_2^a, g_2^x) \\ &= e(g_1, g_2) \cdot e(h_1, g_2)^f \cdot e(h_2, g_2)^y \cdot e(h_2, w)^a \cdot e(h_2, g_2)^{ax} \\ &= e(g_1, g_2) \cdot e(h_1, g_2)^f \cdot e(h_2, g_2)^{y+ax} \cdot e(h_2, w)^a \\ &= e(g_1, g_2) \cdot e(h_1, g_2)^f \cdot e(h_2, g_2)^b \cdot e(h_2, w)^a \\ &= T_1 \cdot T_2^f \cdot T_3^b \cdot T_4^a. \end{aligned}$$

Theorem 1: The EPID scheme is correct.

Theorem 2: The EPID scheme is anonymous in the random oracle model under the DDH assumption in G_3.

Theorem 3: The EPID scheme is unforgeable in the random oracle model under the q-SDH assumption in (G_1, G_2).

Due to the space limit, the detailed proofs of the above theorems are given in the technical report version of this paper [14].

V. Implementation of the EPID Scheme

In this section, we first suggest two choices of elliptic curves and security parameters. We then analyze the efficiency of the EPID scheme and compare our scheme with the original EPID scheme in [12] and other related schemes.

A. Choices of Elliptic Curves and Security Parameters

We suggest the following two choices of security parameters.

1) To achieve 80-bit security level, we choose $k = 6$ and use a family of non-supersingular elliptic curves defined by Miyaji et al. [33] for Tate pairing. As in [6], we choose p and q to be 170-bit prime integers. Each element in G_1 can be represented by a 171-bit string. We then choose a 170-bit prime q' and construct G_3 as an order-p cyclic subgroup of group $E(\mathbb{F}_{q'})$. The security strength of this setting is approximately the same as a standard 1024-bit RSA algorithm.
2) To achieve 128-bit security level, as suggested by Koblitz and Menezes [31], the minimum size of G_1 is 256-bit and the minimum size of $\mathbb{F}^*_{q^k}$ should be at least 3072-bit. We choose embedding degree $k = 12$ for Tate pairing and use a method developed by Barreto and Naehrig [4]. We choose p and q to be 256-bit prime integers and choose G_3 to be an elliptic curve group of order p. The security strength of this setting is approximately the same as a standard 3072-bit RSA algorithm.

B. Efficiency of the EPID Scheme

We now summarize the efficiency of the EPID scheme. In what follows, we use EXP to denote an exponentiation or a multi-exponentiation which has similar efficiency.

- For parameters with 80-bit security, p is a 170-bit prime, each element in G_1 or G_3 is 171-bit in length. The size of the private key is 681 bits or 86 bytes. The size of the signature is 1363 bits or 171 bytes.
- For parameters with 128-bit security, p is a 256-bit prime, each element in G_1 or G_3 is 257-bit in length. The size of the private key is 1025 bits or 129 bytes. The size of the signature is 2051 bits or 257 bytes.
- To sign a signature, the platform can pre-compute T_2, T_3, T_4, and $e(A, g_2)$. The sign algorithm only takes 1 EXP in G_1, 2 EXPs in G_3, 1 EXP in G_T.
- To verify a signature, the verifier can pre-compute T_1, T_2, T_3, and T_4. The verify algorithm takes 1 EXP in G_2, 1 EXP in G_3, 1 EXP in G_T, and 1 pairing operation.
- For each item in pRL, the platform needs to do nothing while the verifier needs to perform 1 EXP in G_3.
- For each item in sRL, the platform needs to compute 3 EXPs in G_3 and the verifier needs to perform 2 EXPs in G_3 if we use the zero-knowledge proof in [20]. If we apply the batched zero-knowledge proof in [39], the platform needs to compute 2 EXPs and the verifier needs to compute 1 EXP.

C. Comparison with EPID, DAA, and BLAC Schemes

Brickell and Li developed an EPID scheme from the strong RSA assumption [12]. Their scheme is derived from the original DAA scheme in [10]. Using 2048-bit RSA modulus which has about 100-bit security, the length of private key is 670 bytes and the size of a signature is 2800 bytes in their scheme. Our EPID scheme offers significant advantage over the EPID scheme in [12] due to much smaller private key size, as hardware devices typically have limited size of secure storage to hide the private key.

	private key size	signature size
Our scheme 80-bit security	86 bytes	171 bytes
Our scheme 128-bit security	129 bytes	257 bytes
EPID scheme [12]	670 bytes	2800 bytes

TABLE I
A COMPARISON BETWEEN OUR PAIRING EPID SCHEME AND THE RSA BASED EPID SCHEME

Brickell, Chen, and Li [11] developed the first pairing-base DAA scheme. Their scheme is based on Camenisch and Lysyanskaya's pairing based group signature scheme [19]. Chen, Morrissey, and Smart further optimized the DAA scheme [26] by transferring the underlying pairing groups from the symmetric to the asymmetric settings. We now show that our EPID scheme is more efficient than the pairing based DAA schemes [11], [26]. See Table II for details.

Note that Chen [25] and Brickell and Li [15] recently proposed DAA schemes built on top of this paper, respectively. Both DAA schemes [25], [15] have similar complexity as our EPID scheme. Again, both DAA schemes focus on the TPM implementation and reduce the TPM workload by outsourcing majority of the computation to the host.

Tsang et al. recently proposed a Blacklistable Anonymous Credentials (BLAC) scheme in which a misbehaved user can be revoked based on his previous signatures [39]. The BLAC scheme [39] shares a similarity of construction as our EPID scheme. However, our EPID scheme is about twice efficient than [39] in both signature creation and verification and size of the signatures. Regarding to revocation, the BLAC scheme does not support private-key based revocation and has the same efficiency for signature based revocation. If the size of sRL is 10 (recall that hardware revocation is a rare event), our EPID scheme is still around 25% more efficient than the BLAC scheme in sign and verify algorithms.

VI. Conclusion

We have presented a new EPID scheme from bilinear pairing. Our EPID scheme is efficient (if revocation list is small) and provably secure in the random oracle model under the q-SDH assumption and the DDH assumption. It is a good candidate for hardware remote authentication and attestation. We expect revocation to be a rare event due to the difficulty of hardware attacks that are not scalable. The future work includes developing new EPID schemes with more efficient revocation while maintaining maximum user's privacy such that EPID scheme can be used beyond hardware authentication and used as a generic anonymous authentication method.

References

[1] G. Ateniese, J. Camenisch, M. Joye, and G. Tsudik. A practical and provably secure coalition-resistant group signature scheme. In *Advances in Cryptology — CRYPTO '00*, volume 1880 of *LNCS*, pages 255–270. Springer, 2000.

	private key size	signature size	sign	verify
Our EPID scheme	86 bytes	171 bytes	4 EXP	3 EXP + 1 P
BCL DAA scheme [11]	213 bytes	512 bytes	10 EXP	2 EXP + 5 P
CMS DAA scheme [26]	86 bytes	148 bytes	8 EXP + 1 P	1 EXP + 5 P
DAA schemes based on this paper [25], [15]	64 bytes	171 bytes	5 EXP	3 EXP + 1 P
BLAC scheme [39]	86 bytes	320 bytes	10 EXP	7 EXP + 2 P

TABLE II
A COMPARISON BETWEEN OUR EPID SCHEME, THE PAIRING BASED DAA SCHEMES [11], [26], [25], [15], AND THE BLAC SCHEME [39] WITH 80-BIT SECURITY LEVEL, WHERE EXP DENOTES MULTI-EXPONENTIATION AND P DENOTES A PAIRING OPERATION.

[2] G. Ateniese, D. X. Song, and G. Tsudik. Quasi-efficient revocation in group signatures. In *Proceedings of the 6th International Conference on Financial Cryptography*, volume 2357 of *LNCS*, pages 183–197. Springer, 2002.

[3] M. Backes, M. Maffei, and D. Unruh. Zero-knowledge in the applied pi-calculus and automated verification of the direct anonymous attestation protocol. In *Proceedings of IEEE Symposium on Security and Privacy*, pages 202–215. IEEE Computer Society, 2008.

[4] P. S. L. M. Barreto and M. Naehrig. Pairing-friendly elliptic curves of prime order. In *Proceedings of the 12th International Workshop on Selected Areas in Cryptography*, volume 3897 of *LNCS*, pages 319–331. Springer, 2005.

[5] D. Boneh and X. Boyen. Short signatures without random oracles. In *Advances in Cryptology — EUROCRYPT '04*, volume 3027 of *LNCS*, pages 56–73. Springer, 2004.

[6] D. Boneh, X. Boyen, and H. Shacham. Short group signatures. In *Advances in Cryptology — CRYPTO '04*, volume 3152 of *LNCS*, pages 41–55. Springer, 2004.

[7] D. Boneh and H. Shacham. Group signatures with verifier-local revocation. In *Proceedings of 11th ACM Conference on Computer and Communications Security*, pages 168–177, Oct. 2004.

[8] S. A. Brands. *Rethinking Public Key Infrastructures and Digital Certificates: Building in Privacy*. MIT Press, Aug. 2000.

[9] E. Bresson and J. Stern. Efficient revocation in group signatures. In *Proceedings of the 4th International Workshop on Practice and Theory in Public Key Cryptography*, pages 190–206. Springer, 2001.

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

[11] E. Brickell, L. Chen, and J. Li. A new direct anonymous attestation scheme from bilinear maps. In *Proceedings of 1st International Conference on Trusted Computing*, volume 4968 of *LNCS*, pages 166–178. Springer, 2008.

[12] E. Brickell and J. Li. Enhanced Privacy ID: A direct anonymous attestation scheme with enhanced revocation capabilities. In *Proceedings of the 6th ACM Workshop on Privacy in the Electronic Society*, pages 21–30. ACM Press, Oct. 2007.

[13] E. Brickell and J. Li. Enhanced Privacy ID: A remote anonymous attestation scheme for hardware devices. *Intel Technology Journal: Advances in Internet Security*, 13(2), 2009.

[14] E. Brickell and J. Li. Enhanced Privacy ID from bilinear pairing. Cryptology ePrint Archive, Report 2009/095, 2009. http://eprint.iacr.org/.

[15] E. Brickell and J. Li. A pairing-based DAA scheme further reducing TPM resources. In *Proceedings of 3rd International Conference on Trust and Trustworthy Computing*, pages 181–195. Springer, 2010.

[16] J. Camenisch and J. Groth. Group signatures: Better efficiency and new theoretical aspects. In *Proceedings of 4th International Conference on Security in Communication Networks*, volume 3352 of *LNCS*, pages 122–135. Springer, 2005.

[17] J. Camenisch and A. Lysyanskaya. An efficient system for non-transferable anonymous credentials with optional anonymity revocation. In *Advances in Cryptology — EUROCRYPT '01*, volume 2045 of *LNCS*, pages 93–118. Springer, 2001.

[18] J. Camenisch and A. Lysyanskaya. Dynamic accumulators and application to efficient revocation of anonymous credentials. In *Advances in Cryptology — CRYPTO '02*, volume 2442 of *LNCS*, pages 61–76. Springer, 2002.

[19] J. Camenisch and A. Lysyanskaya. Signature schemes and anonymous credentials from bilinear maps. In *Advances in Cryptology — CRYPTO '04*, volume 3152 of *LNCS*, pages 56–72. Springer, 2004.

[20] J. Camenisch and V. Shoup. Practical verifiable encryption and decryption of discrete logarithms. In *Advances in Cryptology — CRYPTO '03*, volume 2729 of *LNCS*, pages 126–144. Springer, 2003.

[21] J. Camenisch and M. Stadler. Efficient group signature schemes for large groups. In *Advances in Cryptology — CRYPTO '97*, volume 1296 of *LNCS*, pages 410–424. Springer, 1997.

[22] D. Chaum. Security without identification: Transaction systems to make big brother obsolete. *Communications of the ACM*, 28(10):1030–1044, 1985.

[23] D. Chaum, J.-H. Evertse, and J. van de Graaf. An improved protocol for demonstrating possession of discrete logarithms and some generalizations. In *Advances in Cryptology — EUROCRYPT '87*, volume 304 of *LNCS*, pages 127–141. Springer, 1987.

[24] D. Chaum and E. van Heyst. Group signatures. In *Advances in Cryptology — EUROCRYPT '91*, volume 547 of *LNCS*, pages 257–265. Springer, 1991.

[25] L. Chen. A DAA scheme requiring less TPM resources. In *Proceedings of the 5th China International Conference on Information Security and Cryptology*, LNCS. Springer, 2009.

[26] L. Chen, P. Morrissey, and N. P. Smart. Pairings in trusted computing. In *Proceedings of the 2nd Internation Conference on Pairing-Based Cryptography*, volume 5209 of *LNCS*, pages 1–17. Springer, 2008.

[27] J. H. Cheon. Security analysis of the strong diffie-hellman problem. In *Advances in Cryptology — EUROCRYPT '06*, volume 4004 of *LNCS*, pages 1–11. Springer, 2006.

[28] J. Furukawa and H. Imai. An efficient group signature scheme from bilinear maps. *IEICE Transactions*, 89-A(5):1328–1338, 2006.

[29] A. Kiayias and M. Yung. Group signatures with efficient concurrent join. In *Advances in Cryptology — EUROCRYPT '05*, volume 3494 of *LNCS*, pages 198–214. Springer, 2005.

[30] J. Kilian and E. Petrank. Identity escrow. In *Advances in Cryptology — CRYPTO '98*, volume 1642 of *LNCS*, pages 169–185. Springer, 1998.

[31] N. Koblitz and A. Menezes. Pairing-based cryptography at high security levels. In *Proceedings of the 10th IMA International Conference on Cryptography and Coding*, volume 3796 of *LNCS*, pages 13–36. Springer, 2005.

[32] A. Leung and C. J. Mitchell. Ninja: Non identity based, privacy preserving authentication for ubiquitous environments. In *Proceedings of 9th International Conference on Ubiquitous Computing*, volume 4717 of *LNCS*, pages 73–90. Springer, 2007.

[33] A. Miyaji, M. Nakabayashi, and S. Takano. New explicit conditions of elliptic curve traces for FR-reduction. *IEICE Transactions on Fundamentals*, E84-A(5):1234–1243, 2001.

[34] T. P. Pedersen. Non-interactive and information-theoretic secure verifiable secret sharing. In *Advances in Cryptology — CRYPTO '91*, volume 576 of *LNCS*, pages 129–140. Springer, 1991.

[35] V. Shoup. Lower bounds for discrete logarithms and related problems. In *Advances in Cryptology — EUROCRYPT '97*, volume 1233 of *LNCS*, pages 256–266. Springer, 1997.

[36] D. X. Song. Practical forward secure group signature schemes. In *Proceedings of the 8th ACM Conference on Computer and Communications Security*, pages 225–234. ACM Press, 2001.

[37] Trusted Computing Group. TCG TPM specification 1.2, 2003. Available at http://www.trustedcomputinggroup.org.

[38] Trusted Computing Group website. http://www.trustedcomputinggroup.org.

[39] P. P. Tsang, M. H. Au, A. Kapadia, and S. W. Smith. Blacklistable anonymous credentials: Blocking misbehaving users without TTPs. In *ACM Conference on Computer and Communications Security*, pages 72–81. ACM Press, 2007.