Enhanced Privacy ID from Bilinear Pairing

Ernie Brickell
Intel Corporation
ernie.brickell@intel.com

Jiangtao Li
Intel Corporation
jiangtao.li@intel.com

Abstract

Enhanced Privacy ID (EPID) is a cryptographic scheme that enables the remote authentication 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 signed, 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 with revocation. 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.

1 Introduction

Consider the following problem: a hardware device (e.g., a graphics chip, a trusted platform module, a mobile device, 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., 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 privacy concern as the device certificate can uniquely identify the device.

Brickell, Camenisch, and Chen [13] 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) [46] and was standardized in the TCG TPM Specification version 1.2 [45].

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 [15] 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.

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 signing 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. 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 2.

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 [15]. 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 short group signatures scheme [8]. 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 [15] and yet achieves a higher level of security.

1.1 Motivation for Signature Based Revocation

We now give 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 DRM keys to each platform. Only platform with valid EPID private key can obtain a unique DRM key from the provisioning server. If an attacker breaks one EPID private key, he could use the corrupted EPID private key to obtain DRM keys. If the attacker publishes the private key over the Internet, we can revoke the key. However, in practice, the attacker may embed the obtained DRM key in a media ripper software without publishing the EPID private key. Once we find the ripper software on the Internet and extract the DRM key in it, we can back trace to the EPID signature that was used to obtain the DRM 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 [13, 14], EPID scheme chooses not to have traceability from the revocation manager in order to provide *maximum privacy* for the platforms. Traceability provides the capability that a revocation manager can determine which platform generates which signatures without any acknowledgement from the user that is being traced. This is not desirable from a privacy perspective. With 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.

1.2 Related Work

The EPID scheme can be seemed as a group signature scheme [1, 8, 22, 26] 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 [11, 44, 18, 2, 9]. 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 [33], anonymous credential systems [17, 23], the pseudonym system of Brands [10].

As we mentioned early, EPID can be also seemed 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, as those features are more TPM specific. We could easily add those features back to EPID if necessary. After DAA was first introduced by Brickell, Camenisch, and Chen [13], it has drawn a lot of attention from both industry and cryptographic community (e.g., [16, 35, 4], to list a few). Brickell, Chen, and Li recently constructed the first pairing based DAA scheme [14]. Later Chen, Morrissey, and Smart [27] showed that the DAA scheme in [14] can be further optimized by transferring the underlying pairing groups from the symmetric to the asymmetric settings.

Tsang et al. recently proposed a Blacklistable Anonymous Credentials (BLAC) scheme in which a misbehaved user can be revoked based on his previous signatures [47]. Their BLAC scheme has a similar revocation capability as the signature based revocation in our EPID scheme. Their construction is similar to our EPID construction in this paper. Note that our EPID scheme is at least 30% more efficient than the BLAC scheme [47] in both signature creation and verification and size of the signatures. See Section 5.4 for the comparison between our EPID scheme and the BLAC scheme.

1.3 Organization of This Paper

Rest of this document is organized as follows. We give a formal specification of EPID and present the corresponding security model in Section 2. We then define our notations, present security assumptions, and briefly review some previously known cryptographic techniques in Section 3. We present our EPID scheme in Section 4. In Section 5, 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 pairing based DAA schemes. Finally, we give the security proof of our construction in Section 6.

2 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.

2.1 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 `Priv-RL`, and a signature based revocation list, denoted as `Sig-RL`. 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`.

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

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

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

Sign : On input of the group public key gpk, a private key $\mathtt{sk}_i$, a message m, and a signature based revocation list `Sig-RL`, this sign algorithm outputs $\perp$ if $\mathtt{sk}_i$ has been revoked in `Sig-RL`, or outputs a signature σ otherwise.

$$\perp/\sigma \leftarrow \mathsf{Sign}(\mathsf{gpk}, \mathtt{sk}_i, m, \mathtt{Sig\text{-}RL})$$

Verify : On input of the group public key gpk, a message m, a private-key based revocation list `Priv-RL`, a signature based revocation list `Sig-RL`, and a signature σ, this verify algorithm outputs either `valid` or `invalid`. The latter output means either that σ is not a valid signature on m, or that the platform who generated σ has been revoked.

$$\mathtt{valid}/\mathtt{invalid} \leftarrow \mathsf{Verify}(\mathsf{gpk}, m, \mathtt{Priv\text{-}RL}, \mathtt{Sig\text{-}RL}, \sigma)$$

Revoke : There are two types of revocations.

1. Private-key based revocation: Given the group public key gpk and a private key $\mathtt{sk}_i$, $\mathcal{R}$ updates `Priv-RL` by adding $\mathtt{sk}_i$ to `Priv-RL`.

$$\mathtt{Priv\text{-}RL} \leftarrow \mathsf{Revoke}(\mathsf{gpk}, \mathtt{Priv\text{-}RL}, \mathtt{sk}_i)$$

2. Signature based revocation: Given the group public key gpk, a message m, and a signature σ on m, $\mathcal{R}$ updates `Sig-RL` by inserting σ into `Sig-RL` after verifying σ.

$$\mathtt{Sig\text{-}RL} \leftarrow \mathsf{Revoke}(\mathsf{gpk}, \mathtt{Priv\text{-}RL}, \mathtt{Sig\text{-}RL}, m, \sigma)$$

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

Observe that for private-key based revocation, the revocation list `Priv-RL` is not sent to the platform. This revocation method is known in the literature as verifier-local revocation [9] and has been used in the DAA schemes [13, 14]. One implication of this revocation method is that, given a signature σ on a message m, a private key $\mathtt{sk}_i$, one can easily determine whether σ was generated using $\mathtt{sk}_i$ as follows: first set `Priv-RL` to be empty and make sure σ is a valid signature on m, then revoke $\mathtt{sk}_i$ by setting $\mathtt{Priv\text{-}RL} = \{\mathtt{sk}_i\}$ and run the signature verification algorithm again. If the verification fails, it means that σ was created by $\mathtt{sk}_i$.

2.2 Security Definition of EPID

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

2.2.1 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

$$\mathsf{Verify}(\mathtt{gpk}, m, \mathtt{Priv\text{-}RL}, \mathtt{Sig\text{-}RL}, \mathsf{Sign}(\mathtt{gpk}, \mathtt{sk}_i, m, \mathtt{Sig\text{-}RL})) = \mathtt{valid} \\ \iff (\mathtt{sk}_i \notin \mathtt{Priv\text{-}RL}) \wedge (\Sigma_i \cap \mathtt{Sig\text{-}RL} = \emptyset)$$

2.2.2 Anonymity

We say that an EPID scheme satisfies the anonymity property 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 $\mathcal{A}$ computes $(\mathtt{gpk}, \mathtt{isk}) \leftarrow \mathsf{Setup}(1^k)$ 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 $\mathtt{Sig\text{-}RL}$[1]. $\mathcal{A}$ requests a signature on a message m with $\mathtt{Sig\text{-}RL}$ for platform $\mathcal{P}_i$. The challenger makes sure $\mathcal{P}_i$ has been created, computes $\sigma \leftarrow \mathsf{Sign}(\mathtt{gpk}, \mathtt{sk}_i, m, \mathtt{Sig\text{-}RL})$, 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 $\mathtt{Sig\text{-}RL}$, and two indices i_0 and i_1. $\mathcal{A}$ must have not made a corruption query on either index and $\mathtt{Sig\text{-}RL}$ cannot include any signatures from either $\mathcal{P}_{i_0}$ or $\mathcal{P}_{i_1}$. The challenger makes sure both $\mathcal{P}_{i_0}$ and $\mathcal{P}_{i_1}$ have been created, chooses a random bit $b \leftarrow \{0, 1\}$, computes a signature $\sigma^* \leftarrow \mathsf{Sign}(\mathtt{gpk}, \mathtt{sk}_{i_b}, m, \mathtt{Sig\text{-}RL})$, 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 $\mathtt{Sig\text{-}RL}$.
   (c) Corrupt. As before, but $\mathcal{A}$ cannot make corrupt queries at i_0 and i_1.

[1] $\mathcal{A}$ may choose a different $\mathtt{Sig\text{-}RL}$ for each sign query.

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, if for every probabilistic polynomial-time adversary $\mathcal{A}$, the advantage in winning the anonymity game is negligible.

2.2.3 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. Setup. The challenger computes $(\texttt{gpk}, \texttt{isk}) \leftarrow \mathsf{Setup}(1^k)$ and sends $\texttt{gpk}$ 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 the following two cases:
      i. The challenger and $\mathcal{A}$ run $\langle \perp, \texttt{sk}_i \rangle \leftarrow \mathsf{Join}_{\mathcal{I},\mathcal{A}}\langle (\texttt{gpk}, \texttt{isk}), \texttt{gpk} \rangle$, where $\mathcal{A}$ gets $\texttt{sk}_i$ from the join protocol. $\mathcal{A}$ sends $\texttt{sk}_i$ to the challenger who appends i to U.
      ii. The challenger runs locally the join protocol and generates $\texttt{sk}_i$.
   (b) Sign. $\mathcal{A}$ chooses a subset of the signatures obtained from the challenger as $\texttt{Sig-RL}$. $\mathcal{A}$ requests a signature on a message m with $\texttt{Sig-RL}$ for platform $\mathcal{P}_i$. The challenger makes sure $\mathcal{P}_i$ has been created, computes $\sigma \leftarrow \mathsf{Sign}(\texttt{gpk}, \texttt{sk}_i, m, \texttt{Sig-RL})$, 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 $\texttt{sk}_i$, and appends i to U.
3. Response. Finally, $\mathcal{A}$ outputs a message m^*, a private key based revocation list $\texttt{Priv-RL}^*$, a signature based revocation list $\texttt{Sig-RL}^*$, and a signature σ^*.

The adversary wins if: (1) $\mathsf{Verify}(\texttt{gpk}, \texttt{Priv-RL}^*, \texttt{Sig-RL}^*, \sigma^*, m^*) = \texttt{valid}$; (2) for every $i \in U$, either $\texttt{sk}_i \in \texttt{Priv-RL}^*$ or one of the signatures created by $\mathcal{P}_i$ is placed in $\texttt{Sig-RL}^*$; 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.

We 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.

3 Background and Building Blocks

In this section, we first review the concept of bilinear maps, then discuss some complexity assumptions related to our EPID scheme, and finally review some building blocks that shall be used in our construction.

3.1 Background on Bilinear Maps

We follow the notation of Boneh, Boyen, and Shacham [8] 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 $e : G_1 \times G_2 \rightarrow G_T$ 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 $e : G_1 \times G_2 \rightarrow G_T$ 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.

3.2 Cryptographic Assumptions

The security of our EPID construction relies on the Strong Diffie-Hellman (SDH) assumption and the Decisional Diffie-Hellman (DDH) assumption, where the SDH assumption is used for proving unforgeability of our EPID scheme and the DDH assumption is used for proving unlinkability of our scheme. We now state these two assumptions as follows:

3.2.1 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 [7] 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 [43]. The q-SDH assumption was later used in [8] for constructing a short group signature scheme. The security of the SDH problem was studied by Cheon [28].

3.2.2 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

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

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.*

There are many groups in which the DDH problem is believed to be intractable, and the best known algorithm for DDH is a full discrete log algorithm [6]. However, for a cyclic group with a symmetric pairing, the DDH problem is tractable [32]. In our paper, we build the EPID scheme on the assumption that the DDH problem is intractable in group G_3, where G_3 could be a standard elliptic curve group.

3.3 BBS+ Signature Scheme

In this subsection, we review a signature scheme that will be used in our EPID construction called BBS+ signature scheme [3]. This BBS+ signature scheme is a variant of Boneh-Boyen signature scheme [7]. The idea of constructing this BBS+ signature scheme is informally stated in [8, 20]. Au, Susilo, and Mu [3] formalize this idea and call this signature scheme BBS+ scheme. Let (G_1, G_2) be a bilinear group pair of some prime order p. Let $e : G_1 \times G_2 \rightarrow G_T$ be a computable bilinear pairing function. The BBS+ signature scheme is as follows:

Key Generation Select $g_1, h_1, h_2 \leftarrow G_1$, $g_2 \leftarrow G_2$, $\gamma \leftarrow \mathbb{Z}_p^*$, and compute $w := g_2^\gamma$. The public key is the tuple (g_1, g_2, h_1, h_2, w). The private key is γ.

Sign Given a public key (g_1, g_2, h_1, h_2, w) and the corresponding private key γ, a message $m \in \mathbb{Z}_p$, the sign algorithm chooses $x, y \leftarrow \mathbb{Z}_q$ and computes $A := (g_1 h_1^m h_2^y)^{1/(x+\gamma)}$. The signature on m is $\sigma := (A, x, y)$.

Verify Given a public key (g_1, g_2, h_1, h_2, w), a message m, and a signature $\sigma = (A, x, y)$, the verify algorithm verifies that $e(A, g_2^x w) = e(g_1 h_1^m h_2^y, g_2)$. If the equality holds, the signature is valid.

Lemma 1 *The BBS+ signature scheme is unforgeable against adaptive chosen message attacks under the q-SDH assumption.*

The proof of the above lemma is given by Au, Susilo, and Mu in [3]. The BBS+ signature scheme was used to construct a blacklistable anonymous credential scheme [47]. The same signature scheme was also used by Furukawa and Imai to build an efficient group signature scheme [30], although the BBS+ signature scheme was not explicitly stated.

3.4 Protocols for Proof of Knowledge

In our scheme we will use various protocols to prove knowledge of and relations among discrete logarithms. To describe these protocols, we use notation introduced by Camenisch and Stadler [22] for various proofs of knowledge of discrete logarithms and proofs of the validity of statements about discrete logarithms. For example,

$$PK\{(a, b) : y_1 = g_1^a h_1^b \ \wedge \ y_2 = g_2^a h_2^b\}$$

denotes a proof of knowledge of integers a and b such that $y_1 = g_1^a h_1^b$ and $y_2 = g_2^a h_2^b$ holds, where g_1 and h_1 are generators of some group G_1, and g_2 and h_2 are generators of some group G_2. The variables in the parenthesis denote the values the knowledge of which is being proved, while all other parameters are known to the verifier. Using this notation, a proof of knowledge protocol can be described without getting into all details.

In the random oracle model, such proof of knowledge protocols can be turned into signature schemes using the Fiat-Shamir heuristic [29, 41]. We use the notation $SPK\{(a) : y = z^a\}(m)$ to denote a signature on a message m obtained in this way.

In this paper, we use the following known proof of knowledge protocols:

- Proof of knowledge of discrete logarithms. A proof of knowledge of a discrete logarithm of an element $y \in G$ [42] with respect to a base z is denoted as $PK\{(a) : y = z^a\}$. A proof of knowledge of a representation of an element $y \in G$ with respect to several bases $z_1, \ldots, z_v \in G$ [24] is denoted $PK\{(a_1, \ldots, a_v) : y = z_1^{a_1} \cdot \ldots \cdot z_v^{a_v}\}$.
- Proof of knowledge of equality. A proof of equality of discrete logarithms of two group elements $y_1, y_2 \in G$ to the bases $z_1, z_2 \in G$, respectively, [25] is denoted $PK\{(a) : y_1 = z_1^a \wedge y_2 = z_2^a\}$. Such protocol can also be used to prove that the discrete logarithms of two group elements $y_1 \in G_1$ and $y_2 \in G_2$ to the bases $z_1 \in G_1$ and $z_2 \in G_2$, respectively in two different groups G_1 and G_2, are equal [12].
- Proof of knowledge of inequality. A proof of inequality of discrete logarithms of two group elements $y_1, y_2 \in G$ to the bases $z_1, z_2 \in G$, respectively, is denoted $PK\{(a) : y_1 = z_1^a \wedge y_2 \neq z_2^a\}$. Camenisch and Shoup gave an efficient protocol [21] for proving inequality of discrete logarithms, which only requires three multi-exponentiations for the prover and two multi-exponentiations for the verifier.

4 Our EPID Scheme

We begin with a high-level overview of our construction. In our scheme, each platform chooses a unique membership key f. As in anonymous credential schemes [17, 19], group signature schemes [8, 9], or DAA schemes [13, 14], the issuer in our scheme computes a signature on f. That is, the issuer computes a BBS+ signature (A, x, y). The value f is the platform's unique membership key and the signature (A, x, y) is the platform's membership credential, they together form the private key of the platform. To sign a signature, the platform proves in zero-knowledge that he has a BBS+ signature on his membership key f. To verify a group signature, the verifier verifies the zero-knowledge proof.

In addition, each platform chooses a random base B and computes $K := B^f$. This (B, K) pair serves the purpose of revocation check. To sign a group signature, the platform needs not only to show that he has a BBS+ signature on f, but also to prove that he constructs (B, K) pair correctly. That is, the platform proves in zero-knowledge

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

If a private key (A, x, y, f) was compromised, the revocation manager places f in `Priv-RL`. Given a signature which contains (B, K), one can easily tell whether the signature was created using the private key (A, x, y, f) by checking whether $K = B^f$. To revoke a signature, the revocation manager appends the (B, K) pair of the signature to `Sig-RL`. To sign a group signature, a non-revoked platform needs to prove that he did not generate the (B, K) pair before. This can be done by a zero-knowledge proof protocol for proving inequality of discrete logarithms.

Our Construction of EPID

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 cyclic group G_3 of order p in which the decisional Diffie-Hellman 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

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

Let $e : G_1 \times G_2 \to G_T$ be a bilinear map function and $H : \{0,1\}^* \to \mathbb{Z}_p$ be a collision resistant hash function. We treat H as a random oracle in the proof of security.

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

1. $\mathcal{P}$ chooses at random $f \leftarrow \mathbb{Z}_p$ and $y' \leftarrow \mathbb{Z}_p$, and computes $T := h_1^f \cdot h_2^{y'}$.
2. $\mathcal{P}$ sends T to $\mathcal{I}$, and performs the following proof of knowledge to $\mathcal{I}$

$$PK\{(f, y') \ : \ h_1^f \cdot h_2^{y'} = T\}$$

   Note that we can use the standard zero-knowledge proof protocols such as [24, 40]. Since the security proof requires rewinding to extract f from an adversarial platform, this step can only be run sequentially. With some loss of efficiency, we could modify this step to support concurrent execution by using verifiable encryption [21] of the f value.
3. $\mathcal{I}$ chooses at random $x \leftarrow \mathbb{Z}_p$ and $y'' \leftarrow \mathbb{Z}_p$, and computes

$$A := (g_1 \cdot T \cdot h_2^{y''})^{1/(x+\gamma)}.$$

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 BBS+ signature on f.

Sign : On input of `gpk`, $\texttt{sk} = (A, x, y, f)$, a message $m \in \{0,1\}^*$, and a signature based revocation list `Sig-RL`, 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 in $\mathbb{Z}_p$ that $b := y + ax$, and computes $T := A \cdot h_2^a$.
3. It runs the following signature of knowledge protocol

$$SPK\{(x, f, a, b) \ : \ B^f = K \wedge \\ e(T, g_2)^{-x} \cdot e(h_1, g_2)^f \cdot e(h_2, g_2)^b \cdot e(h_2, w)^a = e(T, w)/e(g_1, g_2)\}(m).$$

   (a) It randomly picks

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

   (b) It computes

$$R_1 := B^{r_f}, \qquad R_2 := e(T, g_2)^{-r_x} \cdot e(h_1, g_2)^{r_f} \cdot e(h_2, g_2)^{r_b} \cdot e(h_2, w)^{r_a}.$$

   (c) It then computes

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

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

$$s_x := r_x + cx, \qquad s_f := r_f + cf, \qquad s_a := r_a + ca, \qquad s_b := r_b + cb.$$

4. It sets $\sigma_0 := (B, K, T, c, s_x, s_f, s_a, s_b)$.
5. Let `Sig-RL` $= \{(B_1, K_1), \ldots, (B_{n_2}, K_{n_2})\}$. For $i = 1, \ldots, n_2$, it proves in zero-knowledge that `sk` has not been revoked in `Sig-RL`, i.e., computes

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

   We can use the zero-knowledge proof protocol from Camenisch and Shoup [21].
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 `gpk`, a message m, a private-key based revocation list `Priv-RL`, a signature based revocation list `Sig-RL`, 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

$$\hat{R}_1 := B^{s_f} \cdot K^{-c}, \\ \hat{R}_2 := e(T, g_2)^{-s_x} \cdot e(h_1, g_2)^{s_f} \cdot e(h_2, g_2)^{s_b} \cdot e(h_2, w)^{s_a} \cdot (e(g_1, g_2)/e(T, w))^c.$$

4. It verifies that

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

5. Let `Priv-RL` $= \{f_1, \ldots, f_{n_1}\}$. For $i = 1, \ldots, n_1$, it checks that $K \overset{?}{\neq} B^{f_i}$.

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

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

7. If all the above verification succeeds, it outputs `valid`, otherwise, outputs `invalid`.

Revoke : Initially, both revocation lists are empty, i.e., `Priv-RL` $:= \emptyset$ and `Sig-RL` $:= \emptyset$.

1. Private-key based revocation: Given `gpk`, `Priv-RL`, and a private key `sk` $= (A, x, y, f)$ to be revoked, $\mathcal{R}$ updates `Priv-RL` as follows: $\mathcal{R}$ verifies the correctness of `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 `Priv-RL`.
2. Signature based revocation: Given `gpk`, `Priv-RL`, `Sig-RL`, a message m, and corresponding signature σ, $\mathcal{R}$ updates `Sig-RL` 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 $\mathsf{Verify}(\texttt{gpk}, m, \texttt{Priv-RL}, \emptyset, \sigma_0) = \texttt{valid}$, then appends (B, K) in σ_0 to `Sig-RL`.

Before we present the security proofs of our EPID scheme in Section 6, we first describe some intuitions of our construction. We show in following two lemmas that steps 2-4 of the join algorithm and steps 2-4 of the verify algorithm indeed form a signature of knowledge

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

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:

$$B^f = K, \qquad e(T, g_2)^{-x} \cdot e(h_1, g_2)^f \cdot e(h_2, g_2)^b \cdot e(h_2, w)^a = e(T, w)/e(g_1, g_2).$$

The first equation holds trivially. The second equation holds because

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

Lemma 2 *The transcript of $(T, c, s_x, s_f, s_a, s_b)$ in σ_0 can be simulated.*

Proof. Observe that T is a commitment of A, given the random choice of a in $\mathbb{Z}_p$, T is distributed uniformly at random in G_1. The simulator can simulate T by choosing $T \leftarrow G_1$. Observe that c is the result of the hash function H. As we model H as a random oracle, c is uniformly distributed over $\mathbb{Z}_p$. It is also easy to see that s_x, s_f, s_a, s_b are uniformly distributed over $\mathbb{Z}_p$ given the random choice of r_x, r_f, r_a, r_b. The simulator chooses a random $c \leftarrow \mathbb{Z}_p$ and then chooses $s_x, s_f, s_a, s_b \leftarrow \mathbb{Z}_p$. The simulator outputs the transcript $(T, c, s_x, s_f, s_a, s_b)$. As we argued above, this transcript is distributed identically to the transcript in σ_0. □

Lemma 3 *There exists a knowledge extractor that extracts an (A, x, y, f) tuple from a prover, who runs steps 2-4 of the sign algorithm, such that $e(A, g_2^x w) = e(g_1 h_1^f h_2^y, g_2)$ and $K = B^f$.*

Proof. Suppose that a knowledge extractor can rewind the prover and control the outcome of H. The prover first computes T, then chooses r_x, r_f, r_a, r_b and computes R_1, R_2. For a hash value c, the prover outputs s_x, s_f, s_a, and s_b. For another hash value c', the prover outputs s'_x, s'_f, s'_a, and s'_b. If the prover is convincing, steps 2-4 of the verify algorithm would pass successfully. In other words, for two set of values, $\hat{R}'_1 = \hat{R}_1$ and $\hat{R}'_2 = \hat{R}_2$.

Let $\Delta c = c - c'$, $\Delta s_x = s_x - s'_x$, and similarly for $\Delta s_f, \Delta s_a$, and Δs_b. Let

$$\hat{x} := \Delta s_x/\Delta c, \qquad \hat{f} := \Delta s_f/\Delta c, \qquad \hat{a} := \Delta s_a/\Delta c, \qquad \hat{b} := \Delta s_b/\Delta c.$$

We can verify that $\hat{x}, \hat{f}, \hat{a}, \hat{b}$ satisfy the following two equations:

$$B^{\hat{f}} = K, \qquad e(T, g_2)^{-\hat{x}} \cdot e(h_1, g_2)^{\hat{f}} \cdot e(h_2, g_2)^{\hat{b}} \cdot e(h_2, w)^{\hat{a}} = e(T, w)/e(g_1, g_2).$$

Let $\hat{y} := \hat{b} - \hat{a}\hat{x}$, we can derive from the last equation that

$$\begin{aligned} e(T, g_2^{\hat{x}} w) &= e(T, w) \cdot e(T, g_2)^{\hat{x}} = e(g_1, g_2) \cdot e(h_1, g_2)^{\hat{f}} \cdot e(h_2, g_2)^{\hat{b}} \cdot e(h_2, w)^{\hat{a}} \\ &= e(g_1, g_2) \cdot e(h_1, g_2)^{\hat{f}} \cdot e(h_2, g_2)^{\hat{y} + \hat{a}\hat{x}} \cdot e(h_2, w)^{\hat{a}} \\ &= e(g_1 h_1^{\hat{f}} h_2^{\hat{y}}, g_2) \cdot e(h_2, g_2)^{\hat{a}\hat{x}} \cdot e(h_2, w)^{\hat{a}} = e(g_1 h_1^{\hat{f}} h_2^{\hat{y}}, g_2) \cdot e(h_2^{\hat{a}}, g_2^{\hat{x}} w) \end{aligned}$$

Thus we have

$$e(T \cdot h_2^{-\hat{a}}, g_2^{\hat{x}} w) = e(g_1 h_1^{\hat{f}} h_2^{\hat{y}}, g_2).$$

Let $\hat{A} := T \cdot h_2^{-\hat{a}}$, the extractor obtains an $(\hat{A}, \hat{x}, \hat{y}, \hat{f})$ tuple such that

$$e(\hat{A}, g_2^{\hat{x}} w) = e(g_1 h_1^{\hat{f}} h_2^{\hat{y}}, g_2) \qquad K = B^{\hat{f}}.$$

□

5 Implementation of the EPID Scheme

In this section, we first describe how to implement the proposed EPID scheme, i.e., we briefly review the construction of an admissible bilinear map from the Tate pairing and then 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 [15] and other related schemes.

5.1 Admissible Bilinear Map from the Tate Pairing

We first review the description of an admissible bilinear map [5, 31, 36] from the Tate pairing. Let $E(\mathbb{F}_q)$ be an elliptic curve over a prime field $\mathbb{F}_q$. Let G_1 be a cyclic subgroup of $E(\mathbb{F}_q)$ of prime order p, generated by a point g_1. We strict ourself to the curves such that p and q are prime integers. Let k be smallest positive integer such that $p \mid q^k - 1$. We call k the embedding degree. A Tate pairing takes the concrete form:

$$e : E(\mathbb{F}_q) \times E(\mathbb{F}_{q^k}) \;\rightarrow\; \mathbb{F}^*_{q^k}$$

Let G_2 be a cyclic subgroup of $E(\mathbb{F}_{q^k})$ of order p, generated by a point g_2 such that g_2 is linearly independent of g_1. Let G_T be the p-th roots of unity in $\mathbb{F}_{q^k}$. We now have an admissible bilinear map function $e : G_1 \times G_2 \rightarrow G_T$ from the Tate pairing. Note that we can optimize the efficiency of the bilinear map function using twist curves [5].

5.2 Choices of Elliptic Curves and Security Parameters

Before we suggest the choices of elliptic curves and security parameters, we first review two known attacks on discrete log in G_1: generic discrete log algorithms such as Pollard's Rho method [38] and the MOV attack [37] which reduces the discrete log problem in G_1 to a discrete log problem in $\mathbb{F}^*_{q^k}$. Therefore, we need to make sure p is sufficiently large and at the same time k is large enough to make discrete log in $\mathbb{F}^*_{q^k}$ intractable. 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. [39] for Tate pairing. As in [8], 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 [34], 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 for finding suitable elliptic curves (see [36] chapter 4.14). 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.

5.3 Efficiency of the EPID Scheme

We now summarize the efficiency of the EPID scheme as follow. For simplicity, we assume that `Priv-RL` and `Sig-RL` are empty.

- 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 $e(A, g_2)$, $e(h_1, g_2)$, $e(h_2, g_2)$, and $e(h_2, w)$. When computing a signature, the platform can compute R_2 as follows

$$R_2 := e(A, g_2)^{-r_x} \cdot e(h_1, g_2)^{r_f} \cdot e(h_2, g_2)^{r_b - ar_x} \cdot e(h_2, w)^{r_a}.$$

  Therefore, the sign algorithm only takes 1 exponentiations in G_1, 2 exponentiations in G_3, 1 multi-exponentiation in G_T, and no pairing needed. The platform can, per signature base, pre-compute $(B, K, a, b, T, r_x, r_f, r_a, r_b, R_1, R_2)$. Once the platform knows what message m to sign, it can start from step 3(c) of the sign algorithm. It only needs to perform one hash and four multiplications given per signature based pre-computation.

- To verify a signature, the verifier can pre-compute $e(g_1, g_2)$, $e(h_1, g_2)$, (h_2, g_2), and $e(h_2, w)$. When verifying a signature, the verifier can compute $\hat{R}_2$ as follows

$$\hat{R}_2 := e(T, g_2^{-s_x} w^{-c}) \cdot e(h_1, g_2)^{s_f} \cdot e(h_2, g_2)^{s_b} \cdot e(h_2, w)^{s_a} \cdot e(g_1, g_2)^{c}.$$

  Hence, the verify algorithm takes 1 multi-exponentiation in G_2, 1 multi-exponentiation in G_3, 1 multi-exponentiation in G_T, and 1 pairing operation.

Now we discuss the efficiency of revocation. For each item in `Priv-RL`, the platform needs to do nothing while the verifier needs to perform one exponentiation in G_3. For each item in `Sig-RL`, the platform needs to compute 3 multi-exponentiations in G_3 and the verifier needs to perform 2 multi-exponentiations in G_3.

5.4 Comparison with EPID, DAA, and BLAC Schemes

Brickell and Li developed an EPID scheme from the strong RSA assumption [15]. Their scheme is derived from the original DAA scheme in [13]. Same as the DAA scheme, using 2048-bit RSA modulus, the length of private key is 670 bytes and the size of a signature is 2800 bytes in [15]. The sizes of private keys and signatures in our EPID construction are only 129 bytes and 257 bytes, respectively, for 128-bit security level (higher security strength than 2048-bit RSA modulus). Thus our EPID scheme offers significant advantage over the scheme in [15], as hardware devices typically have limited size of secure storage.

Brickell, Chen, and Li [14] developed the first pairing-base DAA scheme. Their scheme is based on Camenisch and Lysyanskaya's pairing based group signature scheme [20]. Chen, Morrissey, and Smart further optimized the DAA scheme [27] 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 [14, 27]. In other words, an EPID scheme derived from the DAA schemes [14, 27] would not be as efficient as the EPID scheme described in this paper.

Tsang et al. recently proposed a Blacklistable Anonymous Credentials (BLAC) scheme in which a misbehaved user can be revoked based on his previous signatures [47]. The BLAC scheme [47] shares a similarity of construction as our EPID scheme. However, our EPID scheme is at least 30% more efficient than [47] in both signature creation and verification and size of the signatures, due to our optimizations in the zero-knowledge proof [2].

	private key size	signature size	sign	verify
Our EPID scheme	86 bytes	171 bytes	4 EXP	3 EXP + 1 P
BCL DAA scheme [14]	213 bytes	512 bytes	10 EXP	2 EXP + 5 P
CMS DAA scheme [27]	86 bytes	148 bytes	8 EXP + 1 P	1 EXP + 5 P
BLAC scheme [47]	86 bytes	320 bytes	10 EXP	7 EXP + 2 P

Table 1: A comparison between our EPID scheme, the pairing based DAA schemes [14, 27], and the BLAC scheme [47] with 80-bit security level, where EXP denotes multi-exponentiation and P denotes a pairing operation, assuming the revocation list is empty.

6 Security Proofs

We now show that the EPID scheme in Section 4 is secure under the security definition in Section 2, namely, the EPID scheme is correct, anonymous, and unforgeable. As we have shown in Section 4, steps 2-4 of the join algorithm and steps 2-4 of the verify algorithm form a signature of knowledge

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

[2]The comparison is based on the basic EPID scheme and basic BLAC scheme without revocation, as two schemes have different types of revocation methods. The revocation method used in the BLAC scheme is efficient than the signature based revocation in EPID, but less efficient than the private key based revocation.

In the following security proofs, we assume that the protocols in step 2 of the join protocol and in step 5 of the sign algorithm are indeed zero-knowledge proof of knowledge protocols, i.e., the transcripts of two protocols can be simulated and there exists an extractor that extracts (f, y') such that $h_1^f \cdot h_2^{y'} = T$ and extracts f such that $K = B^f$ and $K_i \neq B_i^f$ from a convincing platform.

Theorem 4 *The EPID scheme is correct.*

Proof. For a given group public key $\texttt{gpk} = (p, G_1, G_2, G_3, g_1, g_2, g_3, h_1, h_2, w)$, each private key takes the format of (A, x, y, f) where (A, x, y) is a BBS+ signature on f. The private key (A, x, y, f) satisfies the following equation $e(A, g_2^x w) = e(g_1 h_1^f h_2^y, g_2)$.

If an honest platform with a legitimate private key (A, x, y, f) follows the sign algorithm correctly, then steps 2-4 of the verify algorithm would succeed as (A, x, y, f) satisfies both $e(A, g_2^x w) = e(g_1 h_1^f h_2^y, g_2)$ and $K = B^f$. Let $\texttt{Priv-RL} = \{f_1, \ldots, f_{n_1}\}$ and $\texttt{Sig-RL} = \{(B_1, K_1), \ldots, (B_{n_2}, K_{n_2})\}$. If the platform has not been revoked in $\texttt{Priv-RL}$, i.e., $f \notin \{f_1, \ldots, f_{n_1}\}$, then step 4 of the verify algorithm would succeed, as for $i = 1, \ldots, n_1$, $K = B^f \neq B^{f_i}$. If the platform has not been revoked in $\texttt{Sig-RL}$, i.e., $i = 1, \ldots, n_2$, $f = \log_B K \neq \log_{B_i} K_i$, then the platform should be able to run the zero-knowledge proof protocol for proving inequality of discrete logarithms in step 5 of the sign algorithm. Observe that, if the platform has been revoked, then the verify algorithm would fail in either step 5 or 6. □

Before we prove the anonymity of the EPID scheme, we first give some intuition. Observe that the sign algorithm has three steps: first chooses B and computes K, then performs a zero-knowledge proof of knowledge of (A, x, y, f) such that $e(A, g_2^x w) = e(g_1 h_1^f h_2^y, g_2)$ and $K = B^f$, and finally proves that it is not revoked in $\texttt{Sig-RL}$ using Camenisch-Shoup's protocol for proving inequality of discrete logarithms. As the last two steps are zero-knowledge proof of knowledge protocols, the transcripts of last two steps can be simulated.

Note that (B, K) cannot be simulated. In fact, we intentionally design this so that the verifier can check (B, K) against $\texttt{Priv-RL}$, i.e., verify whether the (B, K) pair was created by a revoked membership key f. Loosely speaking, an EPID scheme is anonymous if no adversary can determine whether two signatures were created by one platform. Consider two pairs (B_1, K_1) and (B_2, K_2) from two signatures where $K_1 = B_1^{f_1}$ and $K_2 = B_2^{f_2}$, and B_1 and B_2 are chosen randomly. If the adversary can determine whether $f_1 = f_2$ (i.e., whether $\log_{B_1} K_1 = \log_{B_2} K_2$), the he breaks the DDH problem: given a (u, v, w, z) tuple, where $v = u^a$, $w = u^b$, $z = u^c$, the adversary can determine whether $\log_u v = \log_w z$, thus determine whether $c = ab$. The formal proof of anonymity is described as follows.

Theorem 5 *The EPID scheme is anonymous in the random oracle model under the decisional Diffie-Hellman assumption in G_3.*

Proof. Suppose algorithm $\mathcal{A}$ breaks the anonymity of the EPID scheme. We can build an algorithm $\mathcal{B}$ that breaks the decisional Diffie-Hellman assumption in G_3. Algorithm $\mathcal{B}$ is given as input a tuple $(u, v = u^a, w = u^b, z)$ where $u \leftarrow G_3$, $a, b, \leftarrow \mathbb{Z}_p$, and either $z = u^{ab}$ or z is a random element in G_3. Algorithm $\mathcal{B}$ decides which z was given by interacting with $\mathcal{A}$ as follows:

Setup. Let (G_1, G_2) be a bilinear group pair of prime order p with generator g_1 and g_2, respectively. Let G_3 be a cyclic group of prime order p with generator g_3. $\mathcal{B}$ does the following:

1. $\mathcal{B}$ receives a group public key $\texttt{gpk} = (p, G_1, G_2, G_3, g_1, g_2, g_3, h_1, h_2, w)$ from $\mathcal{A}$.

2. For simplicity, assume that the index of the platforms in the join query increases sequentially, i.e., $\mathcal{A}$ requests for creating $\mathcal{P}_1$, $\mathcal{P}_2$, and so on. Let N be the expected join queries $\mathcal{A}$ would make. $\mathcal{B}$ picks two random platform indices $i_0, i_1 \leftarrow \{1, \ldots, N\}$ and keeps i_0, i_1 secret.

Hash Queries. At any time, $\mathcal{A}$ can query the hash function H. $\mathcal{B}$ responds with random values in $\mathbb{Z}_q$ while ensuring consistency.

Queries. Algorithm $\mathcal{A}$ can perform join queries, sign queries, and corrupt queries. $\mathcal{B}$ responds as follows:

1. Join queries: $\mathcal{A}$ requests for creating a new platform $\mathcal{P}_i$. $\mathcal{B}$ runs the join protocol with $\mathcal{A}$ and obtains a private key $\mathtt{sk}_i = (A_i, x_i, y_i, f_i)$. If $i = i_0$ or i_1, $\mathcal{B}$ discards the obtained private key $\mathtt{sk}_i$. To give some intuition here, $\mathcal{B}$ behaves as if $\log_u v$ is f_{i_0} in the platform i_0's private key and $\log_w z$ is f_{i_1} in the platform i_1's private key. Observe that $\mathcal{B}$ knows neither $\log_u v$ nor $\log_w z$.
2. Sign queries: given a message $m \in \{0,1\}^*$, a signature based revocation list Sig-RL, and a platform index i, if $i \neq i_0, i_1$, then $\mathcal{B}$ use the private key $\mathtt{sk}_i = (A_i, x_i, y_i, f_i)$ to respond the query. For queries on platforms i_0 or i_1, $\mathcal{B}$ responds as follows:
   (a) For queries on platforms i_0, $\mathcal{B}$ checks whether Sig-RL contains any (B, K) pair from the signatures generated in previous sign queries for platform i_0, if so, $\mathcal{B}$ responds with $\perp$. Otherwise, $\mathcal{B}$ picks $r \leftarrow \mathbb{Z}_p^*$ and sets
   $$B := u^r, \qquad K := v^r.$$
   (b) Analogously, $\mathcal{B}$ checks whether platform i_1 has been revoked in Sig-RL, if so, $\mathcal{B}$ responds with $\perp$. Otherwise, $\mathcal{B}$ picks $r \leftarrow \mathbb{Z}_p^*$ and sets
   $$B := w^r, \qquad K := z^r.$$
   (c) If $\mathcal{B}$ did not respond $\perp$, $\mathcal{B}$ simulates the rest of σ_0 as follows: $\mathcal{B}$ picks $T \leftarrow G_1$ and $s_x, s_f, s_a, s_b \leftarrow \mathbb{Z}_p$. $\mathcal{B}$ computes R_1, R_2 using equations in step 3 of the verify algorithm, and then computes
   $$c := H(\mathtt{gpk}, B, K, T, R_1, R_2, m).$$
   In the rare event that $\mathcal{A}$ has already issued a hash query for $H(\mathtt{gpk}, B, K, T, R_1, R_2, m)$, $\mathcal{B}$ reports failure and aborts. Algorithm $\mathcal{B}$ sets
   $$\sigma_0 := (B, K, T, c, s_x, s_f, s_a, s_b).$$
   For each (B_i, K_i) pair in Sig-RL, $\mathcal{B}$ simulates the transcripts of the zero-knowledge proof and produce σ_i. As $\mathcal{B}$ controls the hash oracle, it can always create a well-formed σ_i. $\mathcal{B}$ sets the signature $\sigma := (\sigma_0, \sigma_1, \ldots, \sigma_{n_2})$. Note that by Lemma 2, σ is a properly distributed signature.
3. Corrupt queries: If a corrupt query is for a platform $i \neq i_0$ or i_1, then $\mathcal{B}$ responds with $\mathtt{sk}_i = (A_i, x_i, y_i, f_i)$. Otherwise, $\mathcal{B}$ reports failure and aborts.

Challenge. Algorithm $\mathcal{A}$ outputs a message m, a signature based revocation list Sig-RL, and two platforms i_0^* and i_1^*. If $\{i_0^*, i_1^*\} \neq \{i_0, i_1\}$, then $\mathcal{B}$ reports failure and aborts. Otherwise, let us assume $i_0^* = i_0$ and $i_1^* = i_1$. $\mathcal{B}$ picks $b \leftarrow \{0,1\}$ and generate a signature σ^* for m with platform i_b using the same method described in the sign queries above. $\mathcal{B}$ sends σ^* to $\mathcal{A}$.

Restricted Queries. Algorithm $\mathcal{A}$ issues restricted queries. Algorithm $\mathcal{B}$ responds as in regular queries above.

Output. In the end, $\mathcal{A}$ outputs $b' \in \{0, 1\}$ as the guess for b. If $b = b'$, then $\mathcal{B}$ outputs 0, which indicates that z is a random element in G_3. Otherwise $\mathcal{B}$ outputs 1, which means that $z = u^{ab}$.

Let ϵ be the probability that $\mathcal{A}$ succeeds in breaking the anonymity game. Suppose $\mathcal{B}$ does not abort during the above simulation. If z is random in G_3, $\mathcal{B}$ emulates the anonymity game perfectly, i.e., $\Pr[b = b'] > \frac{1}{2} + \epsilon$. If $z = u^{ab}$, then $\log_u v = \log_w z$, the private keys for platforms i_0 and i_1 are identical and thus the signature σ^* is independent of b. It follows that $\Pr[b = b'] = \frac{1}{2}$. Therefore, assuming $\mathcal{B}$ does not abort, it has advantage at least $\epsilon/2$ in solving the DDH problem in G_3.

We now discuss the probability that algorithm $\mathcal{B}$ does not abort in the above game. $\mathcal{B}$ does not abort if it can correctly guesses the values i_0^* and i_1^* in the setup phase and none of the sign queries cause $\mathcal{B}$ to abort. The probability that $\mathcal{B}$ can guess correctly is about $1/n^2$, where n is the total number of private keys $\mathcal{B}$ has created. The probability that $\mathcal{B}$ aborts in the sign queries is negligible if p is large enough. Therefore, the probability that $\mathcal{B}$ does not abort is roughly $1/n^2$. $\square$

We now give some intuition for the proof of the unforgeability game. In the unforgeability game, the adversary can query the challenger for joining the group, signing a message, or corrupting a platform's private key. In the end, the adversary outputs a group signature. The adversary wins if the signature he outputs is valid. As the sign algorithm is indeed a proof of knowledge protocol, we can rewind the proof and extract the knowledge (i.e., a private key) from the signature. Recall that a private key is a BBS+ signature on a membership key f. If an adversary can forge a group signature in the unforgeability game, we can build another adversary that forges a BBS+ signature.

Theorem 6 *The EPID scheme is unforgeable in the random oracle model under the strong Diffie-Hellman assumption in* (G_1, G_2).

Proof. Suppose algorithm $\mathcal{A}$ breaks the unforgeability of the EPID scheme, we build an algorithm $\mathcal{B}$ that breaks the unforgeability of the BBS+ signature scheme described in Section 3.3. Algorithm $\mathcal{B}$ interacts with $\mathcal{A}$ as follows:

Setup. Let (G_1, G_2) be a bilinear group pair of prime order p with generator g_1 and g_2, respectively. $\mathcal{B}$ is given a public key (g_1, g_2, h_1, h_2, w) of the BBS+ signature scheme, where $h_1, h_2 \in G_1$ and $w \in G_2$. $\mathcal{B}$ does the following:

1. $\mathcal{B}$ chooses a cyclic group G_3 of order p with generator g_3 and gives $\mathcal{A}$ the group public key $\texttt{gpk} = (p, G_1, G_2, G_3, g_1, g_2, g_3, h_1, h_2, w)$.
2. $\mathcal{B}$ maintains a list U of corrupted membership keys. Initially, $U := \emptyset$.

Hash Queries. At any time, $\mathcal{A}$ can query the hash function H. $\mathcal{B}$ responds with random values in $\mathbb{Z}_q$ while ensuring consistency.

Join Queries. $\mathcal{A}$ can make a join query on platform $\mathcal{P}_i$, it could be one of the following two cases:

1. $\mathcal{B}$ and $\mathcal{A}$ run the join protocol as follows: $\mathcal{A}$ runs steps 1-2 of the join protocol. As step 2 of the join protocol is a proof of knowledge protocol, $\mathcal{B}$ obtains (f_i, y_i') by rewinds $\mathcal{A}$. $\mathcal{B}$ now queries the BBS+ signature signing oracle on f_i and obtains (A_i, x_i, y_i). $\mathcal{B}$ computes $y_i'' := y_i - y_i'$ and returns (A_i, x_i, y_i'') to $\mathcal{A}$. $\mathcal{B}$ also appends f_i to U.

2. $\mathcal{B}$ generates $\mathtt{sk}_i$ as follows: $\mathcal{B}$ chooses a random $f_i \leftarrow \mathbb{Z}_p$. Next, $\mathcal{B}$ sets $(A_i, x_i, y_i) := \star$ which indicates that $\mathcal{B}$ does not know the corresponding BBS+ signature on f_i. Finally, $\mathcal{B}$ sets $\mathtt{sk}_i = (A_i, x_i, y_i, f_i)$. To give some intuition here, $\mathcal{B}$ will obtain a BBS+ signature on f_i only when $\mathcal{A}$ issues a corrupt query on the platform of index i.

Sign Queries. $\mathcal{A}$ asks for a signature on message m with `Sig-RL` by the private key at index i. If $(A_i, x_i, y_i) \neq \star$, $\mathcal{B}$ follows the signing algorithm with the private key (A_i, x_i, y_i, f_i) to obtain a group signature σ on m, and returns σ to $\mathcal{A}$.

In the case where $(A_i, x_i, y_i) = \star$, i.e., $\mathcal{B}$ does not have a valid private key for platform i but has f_i, $\mathcal{B}$ can simulate a signature on m as follows: $\mathcal{B}$ runs the first step of the sign algorithm and obtains a (B, K) pair. $\mathcal{B}$ then simulates the rest of σ_0 by picking $T \leftarrow G_1$ and $s_x, s_f, s_a, s_b \leftarrow \mathbb{Z}_p$. $\mathcal{B}$ computes R_1, R_2 using equations in step 3 of the verify algorithm, and then patches the hash oracle such that

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

If this causes a collision in the hash oracle, $\mathcal{B}$ reports failure and exits. Otherwise, algorithm $\mathcal{B}$ sets

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

For each (B_i, K_i) pair in `Sig-RL`, $\mathcal{B}$ simulates the transcripts of the zero-knowledge proof and produce σ_i. $\mathcal{B}$ sets the signature $\sigma := (\sigma_0, \sigma_1, \ldots, \sigma_{n_2})$ and returns σ to $\mathcal{A}$.

Corrupt Queries. $\mathcal{A}$ asks for the private key of the platform at some index i. If $(A_i, x_i, y_i) = \star$, $\mathcal{B}$ queries the BBS+ signature signing oracle with f_i and obtains a BBS+ signature (A_i, x_i, y_i) on f_i. $\mathcal{B}$ returns (A_i, x_i, y_i, f_i) to $\mathcal{A}$ and appends f_i to U.

Response. Finally, if algorithm $\mathcal{A}$ is successful, it outputs a forged signature σ^* on a message m^* along with a private-key based revocation list `Priv-RL`* and a signature based revocation list `Sig-RL`*, such that (1) $\mathsf{Verify}(\mathtt{gpk}, \mathtt{Priv\text{-}RL}^*, \mathtt{Sig\text{-}RL}^*, \sigma^*, m^*) = \mathtt{valid}$, and (2) for each $f_i \in U$, either $f_i \in \mathtt{Priv\text{-}RL}^*$ or there exists a (B_i, K_i) pair in `Sig-RL`* such that $K_i = B_i^{f_i}$. The first requirement states that the forged signature σ^* is a valid signature. The second requirement means that each corrupted platform private key has been revoked in either `Priv-RL`* or `Sig-RL`*.

Let $\mathtt{Priv\text{-}RL}^* = \{f_1, \ldots, f_{n_1}\}$ and $\mathtt{Sig\text{-}RL}^* = \{(B_1, K_1), \ldots, (B_{n_2}, K_{n_2})\}$. Observe that the sign algorithm is essentially a proof of knowledge protocol

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

and combined with the following series of proof of knowledge protocols, for $i = 1, \ldots, n_2$,

$$PK\{(f) \ : \ K = B^f \ \wedge \ K_i \neq B_i^f\}$$

Using the knowledge extractor of Lemma 3, $\mathcal{B}$ obtains an (A^*, x^*, y^*, f^*) tuple from σ^* such that

$$e(A^*, g_2^{x^*} w) = e(g_1 h_1^{f^*} h_2^{y^*}, g_2), \qquad\qquad K = B^{f^*}. \tag{1}$$

As G_3 is a cyclic group of prime order p, for the (B, K) pair in σ^*, there is only one $f^* \in \mathbb{Z}_p$ such that $K = B^{f^*}$. Using the knowledge extractor of the zero-knowledge proof of inequality of discrete logs, $\mathcal{B}$ extracts the same f^* such that $K = B^{f^*}$ and $K_i \neq B_i^{f^*}$, for $i = 1, \ldots, n_2$. We next show that $f^* \notin U$. For every $f_i \in U$, either $f_i \in \texttt{Priv-RL}^*$ or there exist a (B_i, K_i) pair in $\texttt{Sig-RL}^*$ such that $K_i = B_i^{f_i}$. If $f^* = f_i \in \texttt{Sig-RL}^*$ then the verification of σ^* would fail. If $f^* = f_i$ and $K_i = B_i^{f_i}$, this would contradict to the result of the knowledge extractor. Thus, $f \neq f_i$ for any $f_i \in U$.

In the end of this game, $\mathcal{B}$ outputs (A^*, x^*, y^*, f^*). Observe that, (A^*, x^*, y^*) is a valid BBS+ signature on f^* from equation (1). Also observe that $\mathcal{B}$ has queried the BBS+ signature oracle only for f_i in U. As $f^* \notin U$, $\mathcal{B}$ has successfully forged a BBS+ signature on f^* if $\mathcal{B}$ does not abort during the simulation. As the BBS+ signature scheme is unforgeable against chosen message attacks under the SDH assumption, our EPID scheme is also unforgeable under the SDH assumption. □

References

[1] Giuseppe Ateniese, Jan Camenisch, Marc Joye, and Gene 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.

[2] Giuseppe Ateniese, Dawn Xiaodong Song, and Gene 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] Man Ho Au, Willy Susilo, and Yi Mu. Constant-size dynamic k-TAA, booktitle = Proceedings of the 5rd International Conference on Security and Cryptography for Networks, pages = 111–125, publisher = Springer, series = LNCS, volume = 4116, year = 2006, note = `http://eprint.iacr.org/2008/136`,.

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

[5] Paulo S. L. M. Barreto, Hae Y. Kim, Ben Lynn, and Michael Scott. Efficient algorithms for pairing-based cryptosystems. In *Advances in Cryptology — CRYPTO '02*, volume 2442 of *LNCS*, pages 354–368. Springer, 2002.

[6] Dan Boneh. The decision diffie-hellman problem. In *Proceedings of the 3rd Algorithmic Number Theory Symposium*, volume 1423 of *LNCS*, pages 48–63. Springer, 1998.

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

[8] Dan Boneh, Xavier Boyen, and Hovav Shacham. Short group signatures. In *Advances in Cryptology — CRYPTO '04*, volume 3152 of *LNCS*, pages 41–55. Springer, 2004.

[9] Dan Boneh and Hovav Shacham. Group signatures with verifier-local revocation. In *Proceedings of 11th ACM Conference on Computer and Communications Security*, pages 168–177, October 2004.

[10] Stefan A. Brands. *Rethinking Public Key Infrastructures and Digital Certificates: Building in Privacy.* MIT Press, August 2000.

[11] Emmanuel Bresson and Jacques 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.

[12] Ernest F. Brickell, David Chaum, Ivan Damgård, and Jeroen van de Graaf. Gradual and verifiable release of a secret. In *Advances in Cryptology — CRYPTO '87*, volume 293 of *LNCS*, pages 156–166. Springer, 1987.

[13] Ernie Brickell, Jan Camenisch, and Liqun Chen. Direct anonymous attestation. In *Proceedings of the 11th ACM Conference on Computer and Communications Security*, pages 132–145. ACM Press, 2004.

[14] Ernie Brickell, Liqun Chen, and Jiangtao 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.

[15] Ernie Brickell and Jiangtao 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*. ACM Press, October 2007.

[16] Jan Camenisch and Jens 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] Jan Camenisch and Anna 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] Jan Camenisch and Anna 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] Jan Camenisch and Anna Lysyanskaya. A signature scheme with efficient protocols. In *Proceedings of the 3rd Conference on Security in Communication Networks*, volume 2576 of *LNCS*, pages 268–289. Springer, 2002.

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

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

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

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

[24] David Chaum, Jan-Hendrik Evertse, and Jeroen 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.

[25] David Chaum and Torben P. Pedersen. Wallet databases with observers. In *Advances in Cryptology — CRYPTO '92*, volume 740 of *LNCS*, pages 89–105. Springer, 1992.

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

[27] Liqun Chen, Paul Morrissey, and Nigel 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.

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

[29] Amos Fiat and Adi Shamir. How to prove yourself: Practical solutions to identification and signature problems. In *Advances in Cryptology — CRYPTO '86*, volume 263 of *LNCS*, pages 186–194. Springer, 1987.

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

[31] Steven D. Galbraith, Keith Harrison, and David Soldera. Implementing the tate pairing. In *Proceedings of the 5th International Symposium on Algorithmic Number Theory*, pages 324–337, London, UK, 2002. Springer.

[32] Antoine Joux and Kim Nguyen. Separating Decision Diffie-Hellman from Diffie-Hellman in cryptographic groups. *Journal of Cryptology*, 16(4):239–247, 2003.

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

[34] Neal Koblitz and Alfred 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.

[35] Adrian Leung and Chris 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.

[36] Ben Lynn. *On the implementation of pairing-based cryptosystems.* PhD thesis, Stanford University, Stanford, California, 2007.

[37] Alfred Menezes, Scott Vanstone, and Tatsuaki Okamoto. Reducing elliptic curve logarithms to logarithms in a finite field. In *Proceedings of the 23rd annual ACM Symposium on Theory of Computing (STOC)*, pages 80–89. ACM Press, 1991.

[38] Alfred J. Menezes, Paul C. Van Oorschot, and Scott A. Vanstone. *Handbook of Applied Cryptography (revised reprint with updates).* CRC Press, 1997.

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

[40] Torben 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.

[41] David Pointcheval and Jacques Stern. Security proofs for signature schemes. In *Advances in Cryptology — EUROCRYPT '96*, volume 1070 of *LNCS*, pages 387–398. Springer, 1996.

[42] Claus P. Schnorr. Efficient identification and signatures for smart cards. *Journal of Cryptology*, 4(3):161–174, 1991.

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

[44] Dawn Xiaodong 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.

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

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

[47] Patrick P. Tsang, Man Ho Au, Apu Kapadia, and Sean W. Smith. Blacklistable anonymous credentials: Blocking misbehaving users without TTPs. In *ACM Conference on Computer and Communications Security*, pages 72–81. ACM Press, 2007.