A Pairing-Based DAA Scheme Further Reducing TPM Resources

Ernie Brickell
Intel Corporation
ernie.brickell@intel.com

Jiangtao Li
Intel Labs
jiangtao.li@intel.com

Abstract

Direct Anonymous Attestation (DAA) is an anonymous signature scheme designed for anonymous attestation of a Trusted Platform Module (TPM) while preserving the privacy of the device owner. Since TPM has limited bandwidth and computational capability, one interesting feature of DAA is to split the signer role between two entities: a TPM and a host platform where the TPM is attached. Recently, Chen proposed a new DAA scheme that is more efficient than previous DAA schemes. In this paper, we construct a new DAA scheme requiring even fewer TPM resources. Our DAA scheme is about 5 times more efficient than Chen's scheme for the TPM implementation using the Barreto-Naehrig curves. In addition, our scheme requires much smaller size of software code that needs to be implemented in the TPM. This makes our DAA scheme ideal for the TPM implementation. Our DAA 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

The concept and a concrete scheme of Direct Anonymous Attestation (DAA) were first introduced by Brickell, Camenisch, and Chen [5] for remote anonymous authentication of a Trusted Platform Module (TPM). The DAA scheme was adopted by the Trusted Computing Group (TCG) [22], an industry standardization body that aims to develop and promote an open industry standard for trusted computing hardware and software building blocks. The DAA scheme was standardized in the TCG TPM Specification Version 1.2 [21] and has recently been adopted by ISO/IEC as an international standard.

A DAA scheme involves three types of entities: an issuer, signers, and verifiers. The issuer is in charge of verifying the legitimation of signers and of issuing a membership credential to each signer. A signer can prove membership anonymously to a verifier by creating a DAA signature. The verifier can verify the membership of the signer from the DAA signature but he cannot learn the identity of the signer. DAA scheme can be seen as a special group signature scheme without the open feature, i.e., a DAA signature cannot be opened by anyone including the issuer to find out the identity of the signer.

One interesting feature of DAA is that the signer role of DAA is split between two entities: a TPM and a host where the TPM is attached. The TPM is the main signer but has limited bandwidth, computational capability, and storage. The host is a helper with more computational power but is less trusted. The TPM is the real signer and has the private signing key. The host helps the TPM to compute DAA signatures, but is not allowed to learn the private signing key or forge a DAA signature without the involvement from the TPM.

After DAA was first introduced, it has drawn a lot of attention from both industry and cryptographic community, e.g., in [10, 18, 1, 8, 6, 12, 11], to list a few. The original DAA scheme [5] is based on the strong RSA assumption. Recently several groups of researchers have constructed pairing-based DAA schemes to achieve better efficiency. The first pairing-based DAA scheme was proposed by Ernie, Chen, and Li [6, 7]. Chen, Morrissey, and Smart improved the BCL-DAA scheme using asymmetric pairing [12, 13]. These DAA schemes are based on the LRSW assumption [19].

Brickell and Li proposed an extension of DAA called Enhanced Privacy ID (EPID) [8] and presented a concrete EPID scheme based on the q-SDH assumption [9]. The EPID schemes focus on the revocation capabilities and treat the signer as a single entity instead of combination of a TPM and a host. Independently Chen and Feng proposed a DAA scheme [15] using q-SDH assumption. Recently, Chen builds a new DAA

scheme [11] on top of the EPID scheme [9] by reducing the size of the private signing key. As compared in [11], q-SDH based DAA schemes [15, 9, 11] are more efficient than LRSW-based DAA schemes [6, 7, 12, 13], especially in the efficiency of the signature verification algorithm.

To the best of our knowledge, Chen's DAA scheme [11] is the most efficient DAA scheme and it requires least amount of TPM resources[1]. In this paper, we give a simple improvement to Chen's DAA scheme. Our DAA scheme is about 5 times more efficient for the TPM implementation using the Barreto-Naehrig curves [2]. In addition, our scheme requires much smaller size of software code that needs to be implemented in the TPM. This makes our DAA scheme ideal for the TPM implementation. More specifically, let $e : G_1 \times G_2 \rightarrow G_T$ be a bilinear map function. The DAA scheme in [11] requires two exponentiations in G_1 and one exponentiation in G_T. Whereas our DAA scheme in this paper requires only three exponentiations in G_1. Our improvement seems to be small, but has significant impact to TPM for the following two reasons:

- Usually operations in G_1 are more efficient than the operations in G_T. According to the arguments in [13], exponentiation in G_1 is about 1/4 the cost of exponentiation in G_T for symmetric pairing. For highly efficient curve choices such as Barreto-Naehrig curves [2] with 128-bit security, G_1 is an elliptic curve group over $\mathbb{F}_q$ while G_T is a subgroup of $\mathbb{F}_{q^{12}}$. Exponentiation in G_1 is about 14 times more efficient than the one in G_T. Thus the computation needed for TPM in our scheme is about 5 times more efficient than the one in DAA scheme [11] using the Barreto-Naehrig curves.

- In our scheme TPM only requires to implement G_1 while the other DAA schemes [13, 11, 15] require TPM to implement both G_1 and G_T. For small hardware devices such as TPM, more software code means larger firmware image, larger flash storage needed, and more software validation required. Note that if TPM has already implemented EC-DSA or other ECC primitives for other purposes, the additional software code needed for implementing our DAA scheme is very minimum.

Rest of this paper is organized as follows. We first review the formal specification and security requirements of DAA in Section 2. We then review the definition of pairing and related security assumptions in Section 3. We present our DAA scheme in Section 4 and give the security proof in Section 5. We compare our scheme with the existing DAA schemes in Section 6 and conclude our paper in Section 7.

2 Review Security Model of DAA

In this section, we review the specification and security model of DAA proposed in [7]. The security model in [7] is simpler than the original DAA definition [5] and easier to understand the security properties of DAA. There are four types of players in a DAA scheme: an issuer $\mathcal{I}$, a TPM $\mathcal{M}_i$, a host $\mathcal{H}_i$ and a verifier $\mathcal{V}_j$. $\mathcal{M}_i$ and $\mathcal{H}_i$ form a platform in the trusted computing environment and share the role of a DAA signer. A DAA scheme has three polynomial-algorithms ($\mathsf{Setup}, \mathsf{Verify}, \mathsf{Link}$) and two interactive protocols ($\mathsf{Join}, \mathsf{Sign}$):

Setup : On input of a security parameter 1^k, $\mathcal{I}$ uses this randomized algorithm to produce a pair ($\mathtt{gpk}$, $\mathtt{isk}$), where $\mathtt{isk}$ is the issuer's secret key, and $\mathtt{gpk}$ is the public key including the global public parameters.

Join : This randomized algorithm consists of two sub-algorithms $\mathsf{Join_t}$ and $\mathsf{Join_i}$. $\mathcal{M}_i$ uses $\mathsf{Join_t}$ to produce a pair $(\mathtt{sk}_i, \mathtt{comm}_i)$, where $\mathtt{sk}_i$ is the TPM's secret key and $\mathtt{comm}_i$ is a commitment of $\mathtt{sk}_i$. On input of $\mathtt{comm}_i$ and $\mathtt{isk}$, $\mathcal{I}$ uses $\mathsf{Join_i}$ to produce $\mathtt{cre}_i$, which is a DAA credential associated with $\mathtt{sk}_i$. Note that the value $\mathtt{cre}_i$ is given to both $\mathcal{M}_i$ and $\mathcal{H}_i$, but the value $\mathtt{sk}_i$ is known to $\mathcal{M}_i$ only.

Sign : On input of $\mathtt{sk}_i$, $\mathtt{cre}_i$, a basename $\mathtt{bsn}_j$ (the name string of $\mathcal{V}_j$ or a special symbol $\perp$), and a message m that includes the data to be signed and the verifier's nonce n_V for freshness, $\mathcal{M}_i$ and $\mathcal{H}_i$ use this randomized algorithm to produce a signature σ on m under $(\mathtt{sk}_i, \mathtt{cre}_i)$ associated with $\mathtt{bsn}_j$. The basename $\mathtt{bsn}_j$ is used for controlling the linkability.

Verify : On input of m, $\mathtt{bsn}_j$, a candidate signature σ for m, and a set of revoked secret keys $\mathtt{RL}$, $\mathcal{V}_j$ uses this deterministic algorithm to return either 1 (accept) or 0 (reject). How to build the revocation list is out the scope of the DAA scheme.

[1] The original CMS-DAA scheme [12] requires lesser TPM resources. However, there was a security flaw in their DAA scheme. The patched version [13] has the same computational complexity for TPM as in Chen's DAA scheme [11].

Link : On input of two signatures σ_0 and σ_1, $\mathcal{V}_j$ uses this deterministic algorithm to return 1 (linked), 0 (unlinked) or $\perp$ (invalid signatures). Link will output $\perp$ if, by using an empty $\mathtt{RL}$, either $\mathsf{Verify}(\sigma_0) = 0$ or $\mathsf{Verify}(\sigma_1) = 0$ holds. Otherwise, Link will output 1 if signatures can be linked or 0 if the signatures cannot be linked.

A DAA scheme is secure if it is correct, user-controlled-anonymous, and user-controlled-traceable.

Correctness If both the signer and verifier are honest, that implies $\mathtt{sk}_i \notin \mathtt{RL}$, the signatures and their links generated by the signer will be accepted by the verifier with overwhelming probability. This means that the DAA scheme must meet the following consistency requirement.

$(\mathtt{gpk}, \mathtt{isk}) \leftarrow \mathsf{Setup}(1^k)$, $(\mathtt{sk}_i, \mathtt{cre}_i) \leftarrow \mathsf{Join}(\mathtt{isk}, \mathtt{gpk})$, $(m_b, \sigma_b) \leftarrow \mathsf{Sign}(m_b, \mathtt{bsn}_j, \mathtt{sk}_i, \mathtt{cre}_i, \mathtt{gpk})|_{b=\{0,1\}}$, $\Longrightarrow 1 \leftarrow \mathsf{Verify}(m_b, \mathtt{bsn}_j, \sigma_b, \mathtt{gpk}, \mathtt{RL})|_{b=\{0,1\}} \ \wedge \ 1 \leftarrow \mathsf{Link}(\sigma_0, \sigma_1, \mathtt{gpk})|_{\mathtt{bsn}_j \neq \perp}$.

User-Controlled-Anonymity A DAA scheme is user-controlled-anonymous if no probabilistic polynomial-time adversary can win the following game between a challenger $\mathcal{C}$ and an adversary $\mathcal{A}$ as follows:

- Initial: $\mathcal{C}$ runs $\mathsf{Setup}(1^k)$ and gives the resulting $\mathtt{isk}$ and $\mathtt{gpk}$ to $\mathcal{A}$.
- Phase 1: $\mathcal{C}$ is probed by $\mathcal{A}$ who makes the following queries:
  - Sign. $\mathcal{A}$ submits a signer's identity S, a basename $\mathtt{bsn}$ (either $\perp$ or a data string) and a message m of his choice to $\mathcal{C}$, who runs Sign to get a signature σ and responds with σ.
  - Join. $\mathcal{A}$ submits a signer's identity S of his choice to $\mathcal{C}$, who runs $\mathsf{Join}_{\mathsf{t}}$ with $\mathcal{A}$ to create $\mathtt{sk}$ and to obtain $\mathtt{cre}$ from $\mathcal{A}$. $\mathcal{C}$ verifies the validation of $\mathtt{cre}$ and keeps $\mathtt{sk}$ secret.
  - Corrupt. $\mathcal{A}$ submits a signer's identity S of his choice to $\mathcal{C}$, who responds with the value $\mathtt{sk}$ of the signer.
- Challenge: At the end of Phase 1, $\mathcal{A}$ chooses two signers' identities S_0 and S_1, a message m and a basename $\mathtt{bsn}$ of his choice to $\mathcal{C}$. $\mathcal{A}$ must not have made any Corrupt query on either S_0 or S_1, and not have made the Sign query with the same $\mathtt{bsn}$ if $\mathtt{bsn} \neq \perp$ with either S_0 or S_1. To make the challenge, $\mathcal{C}$ chooses a bit b uniformly at random, signs m associated with $\mathtt{bsn}$ under $(\mathtt{sk}_b, \mathtt{cre}_b)$ to get a signature σ and returns σ to $\mathcal{A}$.
- Phase 2: $\mathcal{A}$ continues to probe $\mathcal{C}$ with the same type of queries that it made in Phase 1. Again, it is not allowed to corrupt any signer with the identity either S_0 or S_1, and not allowed to make any Sign query with $\mathtt{bsn}$ if $\mathtt{bsn} \neq \perp$ with either S_0 or S_1.
- Response: $\mathcal{A}$ returns a bit b'. We say that the adversary wins the game if $b = b'$.

Definition 1 *Let $\mathcal{A}$ denote an adversary that plays the game above. We denote by $\mathbf{Adv}[\mathcal{A}^{anon}_{\mathcal{DAA}}] = |\mathbf{Pr}[b' = b] - 1/2|$ the advantage of $\mathcal{A}$ in breaking the user-controlled-anonymity game. We say that a DAA scheme is user-controlled-anonymous if for any probabilistic polynomial-time adversary $\mathcal{A}$, $\mathbf{Adv}[\mathcal{A}^{anon}_{\mathcal{DAA}}]$ is negligible.*

User-Controlled-Traceability A DAA scheme is user-controlled-traceable if no probabilistic polynomial-time adversary can win the following game between a challenger $\mathcal{C}$ and an adversary $\mathcal{A}$ as follows:

- Initial: $\mathcal{C}$ executes $\mathsf{Setup}(1^k)$ and gives the resulting $\mathtt{gpk}$ to $\mathcal{A}$. It keeps $\mathtt{isk}$ secret.
- Probing: $\mathcal{C}$ is probed by $\mathcal{A}$ who makes the following queries:
  - Sign. The same as in the game of user-controlled-anonymity.
  - Semi-sign. $\mathcal{A}$ submits a signer's identity S along with the data transmitted from $\mathcal{H}_i$ to $\mathcal{M}_i$ in Sign of his choice to $\mathcal{C}$, who acts as $\mathcal{M}_i$ in Sign and responds with the data transmitted from $\mathcal{M}_i$ to $\mathcal{H}_i$ in the Sign protocol.

– Join. There are two cases of this query. Case 1: $\mathcal{A}$ submits a signer's identity S of his choice to $\mathcal{C}$, who runs Join to create $\mathtt{sk}$ and $\mathtt{cre}$ for the signer. Case 2: $\mathcal{A}$ submits a signer's identity S with a $\mathtt{sk}$ value of his choice to $\mathcal{C}$, who runs $\mathsf{Join}_{\mathsf{i}}$ to create $\mathtt{cre}$ for the signer and puts the given $\mathtt{sk}$ into $\mathtt{RL}$. $\mathcal{C}$ responds the query with $\mathtt{cre}$. Suppose that $\mathcal{A}$ does not use a single S for both of the cases.

– Corrupt. This is the same as in the game of user-controlled-anonymity, except that at the end $\mathcal{C}$ puts the revealed $\mathtt{sk}$ into the list of $\mathtt{RL}$.

- Forge: $\mathcal{A}$ returns a signer's identity S, a signature σ, its signed message m and the associated basename $\mathtt{bsn}$. We say that the adversary wins the game if

  1. $\mathsf{Verify}(m, \mathtt{bsn}, \sigma, \mathtt{gpk}, \mathtt{RL}) = 1$ (accepted), but σ is neither a response of the existing Sign queries nor a response of the existing Semi-sign queries (partially); and/or
  2. In the case of $\mathtt{bsn} \neq \perp$, there exists another signature σ' associated with the same identity and $\mathtt{bsn}$, and the output of $\mathsf{Link}(\sigma, \sigma')$ is 0 (unlinked).

Definition 2 *Let $\mathcal{A}$ be an adversary that plays the game above. Let $\mathbf{Adv}[\mathcal{A}^{trace}_{\mathcal{DAA}}] = \mathbf{Pr}[\mathcal{A}\ wins]$ denote the advantage that $\mathcal{A}$ breaks the user-controlled-traceability game. We say that a DAA scheme is user-controlled-traceable if for any probabilistic polynomial-time adversary $\mathcal{A}$, $\mathbf{Adv}[\mathcal{A}^{trace}_{\mathcal{DAA}}]$ is negligible.*

3 Pairings and Complexity Assumptions

3.1 Background on Bilinear Maps

Our DAA scheme use bilinear maps as a fundamental building block. We follow the notation of Boneh, Boyen, and Shacham [4] to review some background on pairings. 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 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.

3.2 Strong Diffie-Hellman Assumption

The security of our DAA scheme is related to the hardness of the q-SDH problem introduced by Boneh and Boyen [3]. 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 [3] 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 [20]. The q-SDH assumption was later used in [4] for constructing a short group signature scheme. The security of the SDH problem was studied by Cheon [16].

3.3 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 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.

Let (G_1, G_2) be a bilinear group pair. Our DAA scheme requires the DDH problem for G_1 to be hard. The DDH assumption on G_1 is often known as the External Diffie-Hellman (XDH) assumption. This assumption is also used in Chen's DAA scheme [11].

4 The Proposed DAA Scheme

In this section, we present our construction of DAA scheme from bilinear maps. Our construction builds on top of the recent pairing-based EPID scheme [9] and Chen's DAA scheme [11]. The DAA scheme has three algorithms Setup, Verify, Link and two interactive protocols Join and Sign which are defined as follows.

4.1 Setup Algorithm

The setup algorithm is exactly the same as the one in [11]. On input of the security parameters 1^t, the setup algorithm takes the following steps:

1. Choose an asymmetric bilinear group pair (G_1, G_2) of prime order p and a pairing function $e : G_1 \times G_2 \rightarrow G_T$. Let g_1 and g_2 be the generators of G_1 and G_2, respectively.
2. Choose $h_1, h_2 \leftarrow G_1$, $\gamma \leftarrow \mathbb{Z}_p^*$, and compute $w := g_2^{\gamma}$.
3. Select five hash functions $H_1 : \{0,1\}^* \rightarrow \mathbb{Z}_p$, $H_2 : \{0,1\}^* \rightarrow \mathbb{Z}_p$, $H_3 : \{0,1\}^* \rightarrow G_1$, $H_4 : \{0,1\}^* \rightarrow \mathbb{Z}_p$, $H_5 : \{0,1\}^* \rightarrow \mathbb{Z}_p$.
4. Compute $T_1 = e(g_1, g_2)$, $T_2 = e(h_1, g_2)$, $T_3 = e(h_2, g_2)$, and $T_4 = e(h_2, w)$.
5. Output the DAA public key and the issuer's private key

$$(\mathtt{gpk}, \mathtt{isk}) := ((G_1, G_2, G_T, p, e, g_1, h_1, h_2, g_2, w, H_1, H_2, H_3, H_4, H_5, T_1, T_2, T_3, T_4), \gamma)$$

Note that T_1, T_2, T_3, and T_4 are optional in $\mathtt{gpk}$, as they can be computed from g_1, h_1, h_2, g_2, w by the signers and verifiers. Also note that, in the actual implementation, we can choose the same hash function for H_1, H_2, H_4, and H_5. We use different hash functions in order to prove the security.

4.2 Join Protocol

The join protocol is the same as in [11] as well. This protocol is performed by a TPM $\mathcal{M}$, the corresponding host $\mathcal{H}$, and an issuer $\mathcal{I}$. Assume $\mathcal{M}$ and $\mathcal{I}$ have already established a secure authenticated channel using $\mathcal{M}$'s endorsement key [21]. Let DAAseed be $\mathcal{M}$'s internal secret seed. Let K_I be $\mathcal{I}$'s long term public key. In the join protocol, $\mathcal{M}$ chooses a unique secret key $\mathtt{sk} = f$ and then obtains a credential $\mathtt{cre} = (A, x)$ from $\mathcal{I}$ such that $A = (g_1 \cdot h_1^f)^{1/(x+\gamma)}$. The join protocol takes the following steps.

1. $\mathcal{I}$ chooses a nonce $n_I \in \{0,1\}^t$ and sends n_I as a challenge to $\mathcal{M}$.
2. $\mathcal{M}$ computes $f := H_1(\mathsf{DAAseed} \| \mathsf{cnt} \| K_I)$, where cnt is a count value. $\mathcal{M}$ sets its secret key $\mathtt{sk} := f$. The purpose of using K_I and cnt can be found in the original DAA scheme [5].

3. $\mathcal{M}$ chooses at random $r_f \leftarrow \mathbb{Z}_p$ and computes $F := h_1^f$ and $R := h_1^{r_f}$.
4. $\mathcal{M}$ computes $c := H_2(\mathtt{gpk} \| n_I \| F \| R)$ and $s_f := r_f + c \cdot f \pmod p$.
5. $\mathcal{M}$ sets $\mathtt{comm} := (F, c, s_f, n_I)$ and sends $\mathtt{comm}$ to $\mathcal{I}$.
6. $\mathcal{I}$ verifies the value of n_I and checks F against the revocation list.
7. $\mathcal{I}$ computes $\hat{R} := h_1^{s_f} \cdot F^{-c}$ and verifies that $c = H_2(\mathtt{gpk} \| n_I \| F \| \hat{R})$. If verification fails, then **abort**.
8. $\mathcal{I}$ chooses at random $x \leftarrow \mathbb{Z}_p$ and computes $A := (g_1 \cdot F)^{1/(x+\gamma)}$.
9. $\mathcal{I}$ sets the DAA credential $\mathtt{cre} := (A, x)$ and sends $\mathtt{cre}$ to $\mathcal{M}$.
10. $\mathcal{M}$ forwards F and $\mathtt{cre}$ to $\mathcal{H}$.
11. $\mathcal{H}$ verifies that $e(A, w g_2^x) = e(g_1 F, g_2)$. If verification fails, then **abort**.

Note that the TPM $\mathcal{M}$ and the host $\mathcal{H}$ have a DAA signing key (A, x, f) such that $e(A, w g_2^x) = e(g_1 h_1^f, g_2)$. In the DAA schemes [9, 15], the signing key is (A, x, y, f) such that $e(A, w g_2^x) = e(g_1 h_1^f h_2^y, g_2)$. Therefore our scheme has a smaller signing key.

4.3 Sign Protocol

This join protocol is performed by a TPM $\mathcal{M}$ and a host $\mathcal{H}$, where $\mathcal{M}$ has the secret key f and $\mathcal{H}$ has the credential (A, x). The other input of the protocol is the DAA public key $\mathtt{gpk}$, a message m to be signed, and a basename $\mathtt{bsn}$ and a nonce n_V from the verifier. In this protocol, the signer chooses $B \in G_1$ and computes $K := B^f$, then uses zero-knowledge proof to prove

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

As in most of DAA schemes, the (B, K) pair is used for revocation check. To prove $e(A, w g_2^x) = e(g_1 h_1^f, g_2)$ holds, the signer first computes $T = A \cdot h_2^a$ where a is randomly chosen, then proves the following equation

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

The overall approach here is the same as in [11]. The main difference between our scheme and Chen's DAA scheme [11] is on how we divide the computation between $\mathcal{M}$ and $\mathcal{H}$ in a secure way. The join protocol takes the following steps:

1. If $\mathtt{bsn} = \perp$, $\mathcal{M}$ chooses $B \leftarrow G_1$, otherwise, $\mathcal{M}$ computes $B := H_3(\mathtt{bsn})$.
2. $\mathcal{M}$ chooses at random $r_f \leftarrow \mathbb{Z}_p$ and computes

$$K := B^f, \qquad R_1 := B^{r_f}, \qquad R_{2t} := h_1^{r_f}.$$

3. $\mathcal{M}$ sends (B, K, R_1, R_{2t}) to $\mathcal{H}$.
4. $\mathcal{H}$ chooses $a \leftarrow \mathbb{Z}_p$, computes $b := a \cdot x \pmod p$, and $T := A \cdot h_2^a$.
5. $\mathcal{H}$ randomly picks

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

6. $\mathcal{H}$ computes

$$\begin{aligned} R_2 &:= e(T, g_2)^{-r_x} \cdot e(h_1, g_2)^{r_f} \cdot e(h_2, g_2)^{r_b} \cdot e(h_2, w)^{r_a}, \\ &:= e(R_{2t} \cdot T^{-r_x} \cdot h_2^{r_b}, g_2) \cdot T_4^{r_a}. \end{aligned}$$

7. $\mathcal{H}$ computes $c_h := H_4(\mathtt{gpk}\|B\|K\|T\|R_1\|R_2\|n_V)$ and sends c_h to $\mathcal{M}$.
8. $\mathcal{M}$ chooses a random nonce $n_T \leftarrow \{0,1\}^t$ and computes $c := H_5(c_h\|n_T\|m)$.
9. $\mathcal{M}$ computes in $s_f := r_f + c \cdot f \pmod p$.
10. $\mathcal{M}$ sends (c, n_T, s_f) to $\mathcal{H}$. $\mathcal{M}$ erases r_f after sending this message.
11. $\mathcal{H}$ computes

$$s_x := r_x + c \cdot x \pmod p, \qquad s_a := r_a + c \cdot a \pmod p, \qquad s_b := r_b + c \cdot b \pmod p.$$

12. $\mathcal{H}$ outputs $\sigma := (B, K, T, c, n_T, s_f, s_x, s_a, s_b)$.

Note that the signing protocol is a three-message protocol: In the first message, $\mathcal{M}$ sends (B, K, R_1, R_{2t}) to $\mathcal{H}$. In the second message, $\mathcal{H}$ sends c_h to $\mathcal{M}$. In the third message, $\mathcal{M}$ sends (c, n_T, s_f) to $\mathcal{H}$. The way we divide the computation between the TPM and the host is similar to the one in the original DAA paper [5].

As pointed out by Chen et al. [13] recently, such signing protocols are vulnerable to "replay" attack. That is, if a malicious host engages with a TPM on the signing protocol by submitting a c_h value and obtains (c, n_T, s_f) from the TPM. The host then re-submits c_h again and somehow tricks the TPM to start from step 8 of the protocol, then the TPM will output a different set (c', n'_T, s'_f). The host can figure the private key f from (c, c', s_f, s'_f). In fact, this type of vulnerability applies to any interactive zero-knowledge proof protocol. If attacker can replay the challenge, then he can extract the prover's knowledge. Such vulnerability can be easily mitigated using stage control mechanism. For example, if the TPM receives the second message c_h from the host, it checks whether it has a pending signing protocol. If not, it simply rejects c_h. In our scheme, we use an even simpler method: the TPM deletes r_f after sending s_f to the host. This can prevent the replay attack.

4.4 Verify Algorithm

On input of a message m, a basename $\mathtt{bsn}$, a nonce n_V, a signature $(B, K, T, c, n_T, s_f, s_x, s_a, s_b)$, the public key $\mathtt{gpk}$, and the revocation list $\mathtt{RL}$ (a list of revoked secret keys), the verification algorithm takes the following steps:

1. Verify that $B, K, T \in G_3$ and $s_f, s_x, s_a, s_b \in \mathbb{Z}_p$.
2. Compute $\hat{R}_1 := B^{s_f} \cdot K^{-c}$.
3. Compute

$$\begin{aligned}\hat{R}_2 &:= e(T, g_2)^{-s_x} \cdot e(h_1, g_2)^{s_f} \cdot e(h_2, g_2)^{s_b} \cdot e(h_2, w)^{s_a} \cdot (e(g_1, g_2)/e(T, w))^c. \\ &:= e(T, g_2^{-s_x} \cdot w^{-c}) \cdot T_1^c \cdot T_2^{s_f} \cdot T_3^{s_b} \cdot T_4^{s_a}\end{aligned}$$

4. Verify that

$$c \stackrel{?}{=} H_5(H_4(\mathtt{gpk}\|B\|K\|T\|\hat{R}_1\|\hat{R}_2\|n_V)\|n_T\|m).$$

5. For each $f' \in \mathtt{RL}$, if $K = B^{f'}$, output 0 (reject).
6. If any of the above verifications fails, output 0 (reject), otherwise, output 1 (accept).

4.5 Link Algorithm

On input of two message-signature pairs (m_0, σ_0) and (m_1, σ_1), a basename $\mathtt{bsn}$, and the public key $\mathtt{gpk}$, the link algorithm performs the following steps:

1. For each signature σ_b where $b \in \{0, 1\}$, run the verify algorithm $\mathsf{Verify}(\sigma_b, m_b, \mathtt{bsn}, \mathtt{gpk})$. If either of two verifications returns 0 (reject), output $\perp$.
2. If $(B, K) \in \sigma_0$ are the same as $(B, K) \in \sigma_1$, return 1 (linked), otherwise return 0 (unlinked).

5 Security Proof

In this section, we prove our DAA scheme is secure under the security definitions stated in Section 2. We show that our DAA scheme is correct, user-controlled-anonymous, and user-controlled-traceable. The security of the DAA scheme based on the q-SDH assumption and G_1-DDH assumption defined in Section 3.

Theorem 1 *The DAA scheme in Section 4 is correct.*

Proof. To show the DAA scheme is correct, we prove that a signature created by a valid and unrevoked signer can be successfully verified by any verifier. In order to have a success signature verification, $\hat{R}_1, \hat{R}_2$ in the verify algorithm must be equal to R_1, R_2 in the sign protocol, respectively. We prove $\hat{R}_1 = R_1$ and $\hat{R}_2 = R_2$ as follows.

$$\begin{aligned}
\hat{R}_1 &= B^{s_f} \cdot K^{-c} = B^{r_f} \cdot B^{cf} \cdot (B^f)^{-c} = B^{r_f} = R_1 \\
\hat{R}_2 &= e(T, g_2)^{-s_x} \cdot e(h_1, g_2)^{s_f} \cdot e(h_2, g_2)^{s_b} \cdot e(h_2, w)^{s_a} \cdot (e(g_1, g_2)/e(T, w))^c \\
&= R_2 \cdot e(T, g_2)^{-cx} \cdot e(h_1, g_2)^{cf} \cdot e(h_2, g_2)^{cb} \cdot e(h_2, w)^{ca} \cdot (e(g_1, g_2)/e(T, w))^c \\
&= R_2 \cdot (e(g_1, g_2) \cdot e(h_1, g_2)^f \cdot e(h_2, g_2)^b \cdot e(h_2, w)^a \cdot e(T, g_2)^{-x} \cdot e(T, w)^{-1})^c \\
&= R_2 \cdot (e(g_1, g_2) \cdot e(h_1, g_2)^f \cdot e(h_2, g_2)^{ax} \cdot e(h_2, w)^a \cdot e(T, g_2^x w)^{-1})^c \\
&= R_2 \cdot (e(g_1 h_1^f, g_2) \cdot e(h_2^a, g_2^x w) \cdot e(A, g_2^x w)^{-1} \cdot e(h_2^a, g_2^x w)^{-1})^c = R_2
\end{aligned}$$

The last equation holds because for a valid private key (A, x, f), $e(A, g_2^x w) = e(g_1 h_1^f, g_2)$ holds. We now show that two signatures created by a single signer using a basename $\texttt{bsn} \neq \perp$ can be linked. This is obvious from the description of the DAA scheme, as two signatures will have the same (B, K) pair if the signatures are created using the same private key f. □

Theorem 2 *Under the G_1-DDH assumption, the DAA scheme in Section 4 is user-controlled-anonymous. More specifically, if there is an adversary $\mathcal{A}$ that succeeds with a non-negligible probability to break the user-controlled-anonymity game, then there is a polynomial-time algorithm $\mathcal{B}$ that solves the G_1-DDH problem with a non-negligible probability.*

Proof. Suppose an algorithm $\mathcal{A}$ breaks the user-controlled-anonymity game of the DAA scheme with non-negligible probability. We can build a polynomial-time simulator $\mathcal{B}$ that breaks the G_1-DDH problem as follows. $\mathcal{B}$ is given as input a tuple $(u, v = u^a, w = u^b, z)$ where $u \leftarrow G_1$, $a, b, \leftarrow \mathbb{Z}_p$, and either $z = u^{ab}$ or z is a random element in G_1. $\mathcal{B}$ decides which z was given by interacting with $\mathcal{A}$ as follows.

We first give an overview of the proof. $\mathcal{B}$ first creates a special signer S^* where its secret key $f = \log_u v$, however $\mathcal{B}$ does not know the secret key. $\mathcal{B}$ creates rest of the signers by running the join protocol with $\mathcal{A}$. To respond to a sign query for signer S^*, $\mathcal{B}$ simulates the signature using the (u, v) pair. In the challenge phase, if S^* is selected as one of the (S_0, S_1) pair, $\mathcal{B}$ picks S^* for creating a signature by simulating the signature using the (w, z) pair, i.e., simulates using the secret key $f = \log_w z$. If $z = u^{ab}$, then $\log_u v = log_w z$, $\mathcal{A}$ has non-negligible advantage guessing the random bit b correctly. If $z \neq u^{ab}$, $\mathcal{A}$ does not have any advantage guessing b or $\mathcal{A}$ may abort the game. $\mathcal{B}$ can use the output of $\mathcal{A}$ to decide whether $z = u^{ab}$.

Setup. Let (G_1, G_2) be a bilinear group pair of prime order p with generator g_1 and g_2, respectively. $\mathcal{B}$ chooses a random $\gamma \leftarrow \mathbb{Z}_p^*$ as $\texttt{isk}$ and sets the public key $\texttt{gpk} = (G_1, G_2, G_T, p, e, g_1, h_1 := u, h_2, g_2, w := g_2^\gamma, H_1, H_2, H_3, H_4, H_5, T_1, T_2, T_3, T_4)$ by running the setup algorithm. $\mathcal{B}$ sends $\texttt{isk}$ and $\texttt{gpk}$ to $\mathcal{A}$.

Hash Queries. We model the hash functions H_2, H_3, and H_5 as three random oracles. $\mathcal{B}$ responds to the hash queries for H_2, H_3, and H_5 as follows.

- $H_2(m)$: If m has not been queried before, $\mathcal{B}$ chooses $H_2(m)$ uniformly at random from $\mathbb{Z}_p^*$ and returns it to $\mathcal{A}$, otherwise $\mathcal{B}$ returns the previously queried result on m to ensure consistency.

- $H_3(m)$: Let q_h be the expected number of unique H_3 queries. $\mathcal{B}$ chooses a random $i \leftarrow \{1, \ldots, q_h\}$. If m has been queried before, $\mathcal{B}$ returns the previously queried result on m to ensure consistency. Otherwise, if m is the i-th unique query on H_3, $\mathcal{B}$ chooses chooses a random $r \leftarrow \mathbb{Z}_p^*$ and sets $H_3(m) := w^r$. For rest of the queries, $\mathcal{B}$ chooses a random $r \leftarrow \mathbb{Z}_p^*$ and sets $H_3(m) := u^r$. We use $\mathtt{bsn}^*$ to denote the i-th unique query.
- $H_5(m)$: $\mathcal{B}$ chooses $H_5(m)$ uniformly at random from $\mathbb{Z}_p^*$ while ensuring consistency.

Join Queries. $\mathcal{A}$ requests for creating a new signer S. Let q_j be the expected number of join requests from $\mathcal{A}$. $\mathcal{B}$ chooses a random $i \leftarrow \{1, \ldots, q_j\}$. There are two cases for $\mathcal{B}$ to respond:

- If the query is the i-th join query: $\mathcal{B}$ sets $F := v$ without knowing the secret key $f = \log_u v$, and then forges rest of the join protocol as follows: it chooses randomly $c, s_f \leftarrow \mathbb{Z}_p$ and computes $R = h_1^{s_f} \cdot F^{-c}$. It then patches the oracle by setting $H_2(\mathtt{gpk} \| n_I \| F \| R) := c$. If $H_2(\mathtt{gpk} \| n_I \| F \| R)$ has been queried before, $\mathcal{B}$ quites and outputs "**abortion 0**". $\mathcal{B}$ receives a credential from $\mathcal{A}$. We use S^* to denote the identity of this signer.
- If the query is not the i-th join query: $\mathcal{B}$ chooses a random $f \leftarrow \mathbb{Z}_p^*$, computes $F := h_1^f$. If $F = v$, $\mathcal{B}$ quites and outputs "**abortion 0**". $\mathcal{B}$ runs the rest of the join protocol as the signer with $\mathcal{A}$ as the issuer, and obtains a credential $\mathtt{cre} = (A, x)$. $\mathcal{B}$ verifies $\mathtt{cre}$ and stores (S, f, A, x) in its log.

Sign Queries. Given a signer's identity S, a message m to be signed, a nonce n_V from $\mathcal{A}$, a basename $\mathtt{bsn}$, $\mathcal{B}$ responds with a signature σ as follows: Assuming the signer S has already joined, if S is not S^*, $\mathcal{B}$ finds the corresponding secret key and credential (f, A, x) associated with S, runs the sign protocol, and outputs σ to $\mathcal{A}$. If $S = S^*$, $\mathcal{B}$ needs to forge a signature as follows:

1. If $\mathtt{bsn} = \perp$, $\mathcal{B}$ chooses a random $r \leftarrow \mathbb{Z}_p$ and sets $B := u^r$ and $K := v^r$.
2. If $\mathtt{bsn} = \mathtt{bsn}^*$, $\mathcal{B}$ quits and outputs "**abortion 1**".
3. If $\mathtt{bsn} \neq \{\perp, \mathtt{bsn}^*\}$, $\mathcal{B}$ searches the log of H_3 queries and retrieves r where $H_3(\mathtt{bsn}) = u^r$. $\mathcal{B}$ sets $B := u^r$ and computes $K := v^r$.
4. $\mathcal{B}$ chooses $T \leftarrow G_1$, $n_T \leftarrow \{0,1\}^t$, and $c, s_f, s_x, s_a, s_b \leftarrow \mathbb{Z}_p$.
5. $\mathcal{B}$ computes $R_1 := B^{s_f} \cdot K^{-c}$.
6. $\mathcal{B}$ computes $R_2 := e(T, g_2)^{-s_x} \cdot e(h_1, g_2)^{s_f} \cdot e(h_2, g_2)^{s_b} \cdot e(h_2, w)^{s_a} \cdot (e(g_1, g_2)/e(T, w))^c$.
7. $\mathcal{B}$ patches the oracle H_5 by setting $H_5(H_4(\mathtt{gpk} \| B \| K \| T \| R_1 \| R_2 \| n_V) \| n_T \| m) := c$. If $H5(H_4(\mathtt{gpk} \| B \| K \| T \| R_1 \| R_2 \| n_V) \| n_T \| m)$ has been queried before, $\mathcal{B}$ quites and outputs "**abortion 0**".
8. $\mathcal{B}$ outputs the signature $\sigma := (B, K, T, c, n_T, s_f, s_x, s_a, s_b)$.

Corrupt Queries. If a corrupt query is for a signer $S \neq S^*$, then $\mathcal{B}$ responds with the secret key corresponding to S. Otherwise, $\mathcal{B}$ quits and outputs "**abortion 2**".

Challenge. In the challenge, $\mathcal{A}$ outputs a message m, a basename $\mathtt{bsn}$, and two signer's identity S_0 and S_1. If $S^* \notin \{S_0, S_1\}$ or $\mathtt{bsn} \notin \{\perp, \mathtt{bsn}^*\}$, then $\mathcal{B}$ quits and outputs "**abortion 3**". Otherwise, $\mathcal{B}$ picks $b \in \{0, 1\}$ such that $S_b = S^*$, and generates a signature σ^* for m by performing the following steps:

1. If $\mathtt{bsn} = \perp$, $\mathcal{B}$ chooses a random $r \leftarrow \mathbb{Z}_p$ and sets $B := w^r$ and $K := z^r$.
2. If $\mathtt{bsn} = \mathtt{bsn}^*$, $\mathcal{B}$ searches the log of H_3 queries and retrieves r where $H_3(\mathtt{bsn}^*) = w^r$. $\mathcal{B}$ sets $B := w^r$ and computes $K := z^r$.
3. The rest of the sign algorithm follows the sign queries above.

$\mathcal{B}$ sends the resulting σ^* to $\mathcal{A}$.

Output. In the end, $\mathcal{A}$ outputs $b' \in \{0, 1\}$ as the guess for b or aborts without any output. If $b = b'$, then $\mathcal{B}$ outputs 1, which means that $z = u^{ab}$. Otherwise $\mathcal{B}$ outputs 0, which means that z is a random element in G_1.

We now discuss the probability that algorithm $\mathcal{B}$ does not abort in the above game. There are four cases where $\mathcal{B}$ can abort. We study each case as follows:

1. Abortion 0. The chance of this type of abortion is $O(1/p)$. Since p is a large prime, the probability of this abortion is negligible.

2. Abortion 1. Recall that $\mathcal{A}$ cannot use the same non-empty $\mathtt{bsn}$ in the sign query and challenge query for signers S_0 and S_1. In other words, $\mathcal{A}$ cannot query all possible $\mathtt{bsn}$ for S^* in the sign queries. The probability that $\mathcal{B}$ does not abort in this case is at least $1/q_h$.

3. Abortion 2. As $\mathcal{A}$ cannot corrupt all the signers, the probability that $\mathcal{B}$ does not abort is at least $1/q_j$.

4. Abortion 3. $\mathcal{B}$ does not abort in this case if $\mathcal{A}$ selects S^* and $\mathtt{bsn}^*$ in the challenge query. Thus the probability that $\mathcal{B}$ does not abort in this case is $1/(q_h \cdot q_j)$.

$\mathcal{B}$ does not abort if (1) $\mathtt{bsn}^*$ was not chosen in the sign queries for S^*, (2) S^* was not chosen in the corrupt queries, and (3) S^* and $\mathtt{bsn}^*$ were chosen in the challenge query. The probability that $\mathcal{B}$ does not abort the above game is roughly $1/(q_h \cdot q_j)$.

Let ϵ be the probability that $\mathcal{A}$ succeeds in breaking the user-controlled-anonymity game. Suppose $\mathcal{B}$ does not abort during the above simulation. If $z = u^{ab}$, then $\log_u v = \log_w z$, $\mathcal{B}$ simulates the game perfectly, i.e., $\Pr[b = b'] > \frac{1}{2} + \epsilon$. If z is a random element in G_1, then σ^* in the challenge query is simulated using the (w, z) pair. In other words, the secret key used in generated σ^* is different from either secret key of S_0 or S_1. Observe that $\mathcal{B}$ in this case does not simulate the game perfectly, especially in the challenge query. $\mathcal{A}$ could abort the game. If $\mathcal{A}$ does not abort the game, $\mathcal{A}$ does not have any advantage guessing b. It follows that $\Pr[b = b'] = \frac{1}{2}$. Therefore, assuming $\mathcal{B}$ does not abort, it has probability at least $\epsilon/2$ in solving the DDH problem in G_1. □

Theorem 3 *Under the q-SDH assumption, the DAA scheme in Section 4 is user-controlled-traceable. More specifically, if there is an adversary $\mathcal{A}$ that succeeds with a non-negligible probability to break the user-controlled-traceability game, then there is a polynomial-time algorithm $\mathcal{B}$ that solves the q-SDH problem with a non-negligible probability.*

Proof. The proof of this theorem derives from the user-controlled-traceability proof in Chen DAA scheme [11]. We show how to construct an algorithm $\mathcal{B}$ that solves the q-SDH problem by interacting with $\mathcal{A}$ that succeeds with a non-negligible probability to break the user-controlled-traceability game. We use a technique from Boneh and Boyen [3] that, given $g_1 \in G_1$, $g_2 \in G_2$, $w = g_2^{\gamma}$, and $q - 1$ SDH pairs (B_i, x_i) such that $e(B_i, w g_2^{x_i}) = e(g_1, g_2)$, one more SDH pair (B, x) can be transformed into a solution to the q-SDH problem. We now describe how $\mathcal{B}$ interacts with $\mathcal{A}$ as follows:

Setup. $\mathcal{B}$ runs the setup algorithm as follows: Let $e : G_1 \times G_2 \rightarrow G_T$ be a bilinear map function. $\mathcal{B}$ is given (g_1, g_2, w) and $q - 1$ SDH pairs (B_i, x_i) as input, where $g_1 = \psi(g_2)$, $w = g_2^{\gamma}$, and $B_i = g_1^{1/(\gamma + x_i)}$. $\mathcal{B}$ chooses $a, b, x \leftarrow \mathbb{Z}_p^*$ and computes

$$h_1 := \psi(((w \cdot g_2^x)^b \cdot g_2^{-1})^{1/a}) = g_1^{((\gamma+x)b-1)/a}.$$

$\mathcal{B}$ sets the public key $\mathtt{gpk} = (G_1, G_2, G_T, p, e, g_1, h_1, h_2, g_2, w, H_1, H_2, H_3, H_4, H_5, T_1, T_2, T_3, T_4)$. Note that $\mathcal{B}$ does not know the value of $\mathtt{isk} = \gamma$. $\mathcal{B}$ sends $\mathtt{gpk}$ to $\mathcal{A}$.

Hash Queries. We model the hash functions H_2, H_3, and H_5 as three random oracles. $\mathcal{B}$ responds to the hash queries for H_2, H_3, and H_5 as follows.

- $H_2(m)$: $\mathcal{B}$ chooses $H_2(m)$ uniformly at random from $\mathbb{Z}_p^*$ while ensuring consistency.
- $H_3(m)$: If m has not been queried before, $\mathcal{B}$ chooses $H_2(m)$ uniformly at random from G_1 and returns it to $\mathcal{A}$, otherwise $\mathcal{B}$ returns the previously queried result on m to ensure consistency.
- $H_5(m)$: $\mathcal{B}$ chooses $H_5(m)$ uniformly at random from $\mathbb{Z}_p^*$ while ensuring consistency.

Join Queries. $\mathcal{A}$ requests for creating a new signer S. There are two cases of the join queries as defined by the user-controlled-traceability game in Section 2:

1. Given a new signer S, $\mathcal{B}$ has freedom to choose $\mathtt{sk}$ of its choice and then creates $\mathtt{cre}$.
2. Given a new signer S and a secret key $\mathtt{sk}$ chosen by the $\mathcal{A}$, $\mathcal{B}$ creates a credential $\mathtt{cre}$ for $\mathtt{sk}$. $\mathcal{B}$ sends $\mathtt{cre}$ to $\mathcal{A}$ and put $\mathtt{sk}$ in the revocation list $\mathtt{RL}$.

For case 1, $\mathcal{B}$ selects one query randomly and sets $f := a$ and $\mathtt{cre} := (A, x)$ where $A := g_1^b$. For rest of the queries, either $\mathcal{B}$ chooses f_i at random (case 1) or receives f_i from $\mathcal{A}$, $\mathcal{B}$ uses one SDH pair (B_i, x_i) to computes $\mathtt{cre}_i := (A_i, x_i)$ for f_i as follows:

$$A_i = (g_1 \cdot h_1^{f_i})^{1/(\gamma+x_i)} = B_i^{1-f_i/a+f_i b(x+\gamma)/a} := B_i^{1-f_i/a+f_i b(x-x_i)/a} \cdot g_1^{f_i b/a}.$$

Corrupt Queries. $\mathcal{A}$ requests to corrupt a signer S. We assume that the signer S has been joined before, $\mathcal{B}$ responds with the secret key $\mathtt{sk}$ and credential $\mathtt{cre}$ corresponding to S. $\mathcal{B}$ inserts $\mathtt{sk}$ to the revocation list $\mathtt{RL}$.

Sign Queries. Given a signer's identity S, a message m to be signed, a nonce n_V from $\mathcal{A}$, a basename $\mathtt{bsn}$, $\mathcal{B}$ computes the signature σ by invoking the sign protocol. $\mathcal{B}$ returns σ to $\mathcal{A}$.

Semi-sign Queries. Given a signer's identity S, a message m to be signed, a nonce n_V from $\mathcal{A}$, a basename $\mathtt{bsn}$, $\mathcal{B}$ runs the sign protocol as a TPM by interacting with $\mathcal{A}$ as a host. $\mathcal{B}$ computes B, chooses r_f at random, computes K, R_1, R_{2t}, and sends (B, K, R_1, R_{2t}) to $\mathcal{A}$. On receiving c_h from $\mathcal{A}$, $\mathcal{B}$ chooses n_T at random and computes c and s_f.

Output. In the end, $\mathcal{A}$ outputs a signer S, a message m, a basename $\mathtt{bsn}$, and a signature σ. There are two cases to consider:

1. If σ can be successfully verified, i.e., $\mathsf{Verify}(m, \mathtt{bsn}, \sigma, \mathtt{gpk}, \mathtt{RL}) = 1$, $\mathcal{B}$ can rewind $\mathcal{A}$ to extract the underlying the signing key (f^*, A^*, x^*) such that $e(A^*, wg_2^{x^*}) = e(g_1 h_1^{f^*}, g_2)$. To rewind $\mathcal{A}$, $\mathcal{B}$ controls the result of H_5 by choosing two different c values c' and c'' to the same B, K, T, R_1, R_2. $\mathcal{A}$ responds with s'_f, s'_x, s'_a, s'_b and $s''_f, s''_x, s''_a, s''_b$. $\mathcal{B}$ computes $\Delta = c' - c''$ and extracts

$$f^* := (s'_f - s''_f)/\Delta, \qquad x^* := (s'_x - s''_x)/\Delta, \qquad a^* := (s'_a - s''_a)/\Delta, \qquad A^* := T/(h_2^{a^*}).$$

Also note that $f^* \notin \mathtt{RL}$, otherwise, signature verification would fail. There are two possible results

(a) If $x^* \notin \{x_i, x\}$ for any i, $\mathcal{B}$ can compute

$$B^* := (A^* \cdot g_1^{-bf^*/a})^{a/(a-f^*+bf^*(x-x^*))}.$$

(b) If $x^* \in \{x_i, x\}$ and $A^* \notin \{A_i, A\}$ for any i. If $e^* \neq e$, $\mathcal{B}$ aborts and outputs failure. With the probability of $1/q$, $e^* = e$, $\mathcal{B}$ computes

$$B^* := (A^* \cdot g_1^{-bf^*/a})^{a/(a-f^*)}.$$

In either of the above two cases, $\mathcal{B}$ obtains (B^*, x^*) as the extra SDH pair so that it can solve the given q-SDH problem.

2. If $\mathtt{bsn} \neq \perp$, $\mathcal{A}$ wins in this case if there exists another signature σ' by the signer S using the same basename $\mathtt{bsn}$ from the sign queries, such that $\mathsf{Link}(\sigma, \sigma', \mathtt{gpk}) = 0$. Let $\sigma = (B, K, T, c, n_T, s_f, s_x, s_a, s_b)$ and $\sigma' = (B', K', T', c', n'_T, s'_f, s'_x, s'_a, s'_b)$. Since the same $\mathtt{bsn}$ is used, $B = B'$. Because two signatures are unlinkable, thus $K \neq K'$. This means $\mathcal{A}$ managed to create a different $\mathtt{sk}$ for the signer S. $\mathcal{B}$ can use the same method as case (1) to extract a different (f, A, x) tuple as a valid secret key and credential pair, thus obtain an extra SDH pair. $\mathcal{B}$ can also solve the q-SDH problem in this case.

In either of the above two cases, $\mathcal{B}$ can solve the q-SDH problem with a non-negligible probability if $\mathcal{A}$ can wins the game with a non-negligible probability. □

6 Comparisons with Existing DAA Schemes

In this section, we compare our DAA scheme with several existing pairing-based DAA schemes [6, 13, 15, 11]. Note that we do not include the original CMS-DAA scheme [12] in the comparisons, as it is not secure, instead we compare ours with the patched version [13]. We also do not include the pairing-based EPID scheme [9] in the comparison, because EPID does not have the feature of splitting computation between a TPM and a host.

We compare the credential and signature sizes of our scheme with other DAA schemes in Table 1. We use $\mathbb{Z}_q$ to denote the size of an element in $\mathbb{Z}_q$, h to denote the size of a hash result, G_1 to denote the size of an element in G_1, and G_T to denote the size of an element in G_T. For bilinear maps with 128-bit security, G_T needs to be around 3072 bits [17]. Our DAA scheme has the same credential and signature sizes as in Chen's DAA scheme [11]. The credential and signature sizes in our scheme are smaller than other DAA schemes [6, 13, 15].

DAA Scheme	Credential Size	Signature Size
Scheme of [6]	$3G_1$	$2\mathbb{Z}_q + 3G_1 + 2G_T + 1h$
Scheme of [13]	$3G_1$	$1\mathbb{Z}_q + 5G_1 + 1h$
Scheme of [15]	$2\mathbb{Z}_q + 1G_1$	$6\mathbb{Z}_q + 2G_1 + 2G_T + 1h$
Scheme of [11]	$1\mathbb{Z}_q + 1G_1$	$4\mathbb{Z}_q + 3G_1 + 1h$
Our Scheme	$1\mathbb{Z}_q + 1G_1$	$4\mathbb{Z}_q + 3G_1 + 1h$

Table 1: A comparison between our DAA scheme and other DAA schemes in credential and signature sizes

We compare the efficiency of signing and verification algorithms of our scheme with other DAA schemes in Table 2. We use P to denote a pairing operation, G_1 to denote an exponentiation operation in G_1, G_1^2 to denote a multi-exponentiation operation, and so on. A multi-exponentiation is slightly more expensive than an exponentiation. As we mentioned earlier in Section 1, operations in G_1 are much more efficient than ones in G_T. Therefore, our DAA scheme has significant advantage for computationally weak device such as TPM. The efficiency of the sign protocol for our scheme is approximately 5 times more efficient than the rest of pairing-based DAA schemes [6, 13, 15, 11] using Barreto-Naehrig curves.

DAA Scheme	TPM Sign	Host Sign	Verify
Scheme of [7]	$3G_T$	$3G_1 + 1G_T + 3P$	$1G_T^2 + 1G_T^3 + 5P + (n+1)G_T$
Scheme of [13]	$2G_1 + 1G_T$	$3G_1 + 1P$	$1G_T^2 + 1G_T^2 + 5P + nG_1$
Scheme of [15]	$2G_1 + 1G_T^2$	$1G_1 + 2G_1^2 + 1G_1^3 + 1G_T^3$	$1G_1^2 + 2G_1^3 + 1G_T^5 + 3P + nG_T$
Scheme of [11]	$2G_1 + 1G_T$	$1G_1 + 1G_T^3$	$1G_1^2 + 1G_2^2 + 1G_T^4 + 1P + nG_1$
Our Scheme	$3G_1$	$1G_1 + 1G_1^2 + 1G_T + 1P$	$1G_1^2 + 1G_2^2 + 1G_T^4 + 1P + nG_1$

Table 2: A comparison between our DAA scheme and other DAA schemes in the efficiency of sign and verify algorithms

Observe that the efficiency we gain in TPM comes with a price – the host needs to perform an additional pairing. The total efficiency of the sign protocol is $4G_1 + 1G_1^2 + 1G_T + 1P$ in our scheme, whereas it is $3G_1 + 1G_T + 3G_T^3$ in Chen's DAA scheme [11]. Note that TPM is much slower than the host platform, e.g., probably 100 times slower or more. Based on the performance simulation by Chen, Page, and Smart [14], a pairing operation on a host takes less than 15 millie-seconds for a 64-bit 2.4 GHz Intel Core 2 processor, while $2G_1 + 1G_T$ operation takes more than 3 seconds for a 32-bit 33 MHz simulated TPM. Base on these performance data, we estimate that the overall sign protocol will take less than 1 second in our scheme, whereas the sign protocol in [11, 13] will take more than 3 seconds.

We did not compare the efficiency of the join protocol because the join protocol is executed much less frequently than the sign protocol or the verification algorithm. Besides the join protocol in our DAA scheme is the same as the one in Chen's DAA scheme [11], thus has the same efficiency.

7 Conclusions

In this paper, we proposed an efficient DAA scheme from bilinear maps. Our new DAA scheme takes much less resources for TPM implementation (both computation and software code size) compared to the existing DAA schemes. We believe our DAA scheme is a good candidate for the next generation of TPM initiated by the TCG TPM working group, assuming that a TPM can support multiple DAA algorithms.

References

[1] Michael Backes, Matteo Maffei, and Dominique 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.

[2] Paulo S. L. M. Barreto and Michael 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.

[3] 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.

[4] 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.

[5] 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.

[6] 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.

[7] Ernie Brickell, Liqun Chen, and Jiangtao Li. Simplified security notions of direct anonymous attestation and a concrete scheme from pairings. *International Journal of Information Security*, 8(5):315–330, 2009.

[8] 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*, pages 21–30. ACM Press, October 2007.

[9] Ernie Brickell and Jiangtao Li. Enhanced Privacy ID from bilinear pairing. Cryptology ePrint Archive, Report 2009/095, 2009. `http://eprint.iacr.org/`.

[10] 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.

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

[12] 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.

[13] Liqun Chen, Paul Morrissey, and Nigel P. Smart. DAA: Fixing the pairing based protocols. Cryptology ePrint Archive, Report 2009/198, 2009. `http://eprint.iacr.org/`.

[14] Liqun Chen, Dan Page, and Nigel P. Smart. On the design and implementation of an efficient DAA scheme. Cryptology ePrint Archive, Report 2009/598, 2009. `http://eprint.iacr.org/`.

[15] Xiaofeng Chen and Dengguo Feng. Direct anonymous attestation for next generation tpm. *Journal of Computers*, 3(12):43–50, 2008.

[16] 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.

[17] 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.

[18] 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.

[19] Anna Lysyanskaya, Ronald L. Rivest, Amit Sahai, and Stefan Wolf. Pseudonym systems. In *Proceedings of the 6th Workshop on Selected Areas in Cryptography*, volume 1758 of *LNCS*, pages 184–199. Springer, 1999.

[20] 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.

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

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