An Efficient Direct Anonymous Attestation Scheme with Forward Security

DENG-GUO FENG, JING XU, XIAO-FENG CHEN
State Key Laboratory of Information Security
Institute of Software, Chinese Academy of Sciences
4# South Fourth Street, Zhong Guan Cun, Beijing 100190
CHINA
feng@is.iscas.ac.cn, xujing@is.iscas.ac.cn, chenxiaof@is.iscas.ac.cn

Abstract: Direct Anonymous Attestation (DAA) is a cryptographic mechanism adopted by the Trusted Computing Group in its specifications for trusted computing platforms (TCP). In this paper, we propose a new DAA scheme and prove it is secure under the strong RSA assumption and the decisional Diffie-Hellman assumption. While satisfying all the security properties proposed in previous DAA schemes, our scheme provides a new desired security property, forward security: compromise of the current private key of TPM does not enable an adversary to forge signatures pertaining to the past. Such forward security is important to mitigate the damage caused by private key exposure.

Key-Words: Trusted computing platform, Direct anonymous attestation, Forward security, Trusted platform module

1 Introduction

Trusted computing platforms have been proposed as a promising approach to enhance the security [1-4] of general-purpose computing systems. A trusted computing platform is a computing device integrated with a cryptographic chip called a trusted platform module (TPM), which is designed and manufactured in a way such that all parties can trust cryptographic computing results from this TPM. When a TPM is manufactured, a unique asymmetric key pair, called the Endorsement Key (EK), is created and stored in the protected area of the TPM. If the authentication of a TPM is directly based on its EK, all transactions by the same TPM can be linked and the TCP may suffer a loss of privacy. In order to address privacy concerns resulting from routine use of an EK, two approaches have been proposed in the TPM specifications.

The first approach in the TPM Specification Version 1.1 is based on a trusted third party, referred to as a Privacy Certification Authority (Privacy CA), when a TPM needs to authenticate itself to a verifier, it generates a RSA key pair called an Attestation Identity Key (AIK), sends the AIK public key to the Privacy CA, and authenticates this public key using a valid EK. The Privacy CA will check whether it finds the EK in its list and, if so, issue a certificate on the TPM's AIK. The TPM can then forward this certificate to the verifier and authenticate itself. In this way, the TPM hides its identity during the transaction. This approach has the obvious drawback that the Privacy CA needs to be involved in every transaction. Moreover, the compromise of the Privacy CA (or a dishonest Privacy CA) can destroy all privacy guarantees.

The second approach, Direct Anonymous Attestation (DAA), was introduced to counteract this drawback. In the DAA scheme, there exists an issuer who creates a group public key. The first DAA scheme was proposed by Brickell et al. [6] and later was standardized in TPM Specification Version 1.2 [5]. However, their scheme is too inefficient to be suitable for embedded devices with limited computing capabilities. In 2007, Ge et al. [7] presented a new construction with more efficient sign and verify protocols, which is more attractive for embedded devices. Both scheme are provably secure in the random oracle model under the strong RSA and the Decisional Diffie-Hellman (DDH) assumptions. Recently, Chen et al. [8] also proposed an efficient DAA scheme based on the Strong Diffie-Hellman (SDH) and DDH assumptions. Their scheme is more efficient in signature length and in computational complexity.

In fact, DAA can be seen as a group signature[9-11] without the capability to open signatures, i.e.,
the anonymity is not revocable. Moreover, DAA allows for detection of “known” keys: if the DAA secret keys are extracted from a TPM and published, a verifier can detect that a signature was produced using these secret keys. In prior DAA schemes, if a private key of TPM is exposed to an attacker, all previously obtained signatures will be invalid, because one cannot distinguish whether a signature is generated by an attacker after it obtained the private key or by the legitimate signer before the attacker obtained the private key.

In this paper, we propose to use the concept of forward secure to reduce the damage of exposure of TPM’s private key, i.e. even when a private key is exposed, previously generated signatures remain valid and do not need to be re-signed. In particular our construction is built up from the group signature scheme due to Ateniese et al. [11]. The concept of forward secure signatures was first proposed by Anderson [12] for traditional signatures, and the first forward secure group signature scheme was presented by Song [13] in 2001. Similarly, our forward secure DAA scheme shares many properties with the forward secure group signature schemes. However, unlike those schemes, the issuer can not open the signer's identity and thus the signature should also be forward secure for the issuer. In a forward secure DAA scheme, the private key of TPM evolve over time: in time period $i$, TPM’s private key evolves from $f_{i-1}$ to $f_i$ using a public one-way function, and then $f_{i-1}$ is erased from the system.

The remainder of this paper is organized as follows. The next section reviews some cryptographic assumptions and building blocks of our proposed scheme. Section 3 introduces the model of our construction. We then propose a forward secure DAA scheme in Section 4, whose security and performance are analyzed in Section 5. Section 6 concludes.

2 Preliminaries
This section reviews some cryptographic assumptions and introduces the building blocks necessary in the subsequent design of our DAA scheme.

2.1 Cryptographic Assumptions

The security of our DAA scheme relies on the Strong-RSA assumption and the Decisional Diffie-Hellman (DDH) assumption.

**Definition1 (Strong RSA Assumption)** The Strong RSA Assumption states that it is computational infeasible, on input a random RSA modulus $n$ and a random element $u \in \mathbb{Z}_n^*$, to compute values $e > 1$ and $v$ such that $v^e = u \mod n$.

**Definition2 (DDH Assumption)** There is no probabilistic polynomial-time algorithm that distinguishes with non-negligible probability between the distribution $D$ and $R$, where $D = (g, g^x, g^y, g^z)$ and $R = (g, g^x, g^{xy}, g^{yz})$ with $x, y, z \in \mathbb{Z}_n$.

2.2 Signature of Knowledge
In our scheme we will use the signature of knowledge that allows a prover to demonstrate the knowledge of a secret w.r.t. some public information such that no other information is revealed in the process. To describe these protocols, we use notation introduced by Camenisch et al. [14] for various proofs of knowledge. For example, $\text{PK}\{ (a,b): y_1 = g_1^a h_1^b \land y_2 = g_2^a h_2^b \}$ 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 $y_1, y_2, g_1, y_3, g_2$ are elements of some groups $G_1 = \langle g_1 \rangle = \langle h_1 \rangle$ and $G_2 = \langle g_2 \rangle = \langle h_2 \rangle$. Such proof of knowledge protocols can be turned into signature of knowledge using the Fiat-Shamir heuristic [15, 16]. We use the notation $\text{SPK}\{ (a): y = z^a \}(m)$ to denote a signature on a message $m$ obtained in this way.

**Definition 3** 
An $(i+1)$ tuple $(c, s_1, \ldots, s_t) \in \{0,1\}^i \times \mathbb{Z}_n^*$ satisfying the equation
\[
c = H(m \ || \ y \ || \ e \ || \ t_1 \ || \ \ldots \ || \ t_t)
\]
with $t_i = \begin{cases} g^{\langle c \rangle} & \text{if } e[i] = 0 \\ y^{\langle c \rangle} & \text{otherwise} \end{cases}$
is a signature of the knowledge of an $e$-th root of the discrete logarithm of $y$ to the base $g$, and is denoted $\text{SKROOTLOG}\{ \alpha : y = g^{\alpha} \}(m)$.

Such a signature can be computed if the $e$-th root $\alpha$ of the discrete logarithm of $y$ to the base $g$ is known. One first computes the values
for $i = 1, 2, \ldots, l$ with randomly chosen $r_i \in \mathbb{Z}_n^\ast$.

Then, $c$ is set to $H(m \| y \| g \| e \| t_1 \| \ldots \| t_l)$, and finally, $s_i = \begin{cases} r_i & \text{if } c[i] = 0 \\ r_i / x(\text{mod } n) & \text{otherwise} \end{cases}$ for $i = 1, 2, \ldots, l$. It can easily be seen that the resulting tuple $(c, s_1, \ldots, s_l)$ satisfies the verification equation.

**Definition 4** A signature of the knowledge of representations of $y_1, y_2, \ldots, y_w$ with respect to the bases $g_1, g_2, \ldots, g_v$ on the message $m$ is denoted as follows

$$SKREP\left(\{\alpha_1, \ldots, \alpha_u\} : \left(\begin{array}{c}
\left(\begin{array}{c}
y_1 = \prod_{j=1}^{l} g_{b_{ij}}^{\alpha_{ij}} \\
y_{w} = \prod_{j=1}^{l} g_{b_{ij}}^{\alpha_{wj}}
\end{array}\right)\right)\right)(m),$$

where the indices $e_j \in \{1, \ldots, u\}$ refer to the elements $\alpha_1, \ldots, \alpha_u$, and the indices $b_j \in \{1, \ldots, v\}$ refer to the base elements $g_1, \ldots, g_v$. The signature consists of an $(u+1)$ tuple $(c, s_1, \ldots, s_u) \in \{0, 1\}^k \times \mathbb{Z}_q^u$ satisfying the equation

$$c = H(m \| y_1 \| \ldots \| y_w \| g_1 \| \ldots \| g_v \| \left\{\left(e_j, b_j\right)_{j=1}^{l}\right\}_{j=1}^{w} \left(\begin{array}{c}
y_1 \prod_{j=1}^{l} g_{b_{ij}}^{x_{ij}} \\
y_{w} \prod_{j=1}^{l} g_{b_{ij}}^{x_{wj}}
\end{array}\right)),$$

$$\| y_1 \prod_{j=1}^{l} g_{b_{ij}}^{x_{ij}} \| \ldots \| y_w \prod_{j=1}^{l} g_{b_{ij}}^{x_{wj}} \|,$$

and then sets $s_i = r_i - c \alpha_i (\text{mod } n)$ for $i = 1, \ldots, u$.

**Definition 5** An efficient signature of the knowledge of the $e$-th root of the $g$-part of a representation of $y$ to the bases $h$ and $g$, denoted $E-SKROOTREP(\{\alpha, \beta\} : y = h^a g^\beta)(m)$, consists of an $(e-1)$-tuple $(y_1, \ldots, y_{e-1}) \in G^{e-1}$ and of a signature of knowledge

$$U = SKREP\left(\{y_1, \ldots, y_{e-1}\} : y_1 = h^{i_1} g^\delta \land y_2 = h^{i_2} y_1^\delta \land \ldots \land y_{e-1} = h^{i_{e-1}} y_{e-2}^\delta \land y = h^{i_e} y_{e-2}^\delta\right)(m).$$

The signature of knowledge can be verified by checking the correctness of $U$.

### 2.3 Commitment Scheme

In [17], Pederson proposed a computationally binding and unconditionally hiding scheme based on the discrete logarithm problem. The commitment scheme is following: Given are a group $G$ of prime order $q$ and two random generator $g$ and $h$ such that $\log_g h$ is unknown and computing discrete logarithms is infeasible. A value $a \in \mathbb{Z}_q$ is committed to as $c_a := g^a h^r$, where $r$ is randomly chosen from $\mathbb{Z}_q$.

### 3 The Model of DAA Scheme

A forward secure DAA scheme has four types of participants and six procedures. The participants are the certificate issuer, a trusted platform module (TPM), a host that has TPM “built in”, and a verifier. The six procedures are described in the following:

- **Setup:** This probabilistic polynomial time (PPT) algorithm takes as input a security parameter and outputs system-wide public parameters and secret keys for group membership certificate generation.
- **Join:** This interactive protocol between a TPM and the issuer allows TPM to obtain a group membership certificate and become a group member. The public certificate and the TPM’s identity information are stored by the issuer in a database for future use.
- **Evolve:** This deterministic polynomial time (DPT) algorithm takes as input TPM’s private key for time period $i$ and outputs the corresponding private key for time period $i + 1$.
- **Sign:** This PPT algorithm takes as input group membership certificate, TPM’s private key and a message and returns an anonymous group signature of the message.
- **Verify:** This DPT algorithm takes as input the group public keys, a message and its candidate signature and returns either accept or reject. A signature is verified to make sure it originates from a legitimate TPM without knowledge of which particular one.
4 Forward Secure DAA Scheme

In this section, we describe our method for implementing direct anonymous attestation. As mentioned earlier, our construction is based on Ateniese et al.’s group signature scheme [11]. However, the sign and verify protocols are re-designed.

4.1 Setup
Let \( \varepsilon > 1, k \), and \( l_p \) be security parameters and let \( \lambda_1, \lambda_2, r_1, r_2 \) denote lengths satisfying \( \lambda_1 > \varepsilon (\lambda_2 + k) + 2 \), \( \lambda_2 > 4l_p \), \( \eta > \varepsilon (r_2 + k) + 2 \), and \( r_2 > \lambda_1 + 2 \). Let \( H \) be a collision-resistant hash function \( H : \{0,1\}^* \rightarrow \{0,1\}^k \).

The certificate issuer generates the group public key and his secret key as follows:

1) Choose random secret \( l_p \)-bit primes \( p', q' \), such that \( p = 2p' + 1 \) and \( q = 2q' + 1 \) are prime. Set the modulus \( n = pq \).
2) Choose random elements \( a, b, g, h \in QR(n) \) (of order \( p'q' \)). Compute \( \phi(n) = (p - 1)(q - 1) \).

Choose a random integer \( s \) such that \( \gcd(s, \phi(n)) = 1 \) (e.g. \( s = 3 \)). Divide the time during which the group public key is valid into \( T \) time periods and makes the time intervals public.

3) The group public key is \( Y = (n, a, b, g, h, s) \) and the corresponding secret key is \( S = (p', q') \).

In DAA scheme, it is infeasible to reveal real identity of signer even if for the issuer. Thus our DAA scheme does not need the revocation public key and corresponding secret key that are required in all group signature schemes.

4.2 Join Protocol
The join protocol is a protocol runs between the issuer and a TPM. We assume that the TPM and the issuer have established a private and authentic channel. The join protocol takes the following steps:

1) TPM chooses a random private key \( f_0 \in \{0, 2^{l_p}\} \) and keeps it secret.

2) TPM chooses a random secret \( t' \in \{0, 2^{l_p}\} \), computes \( C = a^{f_0} b^{t'} \bmod n \), and sends \( C \) to the issuer. Additionally, TPM proves to the issuer knowledge of \( f_0 \) and \( t' \), it executes as prover the protocol \( U = E - SKROOTREP\{(f_0, t') : C = a^{f_0} b^{t'} \bmod n\}(\cdot) \) with the verifier.

3) The issuer checks that \( C \in QR(n) \). If this is the case and \( U \) is correct, the issuer selects a random \( t'' \in \{0, 2^{l_p}\} \) and a random prime \( e \in \{2^n - 2^\eta, 2^n + 2^\eta\} \) and computes \( A = (Cb^{t''} d)^{t''} \bmod n \). Finally, the issuer sends \( (A, e, t'') \) to the host.

4) The host stores \( (A, e, t') \) and forwards \( (A, e, t') \) to the TPM. The TPM computes \( t = t' + t'' \), verifies that \( A^e = a^{f_0} b^{t'} \bmod n \) and stores \( t \).

As a result of the protocol, the TPM will have obtained secret values \( f_0 \) and \( t \), and the host will have values \( A \) and \( e \). Moreover, the TPM is
allowed to re-run the Join protocol with the same secret \( f_0 \) value many times.

### 4.3 Evolve

The TPM can evolve its private key using a public one-way function \( f(x) = x^r \mod n \). Assume the TPM has membership private key \((A,e,t,f_i)\) at time period \( i \). Then at time period \( i + 1 \), his private key becomes \((A,e,t,f_{i+1})\), where \( f_{i+1} = f_i^s \mod n \), and the verification equation is

\[
A' = a^{b^{f_i^s}} b^d \mod n .
\]

### 4.4 Sign

Let \( m \) be the message to be signed. The TPM has secret key \((f_i,t)\) and a credential \((A,e)\), whereas the host only knows the credential \((A,e)\). The signing algorithm takes the following steps:

1. The host picks random integer \( w \in_R \{0, 1\}^{2t} \) and computes
   \[
   T_1 = Ag^w \mod n , \quad T_2 = g^e h^w \mod n .
   \]
   The TPM computes \( T_3 = a^{f_i^{w^e}} \mod n \) and sends \( T_3 \) to the host.

2. The TPM and host together produce a signature of knowledge that \( T_1 \) and \( T_2 \) are commitments to the credential and \( T_3 \) is computed using the TPM’s private key \( f_i \). That is, they compute the signature of knowledge:

   \[
   SPK\{(e, f_i, t, w) : T_1^e g^{-cw} = T_3^e b^i \mod n \land \ T_2 = g^e h^w \mod n \land \ T_3 = a^{f_i^{w^e}} \mod n \land \ e \in (2^t - 2^t, 2^t + 2^t)\}(m)
   \]

   Actually, it equals two signatures of knowledge:

   \[
   V_1 = \text{SKROOTLOG}\{f_i : T_3 = a^{f_i^{w^e}} \mod n\}(m)
   \]

   Most of the secrets involved are actually known by the host; in fact only the values involving \( f_i \) and \( t \) need to be computed by the TPM, as the reader can see below.

a) The TPM picks random integer \( r_i \in \{0, 1\}^{e(\delta_e + k)} \) and computes \( \tilde{R}_i = b^{r_i} \mod n \). The TPM sends \( \tilde{R}_i \) to the host.

b) The host picks random integers \( r_e \in \{0, 1\}^{e(\delta_e + k)} \), \( r_{\delta_e} \in \{0, 1\}^{e(\delta_e + 2k)} \), and computes

   \[
   R_i = \tilde{R}_i T_1^{e v_{\delta_e}} \mod n , \quad R_2 = g^e h^{r_e} \mod n , \quad R_3 = T_2^{e v_{\delta_e}} h^{r_{\delta_e}} \mod n .
   \]

c) The host also computes \( c_h = H(g || h || n || s || T || a || b || d || T_1 || T_2 || T_3 || R_1 || R_2 || R_3) \) and sends \( c_h \) to the TPM.

d) The TPM chooses a random \( n_i \in \{0, 1\}^k \), computes \( c = H(H(c_h || n_i) || m) \) and sends \( c, n_i, s_i \) to the host.

e) The host computes \( s_c = r_e + c(e - 2^k) , \)

\[
\begin{align*}
  s_e &= r_n + cw , \\
  s_{\delta_e} &= r_{\delta_e} + c \cdot e \cdot e , \\
  s_{\delta_i} &= r_{\delta_i} + c \cdot e \cdot w .
\end{align*}
\]

The signature

\[
V_1 = (T_1, T_2, T_3, c, n_i, s_e, s_{\delta_e}, s_{\delta_i}, s_{\delta_i})
\]

f) The TPM computes the values

\[
t_j^* = a^{x_{j}^*} \mod n
given \( j = 1, \ldots, l \) with randomly chosen \( x_j \in \mathbb{Z}_n^* \) and security parameter \( l \leq k \). Then, \( c' \) is set to

\[
H\left(m || T_3 || a || s^{T_3} || t_1^* || \ldots || t_l^* \right) ,
\]

and finally,

\[
w_j = \begin{cases} 
  x_j & \text{if } c'[j] = 0 \\
  x_j / f_i(mod n) & \text{otherwise}
\end{cases}
\]

for \( j = 1, \ldots, l \). Then \( V_2 = (c', w_1, \ldots, w_l) \) and the host outputs the signature \( \sigma = (V_1, V_2) \).

### 4.5 Verify

A signature \( \sigma = (T_1, T_2, T_3, c, n_i, s_e, s_{\delta_e}, s_{\delta_i}, s_{\delta_i}, c', w_1, \ldots, w_l) \) on a message \( m \) w.r.t the public key \((n,a,b,d,g,h,s)\) is as verified as follows.

1. Compute

\[
R'_i = T_1^{e v_{\delta_e}} g^{-s_{\delta_i} b^{-r_i}(T_3)^{-c}} \mod n ,
\]
\[ R_2' = T_2^{c_5, c_6, h} \mod n, \]
\[ R'_1 = T_1^{c_1, c_2} h^{s_2} \mod n. \]

2) Verify that
\[ c = H(H(g_H || n || s || T || a || b || d || T_i || T_j || T_k || R_i' || R_j' || R_k' || n_i) || m), \]
\[ s_r \in \{0,1\}^{(j_1+k+1)}, \quad s_r \in \{0,1\}^{(j_1+k+1)}, \]
\[ s_r \in \{0,1\}^{(j_1+k+1)}, \quad s_r \in \{0,1\}^{(j_1+k+1)}, \]
\[ s_r \in \{0,1\}^{(j_1+k+1)}. \]

3) Compute
\[ t_i = \begin{cases} a_i & \text{if } c[T][i] = 0 \\ T_3^{s(i)} & \text{otherwise} \end{cases} \]
and verify \( c' = H(m || T_3 || a || s ^{-t_i} || t_1 || \cdots || t_l) \).

4) If all the above verifications succeed, the verifier outputs success, otherwise outputs fail.

### 4.6 Sign with Variable Anonymity

Let variable anonymity [12] is a conditionally linkable anonymous authentication, in which the signatures signed by the same TPM in a certain time interval are linkable. To achieve variable anonymity, each signature will belong to a “linkability class” that is identified using a Solely Signature Identifier (SSID). All signatures made by the same TPM with the same SSID are linkable. If complete anonymity is desired, the signer can simply pick a random SSID.

To implement variable anonymity, we add the following computations to the Sign protocol:

\[ \eta = H_I(SSID), \quad T_3 = \eta^h \mod n, \]
\[ V_2 = SPK \{ f_i : T_3 = a^{r_i} \mod n \wedge T_4 = \eta^{s_i} \mod n \} (m) \]

where \( H_I : \{0,1\} \rightarrow Z_n^* \), and outputs the signature \( \sigma = (\eta, V_1, V_2) \). The computation of \( V_2 \) can be found in the Appendix A.

### 4.7 Rogue TPM Tagging

If TPM’s private key \( f_i \) at time period \( i \) is compromised, a verifier should be able to identify the revocation list of rogue TPMs. To do so, the secret of a corrupted TPM should be published on the revocation list. For each private key \( f_i \), on the revocation list, a verifier checks \( T_4 = \eta^e \mod n \). If the equation holds, the request comes from a revoked TPM.

### 5 Analysis of Proposed DAA Scheme

In this section, we analyze the security and performance of our proposed DAA scheme.

#### 5.1 Security Analysis

**Lemma 6** Let \( n \) be a special RSA number, \( u, g \in QR(n) \) and \( g \) be a generator of \( QR(n) \) . Under the strong RSA assumption, if \( g^x \equiv u^y \mod n \) for given \( x, y \in Z_n^* \), then
\[ \gcd(x, y) = 1. \]

**Proof:** Let \( r = \gcd(x, y) \), by the extended Euclidean algorithm, there exist \( \alpha, \beta \) s.t. \( \alpha x + \beta y = r \). Hence,
\[ g \equiv g^{(\alpha x + \beta y)/r} \equiv (u^\alpha g^\beta)^{y/r} \mod n \]

Note that we cannot have \( y > r \) because otherwise \( u^\alpha g^\beta \) is a \( y/r \)-th root of \( g \), which contradicts the strong RSA assumption. Thus, we have \( y = r \), i.e. \( \gcd(x, y) = y \).

**Theorem 7** Under the strong RSA assumption, the interactive protocol underlying the Sign and Verify protocol is a statistical zero-knowledge (honest-verifier) proof of knowledge of a membership credential \( (A, e) \) and the corresponding secret key \( (f_i, e) \).

**Proof:** The proof that the interactive protocol is statistical zero-knowledge is quite standard. We restrict our attention the proof of knowledge part.

Now we show that the knowledge extractor is able to recover the membership credential and the corresponding secret key once it has found two accepting tuples. Let \( (V_1' = (T_1, T_2, T_3, R_1, R_2, R_3, n_i, c', s_1', s_2', s_3', s_4') V_2) \) and \( (V_2' = (T_1, T_2, T_3, R_1, R_2, R_3, n_i, c', s_1', s_2', s_3', s_4') V_2) \) be two accepting tuples. Then we have
\[ R_1 = T_1^{c_1, c_2, c_3} g^{s_1} b^{s_2} \mod n \]
\[ R_2 = T_2^{c_4} g^{s_5} h^{s_6} \mod n \]

\[ R_2' = T_2^{c_1, c_2, c_3} g^{s_1} b^{s_2} \mod n \]
\[ R_2'' = T_2^{c_4} g^{s_5} h^{s_6} \mod n \]
Therefore under the DDH assumption, it is infeasible to decide whether or not there exists an \( f_j \) such that \( T_4 = \eta^{f_j} \mod n \) and \( T'_4 = \eta^{f_j} \mod n \).

**Theorem 8** Under the decisional Diffie-Hellman assumption over subset of \( QR_n \), it is infeasible to link the transactions by a TPM with different SSID.

**Proof.** To decide whether two transactions are linked to a TPM, one needs to decide whether two signatures \( \sigma = (\eta, V_1, V_2) \) and \( \sigma' = (\eta', V_1', V_2') \) are produced from the same TPM.

Since the interactive protocol is statistically zero knowledge, no information is statistically revealed by \( \{V_1, V_2\} \) in the random oracle. For the different SSID, we have \( \eta \neq \eta' \). Therefore, under the DDH assumption, it is infeasible to decide whether or not there exists an \( f_j \) such that \( T_4 = \eta^{f_j} \mod n \) and \( T'_4 = \eta^{f_j} \mod n \).

### 5.2 Performance Analysis

Suppose we set the modulus \( n \) be about 2048 bits. We further choose the security parameters \( k = 160, \ v = 4/3, \ s = 2, \ T = 3, \ \lambda = 1024 \). For the signature with total anonymity, the total bit-length is 44128. To generate a DAA signature, the host needs to perform 10 modular exponentiations and the TPM 16 modular exponentiations. To verify a signature, 26 modular exponentiations is needed. Moreover, the key evolve procedure is finished by the TPM and no interactive protocol transmitted in the network is needed. In other words, our key-evolve features self-healing.

**Corollary 9** The DAA scheme presented in Section 4 is secure under the strong RSA assumption and the decisional Diffie-Hellman assumption in the random oracle model.

**Proof.** We have to show that our scheme satisfies all the security properties listed in Section 3.

**Correctness:** By inspection.

**Unforgeability:** It can be easily proven from the Theorem 7.

**Unlinkability:** It can be easily proven from the Theorem 8.

**Anonymity:** If the SSID is chosen randomly, it is infeasible to identify the actual signer for a valid signature \( \sigma = (\eta, V_1, V_2) \). The reason is that the underlying interactive protocol is statistically zero-knowledge, and no information is statistically revealed in the random oracle model.

**Forward Security:** The key evolve function is \( f(x) = x^e \mod n \). As only the issuer knows \( p, q \) such that \( n = pq \), even if the TPM is compromised and the private key \( f_j \) in time period \( i \) is known by an attacker, he still cannot compute the past private key \( f_j (j < i) \). Therefore, the signature produced in previous time periods is still anonymous and unlinkable. Moreover, the key evolve procedure is finished by the TPM and no interactive protocol transmitted in the network is needed. In other words, our key-evolve features self-healing.

**6 Conclusion**
In this paper, we present the first forward secure DAA scheme. We believe that forward security provides really useful features of direct anonymous attestation mechanism. Our scheme satisfies forward security as well as all the traditional security properties shared with previous DAA schemes. However, the efficiency of our scheme should be improved and it is left as our future work.

Acknowledgements: The research was supported by the National Grand Fundamental Research (973) Program of China under Grant No. 2007CB311202 and the National Natural Science Foundation of China (NSFC) under Grants No.60673083 and No.60873197.

References:

Appendix A: The Signature of Knowledge

\[ V_2 = SPK \{ f_j : T_3 = a^{r_j} \mod n \land T_4 = \eta^r \mod n_1 \} (m) \]

Such a signature can be computed if \( f_j \) is known. One first computes the values

\[ t_j^* = a^{J^i}, \quad T_j^* = \eta^r \]

for \( j = 1, \ldots, l \) with randomly chosen \( r_j \in Z_n^* \) and security parameter \( l \leq k \). Then, \( c \) is set to

\[
H(m || T_3 || T_4 || \eta || a || s^{T_3} || t_1^* || \cdots || t_l^* || T_1^* || \cdots || T_l^*)
\]

and finally,

\[
s_j = \begin{cases} r_j & \text{if } c[j] = 0 \\ r_j \mod (m) & \text{otherwise} \end{cases}
\]

for \( j = 1, \ldots, l \). The signature is \( (c, s_1, \ldots, s_l) \).
Given the signature \((c,s_1,\ldots,s_f)\), the verification procedure is as follows: One first computes
\[
\begin{align*}
t_j &= \begin{cases} 
a_i^{r_j} & \text{if } c[j] = 0 \\
\eta_j & \text{otherwise}
\end{cases} \\
\overline{t}_j &= \begin{cases} 
\eta_j & \text{if } c[j] = 0 \\
T_j^{\eta_j} & \text{otherwise}
\end{cases}
\end{align*}
\]
and verify
\[
c = \frac{1}{2} \log_2 \left( \frac{1}{n} \right) T_3 \| T_4 \| \eta \| a \| s^{T_j} \| t_1 \| \cdots \| t_j \| \bar{t}_j \| \cdots \| \bar{t}_j \).
\]

**Appendix B: Direct Anonymous Attestation Scheme without Forward Security**

**B.1 Setup**

Let \(e>1, k,l_p,\lambda_1,\lambda_2,\eta,\nu\) be security parameters and let \(H\) be a collision-resistant hash function \(H:\{0,1\}^k \rightarrow \{0,1\}^k\). The certificate issuer generates the group public key and his secret key as follows:

1. Choose random secret \(l_p\)-bit primes \(p, q\), such that \(p=2p+1\) and \(q=2q+1\) are prime. Set the modulus \(n=pq\).
2. Choose random elements \(a,b,g,h\in QR(n)\) (of order \(p\), \(q\)).
3. The group public key is \(Y=(n,a,b,g,h)\) and the corresponding secret key is \(S=(p,q)\).

**B.2 Setup**

1. The host picks random integer \(f\in\{0,1\}^k\) and keeps it secret.
2. The TPM chooses a random secret \(t'\in\{0,2^{\frac{k}{2}}\}\), computes \(C=a^f b^{t'} \mod n\), and sends \(C\) to the issuer. Additionally, TPM proves to the issuer knowledge of \(f\) and \(t'\), it executes as prover the protocol \(SPK((f,t')):\ C=a^f b^{t'} \mod n\) with the verifier.
3. The issuer checks that \(C\in QR(n)\). If this is the case and \(U\) is correct, the issuer selects a random \(t''\in\{0,2^{\frac{k}{2}}\}\) and a random prime \(e=2^k-2,2^{k}+2\) and computes \(A=(C b^e)^{\frac{1}{\gamma}} \mod n\). Finally, the issuer sends \((A,e,t'')\) to the host.

4. The host stores \((A,e)\) and forwards \((A,e,t'')\) to the TPM. The TPM computes \(t=t'+t''\), verifies that
\[
A^e = a^{t'} b^t \mod n
\]
and stores \(t\).

**B.3 Sign**

Let \(m\) be the message to be signed. The TPM has secret key \((f,t)\) and a credential \((A,e)\), whereas the host only knows the credential \((A,e)\). The signing algorithm takes the following steps:

1. The host picks random integer \(w\in\{0,1\}^2\) and computes \(T_1 = Ag^w \mod n\), \(T_2 = g^e h^w \mod n\).
2. The TPM and host together produce a signature of knowledge. That is, they compute the signature of knowledge
\[
SPK((e,f,t,w)): T_1^e g^{-ew} = a^t b^t \mod n \land T_2^e g^{ew} h^{ew} = 1 \mod n\}
\]

a) The TPM picks random integer \(r_1 \in \{0,1\}^{(\ell_1)+k}\) and computes \(R_1 = b^{-r_1} \mod n\). The TPM sends \(R_1\) to the host.

b) The host picks random integers \(r_1 \in \{0,1\}^{(\ell_1)+k}\), \(r_2 \in \{0,1\}^{(\ell_2)+k}\), \(r_3 \in \{0,1\}^{(\ell_3)+k}\), and computes \(R_1 = b^{-r_1} g^{\ell_2} \mod n\), \(R_2 = g^{r_2} h^{\ell_2} \mod n\).

c) The host also computes \(c_h = H(g \| h \| a \| b \| T_2 \| R_1 \| R_2 \| R_3)\) and sends \(c_h\) to the TPM.

d) The TPM chooses a random \(n_1 \in \{0,1\}^{k}\), computes \(c = H(\ell_1, n_1) \mod m\) and sends \(c, n_1, s_1, s_1\) to the host.

e) The host computes \(s_1 = r_1^c\), \(s_2 = r_2 + c t\), and sends \(s_1, s_2, s_3\) to the TPM.

The signature is
\[ \sigma = (T_1, T_2, c, n_1, s_e, s_f, s_w, s_\delta, s_\delta_2). \]

**B.4 Verify**

A signature \( \sigma = (T_1, T_2, c, n_1, s_e, s_f, s_w, s_\delta, s_\delta_2) \) on a message \( m \) w.r.t. the public key \((n,a,b,g,h)\) is as verified as follows.

1. Compute
   \[ R_1 = T_1^{s_e + c^2} g^{-s_\delta} b^{-s_f} (T_3)^{-c} \mod n, \]
   \[ R_2 = T_1^{-c} g^{s_e + c^2} h^{s_f} \mod n, \]
   \[ R_3 = T_2^{-s_e} g^{s_\delta} h^{s_\delta_2} \mod n. \]

2. Verify that
   \[ c = H(H(g \| h \| n \| a \| b \| T_1 \| T_2 \| R_1 \| R_2 \| R_3 \| n_1 \| m) \]
   \[ s_e \in \{0,1\}^{(\delta_\gamma + k) + 1}, \quad s_f \in \{0,1\}^{(\delta_\gamma + k) + 1}, \]
   \[ s_w \in \{0,1\}^{(2k + 1) + 1}, \quad s_\delta \in \{0,1\}^{(2\gamma + k + 1) + 1}, \]
   \[ s_\delta_2 \in \{0,1\}^{(\gamma + 2k + 1) + 1}. \]