A Direct Anonymous Attestation Scheme for Embedded Devices

He Ge[1] * and Stephen R. Tate[2]

[1] Microsoft Corporation, One Microsoft Way, Redmond 98005
hege@microsoft.com
[2] Department of Computer Science and Engineering
University of North Texas, Denton, TX 76203
srt@cse.unt.edu

Abstract. Direct anonymous attestation (DAA) is an anonymous authentication scheme adopted by the Trusted Computing Group in its specifications for trusted computing platforms. This paper presents an efficient construction that implements all anonymous authentication features specified in DAA, including authentication with total anonymity, authentication with variable anonymity, and rogue TPM tagging. The current DAA construction is mainly targeted for powerful devices such as personal computers, and their corresponding application areas, but is not entirely suitable for embedded devices with limited computing capabilities (e.g., cell phones or hand-held PDAs). We propose a new construction with more efficient sign and verify protocols, making it more attractive for embedded devices. We prove that the new construction is secure under the strong RSA assumption and the decisional Diffie-Hellman assumption.

Keywords: Direct Anonymous Attestation, Group signature, Privacy, Authentication, Trusted Computing Platform, Cryptographic Protocol.

1 Introduction

In this paper, we present an efficient direct anonymous attestation scheme for embedded devices. DAA is a group signature variant designed to protect the privacy of the owner of a trust computing platform, and has been adopted by the Trusted Computing Group, an industry consortium developing standards for "trusted computing platforms." A group signature is a privacy-preserving signature scheme introduced by Chaum and Heyst [12]. In such a scheme, there are two basic types of entities: a group manager and certain number of group members. The group manager issues a group membership certificate/credential for each group member. Later, based on its own group membership certificate, a group member can sign a message on behalf of the group without revealing its identity. That is, a third party can only verify that the signature was produced

* Work done while at the Department of Computer Science and Engineering, University of North Texas.

by a legitimate group member without being able to find which particular one. Only the group manager is able to open a signature and reveal its originator (in some cases this ability is held by a separate party known as an "open authority"). In addition, signatures signed by the same group member cannot be identified as from the same source, i.e., "linked." Recently, the study of group signature schemes has attracted considerable attention, and many solutions have been proposed in the literature (e.g., [1, 4, 5, 7–9]).

1.1 Background

The Trusted Computing Group [21] (TCG) is an industry consortium formed to develop standards for Trusted Computing Platforms. 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. Based on the TPM, a trusted computing platform can implement many security related features, such as secure boot, sealed storage, and software integrity attestation. More information about TPMs and trusted computing platforms can be found at the TCG website [21].

TPMs are tamper-resistant cryptographic chips. When a TPM is manufactured, a unique RSA keypair, called the Endorsement Key (EK), is created and stored in the protected area of the TPM. The EK might be generated inside a TPM, or imported from an outside key generator. The public part of the EK is authenticated by the manufacturer, while the private part of the EK will never be revealed to the outside. A TPM independently performs cryptographic computations inside itself, and even its manufacturer cannot obtain knowledge of these computations. TPMs are embedded into computing devices by a device manufacturer, and these devices are called trusted computing platforms when coupled with appropriate software. At the heart of trusted computing platform is the assumption that TPMs should independently work as expected, and be "trusted" by remote parties. Essentially, trusted computing platforms are based on trust of TPMs.

The deployment and use of TPMs introduces privacy concerns. If the authentication of a TPM is directly based on its EK, all transactions by the same TPM can be linked through the public part of the EK. Furthermore, if the TPM is associated with a user's identity, the user may suffer a loss of privacy. To protect the privacy of a TPM owner, two solutions have been proposed in the TPM specifications.

Privacy in the TPM v1.1 specification is based on a trusted third party, called a Privacy CA. A TPM generates a second RSA keypair called an Attestation Identity Key (AIK). The TPM sends an AIK to the Privacy CA, applying for a certificate on the AIK. After the TPM proves its ownership using a valid EK, the Privacy CA issues a certificate for this AIK. Later, the TPM sends the certificate for this AIK to a verifier, and proves it owns this AIK. This way, the TPM hides its identity during the transaction. Obviously, this is not a completely satisfactory solution, since each AIK creation needs the involvement

of the Privacy CA, and compromise of the Privacy CA (or a dishonest Privacy CA) can destroy all privacy guarantees.

An alternate solution added in TPM v1.2 is called Direct Anonymous Attestation (DAA), adopting techniques from group signatures: A TPM applies for a credential from an issuer, and later the TPM generates a special signature using this credential. A remote verifier can verify the signature has been constructed from a valid credential without the ability to recover the underlying credential. Different signatures based on the same credential might be linkable or unlinkable depending on a verifier's requirements. If the method implements unlinkable authentication, it is called total anonymity. It should be noted that the open operation defined in standard group signature schemes, which allows the group manager to learn the creator of a signature, is not included in DAA for privacy protection.

Variable anonymity [22] is a conditionally linkable anonymous authentication, in which the signatures signed by the same TPM in a certain time interval are linkable. However, when the signing parameters change, the signatures across the different periods cannot be linked. When the time interval becomes short, the method works like perfectly unlinkable authentication. When the period never expires, this leads to pseudo-anonymity. A verifier can adjust the time interval to detect suspicious attestation. If too many attestation requests come from the same TPM in a period of time, it is likely this TPM has been compromised.

Rogue TPM tagging is about the revocation of the key of a corrupted TPM. When a broken TPM is discovered and identified by its EK, its secrets will be published on the revocation list. A verifier can identify and exclude any rogue TPM on the list, and an issuer can refuse to issue new credentials to a TPM with a revoked EK.

The current solution for DAA is due to Brickell, Camenisch, and Chen [6], which we refer to as the BCC scheme in this paper. The BCC scheme is designed mainly for devices with powerful computing capabilities such as personal computers. The scheme is quite complex with high computing overhead. To expedite the authentication process, the computation has been distributed between a TPM and the host into which the TPM is embedded. The TPM finishes the computation related to the signature generation, while the host finishes the computation related to anonymity. The BCC scheme works fine with personal computers. However, it would be an expensive solution for devices with low computing capabilities, such as cell phones, hand-held PDA, etc.

1.2 Our Results

In this paper, we propose a new construction that can carry out all required features in DAA (total anonymity, variable anonymity, and rouge TPM tagging), and has much more efficient sign and verify protocols.

Our construction is built up from the group signature scheme due to Camenisch and Michels [8], which we will refer to as the CM scheme. We directly adopt their join protocol. However, our sign and verify protocols are totally different. We have devised an efficient way to carry out anonymous authentication

with much less computation. So far we are not aware of any similar method being adopted in other cryptographic constructions for anonymous authentication. Due to the simplicity and efficiency of our method, the new construction is more appealing for embedded devices with low computing capability. We will demonstrate this point in Section 4.6 when we present a performance analysis.

However, we also need to point out that the join protocol, which we directly adopt from the CM scheme, is not an efficient one. Furthermore, the security argument for the join protocol assumes a static adversary, while the counterpart in the BCC scheme can be proved secure under an adaptive adversary. However, we consider this to be a minor issue in real applications. The join protocol is the way a TPM obtains its anonymous certificate/credential. In practice, the join protocol normally is conducted in the system setup stage, and is run infrequently in later phases. Meanwhile, the join protocol generally should be completed in more strict environments with rigorous security requirements, so security under static attack should be reasonable and acceptable. Furthermore, the join protocol may not be the only option for certificate generation. In some applications, certificates could be produced at manufacturing time, just as the endorsement key (EK) is. In such a situation, the join protocol might not even be necessary.

The rest of this paper is organized as follows. The next section introduces the model for our construction. Section 3 reviews some definitions, cryptographic assumptions, and building blocks of our proposed scheme. Section 4 presents the proposed scheme. Security properties are considered in Section 5. Finally, we summarize and give conclusions in Section 6.

2 The Model

This section introduces the model for direct anonymous attestation, which is a variant of the group signature model [1]. Both these two models support procedures Setup, Join, Sign, and Verify, while DAA further supports mechanism such as variable linkability and rogue group member identification, i.e., rogue TPM tagging.

Definition 1. *Direct anonymous attestation is a digital signature scheme with two types of participants: the certificate issuer, and TPMs. It consists of the following procedures:*

- **Setup***: For a given security parameter σ, the issuer produces system-wide public parameters and a group master key for group membership certificate generation.*
- **Join***: An interactive protocol between a TPM and the issuer. The TPM obtains a group membership certificate to become a group member. The public certificate and the TPM's identity information are stored by the issuer in a database for future use.*
- **Sign***: Using its group membership certificate and private key, the TPM creates an anonymous group signature for a message.*

- **Verify**: *A signature is verified to make sure it originates from a legitimate TPM without knowledge of which particular one.*
- **Rogue tagging**: *A rogue TPM can be identified and excluded for the group.*

Similar to a group signature, DAA should satisfy the following properties:

- **Correctness**: Any valid signature can be correctly verified by the Verify protocol.
- **Forgery-Resistance**: A valid group membership certificate can only be created by a TPM and the issuer through the Join protocol.
- **Anonymity**: It is infeasible to identify the real TPM of a signature unless this TPM is on the revocation list.
- **Unlinkability**: It is infeasible to link two different signatures of the same TPM.
- **Non-framing**: No one (including the issuer) can sign a message in such a way that it appears to come from another TPM.

3 Definitions and Preliminaries

This section reviews some definitions, widely accepted complexity assumptions, and building blocks that we will use in this paper.

3.1 Number-Theoretic Assumption

Definition 2 (Special RSA Modulus). *An RSA modulus $n = pq$ is called special if $p = 2p' + 1$ and $q = 2q' + 1$ where p' and q' also are prime numbers.*

Definition 3 (Quadratic Residue Group QR_n). *Let Z_n^* be the multiplicative group modulo n, which contains all positive integers less than n and relatively prime to n. An element $x \in Z_n^*$ is called a* quadratic residue *if there exists an $a \in Z_n^*$ such that $a^2 \equiv x \pmod{n}$. The set of all quadratic residues of Z_n^* forms a cyclic subgroup of Z_n^*, which we denote by QR_n. If n is the product of two distinct primes, then $|QR_n| = \frac{1}{4}|Z_n^*|$.*

We list two properties about QR_n which will be be used in the later security proof.

Property 1 *If n is a special RSA modulus, with p, q, p', and q' as in Definition 2 above, then $|QR_n| = p'q'$ and $(p' - 1)(q' - 1)$ elements of QR_n are generators of QR_n.*

Property 2 *If g is a generator of QR_n, then $g^a \bmod n$ is a generator of QR_n if and only if $GCD(a, |QR_n|) = 1$.*

The security of our techniques relies on the following security assumptions which are widely accepted in the cryptography literature. (see, for example, [1, 2, 9, 10, 15]).

Assumption 1 (Strong RSA Assumption) *Let n be an RSA modulus. The* Flexible RSA Problem *is the problem of taking a random element $u \in Z_n^*$ and finding a pair (v, e) such that $e > 1$ and $v^e = u \pmod{n}$. The* Strong RSA Assumption *says that no probabilistic polynomial time algorithm can solve the flexible RSA problem with non-negligible probability.*

Assumption 2 (Decisional Diffie-Hellman Assumption for QR_n) *Let n be a special RSA modulus, and let g be a generator of QR_n. For two distributions (g, g^x, g^y, g^{xy}), (g, g^x, g^y, g^z), $x, y, x \in_R Z_n$, there is no probabilistic polynomial-time algorithm that distinguishes them with non-negligible probability.*

Kiayias et al. have investigated the Decisional Diffie-Hellman Assumption over a subset of QR_n in [17], i.e., x, y, z are randomly chosen from some subsets, truncation of QR_n. They showed that the Decisional Diffie-Hellman Assumption is still attainable over subsets of QR_n with the size down to at least $|QR_n|^{1/4}$.

3.2 Building Blocks

Our main building blocks are *statistical honest-verifier zero knowledge proofs of knowledge* related to discrete logarithms over QR_n [10, 11, 16]. They include protocols for things such as knowledge of a discrete logarithm, knowledge of the equality of two discrete logarithms, and knowledge of a discrete logarithm that lies in certain interval, etc. We introduce one of them here. Readers may refer to the original papers for more details.

Definition 4 (Protocol 1). *Let n be a special RSA modulus, QR_n be the quadratic residue group modulo n, and g be a generator of QR_n. Let α, l, and l_c be security parameters that are all greater than 1, and let X be a constant number. In the following protocol, Alice knows x, the discrete logarithm of T_1 (so $g^x \equiv T_1 \pmod{n}$), where $x \in [X - 2^l, X + 2^l]$. After the protocol is executed, Bob is convinced that Alice knows the discrete log x of T_1 such that $x \in [X - 2^{\alpha(l+l_c)+1}, X + 2^{\alpha(l+l_c)+1}]$.*

1. *Alice picks a random $t \in \pm\{0,1\}^{\alpha(l+l_c)}$ and computes $T_2 = g^t \pmod{n}$. Alice sends (T_1, T_2) to a verifier Bob.*
2. *Bob picks a random $c \in \{0,1\}^{l_c}$ and sends it to Alice.*
3. *Alice computes*
$$w = t - c(x - X),$$
which she sends to Bob. Notice that an honest Alice knows a value of $x \in [X - 2^l, X + 2^l]$, so given the range in which t and c were selected, an honest Alice will produce a w that satisfies $w \in [-2^{\alpha(l+l_c)+1}, 2^{\alpha(l+l_c)+1}]$ (actually in a slightly smaller interval than this, but this is a sufficiently tight bound for our purposes).
4. *Bob checks that $w \in [-2^{\alpha(l+l_c)+1}, 2^{\alpha(l+l_c)+1}]$ and*
$$g^{w-cX} T_1^c \equiv T_2 \pmod{n}.$$
If both tests pass, then Bob is convinced that Alice knows the discrete logarithm of T_1 and that it lies in the range $[X - 2^{\alpha(l+l_c)+1}, X + 2^{\alpha(l+l_c)+1}]$.

Remark 1: The parameter $\alpha > 1$ is used since we do not know the size of the group QR_n, and determines the statistical closeness of our actual distribution to the ideal one. In other words, α determines the statistical zero-knowledge property of this protocol. For a more in-depth discussion and analysis, we refer the reader to [8].

Remark 2: Using the Fiat-Shamir heuristic [14], the protocol can be turned into a non-interactive "signature of knowledge," which is secure in the random oracle model [3]. We will introduce our new signature scheme in the manner of a "signature of knowledge" in the next section.

4 The Direct Anonymous Attestation Scheme

In this section, we describe our method for implementing direct anonymous attestation. As mentioned earlier, our construction is based on the same group certificate as the CM scheme [8]. However, the sign and verify protocols are re-designed.

4.1 System Parameter Setting

The certificate issuer picks a security parameter σ, and generates the system parameters as follows:

- n, g: n is a special RSA modulus such that $n = pq$, where p and q are each at least σ bits long (so $p, q > 2^\sigma$), and $p = 2p' + 1$, and $q = 2q' + 1$, with p' and q' both being prime. g is a random generator of the cyclic group QR_n. n, g are public values while p and q are kept secret by the group manager.
- α, l_c, l_s, l_b: Security parameters that are greater than 1.
- X, Y: constant integers. $Y > 2^{\alpha(l_c+l_b)+1}$, and $X > 2Y + 2^{\alpha(l_s+l_c)+2}$.
- Two strong collision-resistant hash functions: $\mathcal{H}_1 : \{0,1\}^* \rightarrow Z_n^*$, and $\mathcal{H}_2 : \{0,1\}^* \rightarrow \{0,1\}^{l_c}$.

An illustration of the system parameters is the setting of $\sigma = 1024$ (so n is 2048 bits), $\alpha = 9/8$, $X = 2^{792}$ (99 bytes), $Y = 2^{520}$ (65 bytes), $l_s = 540$, $l_b = 300$, and $l_c = 160$.

4.2 Join Protocol

We adopt the same join protocol as in the CM group signature. A TPM obtains its group membership certificate as a keypair (E, s), such that s is prime, $s \in (X, X + 2^{l_s})$, and

$$E^s \equiv g \,(\mathrm{mod}\, n).$$

s is the TPM's private key and is kept secret by the TPM. For further details on how the join protocol works, see [8].

4.3 Authentication with Total Anonymity

The idea of our method for implementing authentication with total anonymity is as follows: the TPM picks a random blinding integer $b < s$, computes $T_1 = E^b = g^{s^{-1}b} \pmod{n}$, $T_2 = g^b \pmod{n}$. Then the TPM sends (T_1, T_2) to a verifier along with a proof that (T_1, T_2) is constructed from a legitimate keypair. Thus, a TPM's keypair is covered by this blinding integer b. The requirement for $b < s$ is important, which will be seen more clearly in the later security proof.

This method is very different from the one used in many group signature schemes (e.g., [1, 6, 8]). In those schemes, a group member basically adopts the ElGamal encryption to hide its identity [13]. For instance, in the CM group signature, a group member hide itself by computing

$$T_1 = Ey^b \pmod{n},\ T_2 = g^b \pmod{n},$$

where y is the group manager's public key. Afterwards, the task for the group member is to prove that (T_1, T_2) was constructed from a legitimate keypair, which is much less efficient than our method.

Now, we introduce our sign protocol. For a message m, the TPM executes the following steps to complete the sign protocol:

1. Generate a random $b \in_R [Y - 2^{l_b}, Y + 2^{l_b}]$, $t_1 \in_R \pm\{0,1\}^{\alpha(l_s+l_c)}$, $t_2 \in_R \pm\{0,1\}^{\alpha(l_b+l_c)}$, and compute

   $$T_1 = E^b \ (\bmod\ n),\ T_2 = g^b \ (\bmod\ n);\ d_1 = T_1^{t_1} \ (\bmod\ n),\ d_2 = g^{t_2} \ (\bmod\ n);.$$

2. Compute:

   $$c = \mathcal{H}_2(g||T_1||T_2||d_1||d_2||m);$$
   $$w_1 = t_1 - c(s - X),\ w_2 = t_2 - c(b - Y).$$

3. Output (c, w_1, w_2, T_1, T_2).

To verify a signature, the verifier computes

$$c' = \mathcal{H}_2(g||T_1||T_2||T_1^{w_1 - cX} T_2^c || g^{w_2 - cY} T_2^c || m),$$

and accepts the signature if and only if $c = c'$, $w_1 \in \pm\{0,1\}^{\alpha(l_s+l_c)+1}$, and $w_2 \in \pm\{0,1\}^{\alpha(l_b+l_c)+1}$.

4.4 Authentication with Variable Anonymity

To achieve variable anonymity, each signature will belong to a "linkability class" that is identified using a "linkability class identifier," or LCID. All signatures made by the same TPM with the same LCID are linkable, and in an interactive authentication protocol the LCID can be negotiated and determined by the TPM and verifier. For example, to link authentications to a single server over a single day, the LCID could simply be the server name concatenated with the date. If the same LCID is always used with a particular server (e.g., the server name),

then the result is a pseudo-anonymity system. If complete anonymity is desired, the signer can simply pick a random LCID (which is possible if the server is not concerned with linkability and allows arbitrary LCIDs).

The TPM derives a generator j of QR_n by hashing the LCID of this signature.

$$j = (\mathcal{H}_1(LCID))^2 \pmod{n}.$$

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

$$T_3 = j^s \pmod{n},\ d_3 = j^{t_1} \pmod{n},$$
$$c = \mathcal{H}_2(g||j||T_1||T_2||T_3||d_1||d_2||d_3||m);$$

and outputs $(c, w_1, w_2, T_1, T_2, T_3, m)$. The verifier then computes

$$c' = \mathcal{H}_2(g||j||T_1||T_2||T_3||T_1^{w_1-cX}T_2^c||g^{w_2-cY}T_2^c||j^{w_1-cX}T_3^c||m).$$

Since j will remain unchanged for a certain time interval, the same TPM will always produce the same T_3 during this interval. The frequency of T_3 will be used by the verifier to identify suspicious authentication, and may refuse to provide further services. Since j changes in different periods of time, this ensures the unlinkability of the same TPM between periods.

4.5 Rogue TPM Tagging

As described earlier, TPMs are manufactured to provide tamper-resistance. Otherwise, the basic benefits of trusted computing platforms would become meaningless. However, in extreme circumstances, a TPM may be compromised and its keypair exposed, so a verifier should be able to identify the attestation request from rogue TPMs. To do so, the secrets of a corrupted TPM (e.g., EK, E, and s) should be published on the revocation list. For a keypair (E, s) on the revocation list, a verifier checks

$$T_1^s =? T_2 \pmod{n}.$$

If the equation holds, the request comes from a revoked TPM.

4.6 Performance Analysis

We present a performance analysis of our scheme in the section. It can be observed that the computation complexity in our scheme is dominated by the modular squaring and multiplication operations. To estimate the computation cost, it is sufficient to count total modular squarings and multiplications in the protocol. For simplicity, we estimate the computation cost based on techniques for general exponentiation [19]. For a particular exponentiation operation, let m_1 be the bit length of the exponent, and m_2 be the number of 1's in the binary representation. Then the total computation cost can be estimated as m_1 squarings and m_2 multiplications. For example, if $y = g^x \pmod{n}$, and $x \in_R \{0,1\}^{160}$,

then the expected number of 1's in x is 80, so the total expected computation includes 160 squarings and 80 multiplications.

Suppose we set $\sigma = 1024$, so n is 2048 bits (p, q are 1024 bits). We further choose $\alpha = 9/8$, $l_c = 160, l_s = 540, l_b = 300$. We also set $X = 2^{792}$ (99 bytes), $Y = 2^{520}$ (65 bytes). This parameter setting conforms to the requirements of the decisional Diffie-Hellman assumption over the subset of QR_n. We can observe that most bits of s, b are 0's. The computation with exponent b has 520 squarings and 151 expected multiplications. For authentication with total anonymity, a TPM needs 2352 ($520 \times 3 + 792$) squarings, and 958 ($151 \times 2 + 520/2 + 792/2$) multiplications.

We have counted the total exponent bit-length in the BCC scheme, which is 25844 for authentication with total anonymity. However, due to the computation distribution between the TPM and its host, efficient algorithm for mult-based exponentiation can be used on the host part (Algorithm 15.2 in [18]). According to our counting result, in the BCC scheme, the total exponent bit-length for the TPM is around 4088, and 12098 for the host. So the total exponent bit-length is 16186 ($4088 + 12098$), which includes 16186 squarings and 8093 expected multiplications. If we assume the cost of squaring is equal to that of multiplication (squaring can be at most two times faster than multiplication), our scheme is about 7 (24279/3310) times faster than the BCC scheme. Even if we only consider the computation inside the TPM, our scheme is almost 2 (6132/3310) times faster than the BCC scheme. For variable anonymity, our scheme needs 5561 modular multiplications, which still can be carried out by the TPM alone.

It should be noticed that the computation can also be distributed in our scheme. T_1, T_2, d_2, w_2 can be calculated by the host, and T_3, d_1, d_3, w_1 must be computed inside the TPM. Generally speaking, this should be unnecessary since all the computation can be done by the TPM alone.

Without the distribution of computation, the system design can be greatly simplified. Thus, our method is more appropriate for mobile devices with low computing capabilities.

5 Security Properties

We first propose a lemma that deals with the valid range of system parameters.

Lemma 1. *If* $X > 2^{\alpha(l_s+l_c)+2}$, $\alpha, l_s, l_c > 1$, *then* $(X - 2^{\alpha(l_s+l_c)+1})^2 > X + 2^{\alpha(l_s+l_c)+1}$.

Proof.

$$\begin{aligned}
&(X - 2^{\alpha(l_s+l_c)+1})^2 - (X + 2^{\alpha(l_s+l_c)+1}) \\
&\quad = X^2 - X2^{\alpha(l_s+l_c)+2} + 2^{2\alpha(l_s+l_c)+2} - X - 2^{\alpha(l_s+l_c)+1} \\
&\quad = X(X - 2^{\alpha(l_s+l_c)+2} - 1) + 2^{2\alpha(l_s+l_c)+2} - 2^{\alpha(l_s+l_c)+1}
\end{aligned}$$

Since $\alpha, l_s, l_c > 1$, and $X > 2^{\alpha(l_s+l_c)+2}$, the equation is greater than 0. □

Next we introduce a lemma due to Shamir [20].

Lemma 2. *Let n be an integer. Given values $u, v \in Z_n^*$ and $x, y \in Z_n$ such that $GCD(x, y) = r < x$, and $v^x \equiv u^y \pmod{n}$, there is an efficient way to compute a value z such that $z^k \equiv u \pmod{n}$, where $k = x/r$.*

Proof. Since $GCD(x, y) = r, r < x$, using the extended Euclidean GCD algorithm, we can obtain values α and β such that $\alpha x/r + \beta y/r = 1$. Then we have

$$u \equiv u^{\alpha x/r + \beta y/r} \equiv u^{\alpha x/r} u^{y\beta/r} \equiv u^{\alpha x/r} v^{\beta x/r} \equiv (u^\alpha v^\beta)^{x/r} \pmod{n}.$$

Therefore, setting $k = x/r$ and $z = u^\alpha v^\beta$, we have $z^k \equiv u \pmod{n}$. □

Based on this lemma, we can immediately obtain a corollary for later proof.

Corollary 1. *Let n be an integer. For given values $u, v \in Z_n^*$ and $x, y \in Z_n$ such that $x > y$ and $v^x = u^y \pmod{n}$, there is an efficient way to compute values (x, k) such that $x^k = u \pmod{n}$.*

Proof. Since $x > y$, we have $GCD(x, y) = r, 1 \leq r \leq y < x$. Due to Lemma 2, we can find a pair (x, k) such that

$$x^k \equiv u \ (mod\ n),$$

where $k = x/r$. Therefore $y \leq k \leq e$. □

Now, we start addressing the security of our scheme. We need to address the issue of keypair forgery in case an attacker can obtain a set of legitimate keypairs. A successful attack is one in which a new keypair is generated that is valid and different from current keypairs. The following theorem shows that, assuming the strong RSA assumption, it is intractable for an attacker to forge such a keypair. This analysis assumes a static adversary, not an adaptive adversary who can adaptively obtain polynomial amount of keypars at his own choice.

Theorem 1 (Forgery-resistance). *If there exists a probabilistic polynomial time algorithm which takes a list of valid keypairs, $(E_1, s_1), (E_2, s_2), \ldots, (E_k, s_k)$ and with non-negligible probability produces a new valid keypair (E, s) such that $E^s \equiv g \pmod{n}$ and $s \neq s_i$ for $1 \leq i \leq k$, then we can solve the flexible RSA problem with non-negligible probability.*

Proof. Suppose there exists a probabilistic polynomial-time algorithm which computes a new legitimate keypair based on the available keypairs, and succeeds with some non-negligible probability $p(\sigma)$. Then we construct an algorithm for solving the flexible RSA problem, given a random input (u, n), as follows (the following makes sense as long as u is a generator of QR_n, which is true with non-negligible probability for random instances — we consider this more carefully below when analyzing the success probability of our constructed algorithm):

1. First, we check if GCD$(u, n) = 1$. If it's not, then we have one of the factors of n, and can easily calculate a solution to the flexible RSA problem. Therefore, in the following we assume that GCD$(u, n) = 1$, so $u \in Z_n^*$.
2. We pick random prime numbers $s_1, s_2, \ldots, s_k$ in the required range $s \in [X - 2^{\alpha(l_s+l_c)+1}, X + 2^{\alpha(l_s+l_c)+1}]$, and compute

$$r = s_1 s_2 ... s_k,$$

$$g = u^r = u^{s_1 s_2 \cdots s_k} \pmod{n}.$$

   Note that since the s_i values are primes strictly less than either p' or q', it must be the case that GCD$(r, |QR_n|) = 1$, so Property 2 says that g is a generator of QR_n if and only u is a generator of QR_n.
3. Next, we create k group keypairs, using the s_i values and E_i values calculated as follows:

$$\begin{aligned} E_1 &= u^{s_2 \cdots s_k} \pmod{n} \\ E_2 &= u^{s_1 s_3 \cdots s_k} \pmod{n} \\ &\vdots \\ E_k &= u^{s_1 s_2 \cdots s_{k-1}} \pmod{n} \end{aligned}$$

   Note that for all $i = 1, \ldots, k$, raising E_i to the power s_i "completes the exponent" in a sense, giving $E_i^{s_i} = u^{s_1 s_2 \cdots s_k} = u^r = g \pmod{n}$.
4. We use the assumed forgery algorithm for creating a new valid keypair (E, s), where $s \in [X - 2^{\alpha(l_s+l_c)+1}, X + 2^{\alpha(l_s+l_c)+1}]$, and $E^s = g = u^r \pmod{n}$.
5. If the forgery algorithm succeeds, then s will be different from all the s_i's. By Lemma 1, s cannot be the product of s_i, s_j, $1 \leq i, j \leq k$. Therefore, either GCD$(s, s_1 s_2 \cdots s_k) = 1$, or $GCD(s, s_1 s_2 \cdots s_k) = s_i$, $1 \leq i \leq k$. In the first case, due to Lemma 2, we can find a pair (y, s) such that

$$y^s = u \pmod{n}$$

   so the pair (y, s) is a solution to our flexible RSA problem instance. In the second case, assume $s = v \times s_i$, then $v < X - 2^{\alpha(l_s+l_c)+1}$, and $GCD(v, s_1 s_2 \cdots s_k) = 1$ (or $GCD(v, r) = 1$). We then have

$$E^s \equiv E^{v s_i} \equiv u^r \pmod{n}.$$

   Again by Lemma 2, we can find a pair (y, v) such that

$$y^v = u \pmod{n},$$

   so the pair (y, v) is a solution to our flexible RSA problem instance.

We now analyze the probability that the above algorithm for solving the flexible RSA problem succeeds. The algorithm succeeds in Step 1 if GCD$(u, n) \neq 1$, so let P_1 represent the probability of this event, which is negligible. When GCD$(u, n) = 1$, the algorithm succeeds when the following three conditions are satisfied: (1)

$u \in QR_n$, which happens with probability $\frac{1}{4}$, (2) u is a generator of QR_n, which fails for only a negligible fraction of elements of QR_n, due to Property 1, and (3) the key forgery algorithm succeeds, which happens with probability $p(\sigma)$. Putting this together, the probability that the constructed algorithm succeeds is $P_1 + (1 - P_1)\frac{1}{4}(1 - \mathsf{negl}(\sigma))p(\sigma)$, which is non-negligible. □

In step 5 of the proof about forgery resistance (Theorem 1), we can obtain a corollary as follows.

Corollary 2. *Under the strong RSA assumption, it is intractable to forge a keypair* (E, s) *such that* s *lies in the interval* $(0, X - 2^{\alpha(l_s+l_c)+1})$ *or* $(X + 2^{\alpha(l_s+l_c)+1}, (X - 2^{\alpha(l_s+l_c)+1})^2)$*, and* $E^s = g \pmod{n}$*.*

Proof. In step 5 of the proof for Theorem 1, if $s \in (0, X - 2^{\alpha(l_s+l_c)+1})$, since all $s_i \in [X - 2^{\alpha(l_s+l_c)+1}, X + 2^{\alpha(l_s+l_c)+1}]$ are prime, then $GCD(s, s_1 s_2 \cdots s_k) = 1$, and we can solve a flexible RSA problem.

If $s \in (X + 2^{\alpha(l_s+l_c)+1}, (X - 2^{\alpha(l_s+l_c)+1})^2)$, due to Lemma 1, s can not be the product of any $s_i s_j$, $i, j < k$. Thus the proof is as before to solve a flexible RSA problem. Therefore, under the strong RSA assumption, we have the corollary as given above. □

Now we address the security of the sign and verify protocol.

Theorem 2. *Under the strong RSA assumption, the interactive protocol underlying the Sign and Verify protocol is a statistical zero-knowledge proof in honest-verifier mode that the TPM holds a keypair* (E, s) *such that* $E^s \equiv g \pmod{n}$ *and* s *lies in the correct interval.*

Proof. The proofs of completeness and statistical zero-knowledge property (simulator) follow the standard method. Here we only outline the existence of the knowledge extractor.

In the sign protocol, the TPM proves $T_2 \equiv g^b \pmod{n}$, and $b \in [Y - 2^{\alpha(l_c+l_b)+1}, Y + 2^{\alpha(l_c+l_b)+1}]$. This is a statistical honest-verifier zero-knowledge protocol that is secure under the strong RSA assumption. b can be recovered by a knowledge extractor following the standard method.

We need to show a knowledge extractor is able to recover the legitimate keypair once it has found two accepting tuples. Let (T_1, T_2, d_1, c, w_1), $(T_1, T_2, d_1, c', w_1')$ be two accepting tuples. Without loss of generality, we assume $c > c'$. Then we have

$$T_1^{w_1 - cX} T_2^c \equiv T_1^{w_1' - c'X} T_2^{c'} \equiv d_1 \pmod{n}.$$

It follows that

$$T_1^{(w_1' - w_1) + (c - c')X} \equiv T_2^{c - c'} \equiv g^{b(c - c')} \pmod{n}. \tag{1}$$

By the system parameter settings, we require $X > 2Y + 2^{\alpha(l_s+l_c)+2}$, and $Y > 2^{\alpha(l_c+l_b)+1}$. Then we can have

$$(c - c')X > (c - c')(Y + 2^{\alpha(l_c+l_b)+1} + 2^{\alpha(l_s+l_c)+2}).$$

Since we already have $b < Y + 2^{\alpha(l_c+l_b)+1}$, we further obtain

$$(c - c')X > (c - c')(b + 2^{\alpha(l_s+l_c)+2}).$$

Since $w_1, w_1' \in \pm\{0,1\}^{\alpha(l_s+l_c)+1}$, $w_1' - w_1$ is at least $-2^{\alpha(l_s+l_c)+2}$. Since $c - c'$ is at least 1, we finally have

$$(w_1' - w_1) + (c - c')X > b(c - c').$$

Due to Corollary 1, we can solve Equation 1 to obtain a pair (E, s) such $E^s \equiv g \pmod{n}$, $s \leq (w_1' - w_1) + (c - c')X$.

In our parameter settings, $(w_1' - w_1) + (c - c')X < (X - 2^{\alpha(l_s+l_c)+1})^2$. Due to Corollary 2, s must be a legitimate keypair in the correct interval. Therefore, (E, s) is a valid keypair, which completes the proof. □

For variable anonymity, $(j, T_3, d_3; T_1, T_2, d_1)$ are used to prove equality of the discrete logarithms of T_3 with base j, and T_2 with base T_1. This is also a statistical honest-verifier zero-knowledge protocol which has been proved secure under the strong RSA assumption.

Finally, we present a theorem for the unlinkability of a TPM's signatures.

Theorem 3 (Unlinkability). *Under the decisional Diffie-Hellman assumption over subset of QR_n, the protocol implements anonymous authentication such that it is infeasible to link the transactions by a TPM with different LCID.*

Proof. To decide whether two transactions are linked to a TPM, one needs to decide whether two equations are produced from the same E.

$$T_1,\ T_2 \equiv g^b \equiv T_1^s \pmod{n}$$
$$T_1',\ T_2' \equiv g^{b'} \equiv (T_1')^s \pmod{n}$$

Since T_1, T_1' are random generators of QR_n, under the DDH assumption it is infeasible to decide whether or not there exist an s such that $T_1^s \equiv T_2$, and $(T_1')^s \equiv T_2'$. The same argument can be applied to variable anonymity, in which case

$$T_3 \equiv j^s \pmod{n},\ T_3' \equiv j'^s \pmod{n}$$

where j, j' are two random generators of QR_n in different periods of time. □

6 Conclusion

In this paper, we have presented an efficient direct anonymous attestation scheme for Trusted Computing Platform. We adopt the same group certificate as the CM group signature scheme with new sign and verify protocols. Our construction supports authentication with total anonymity, variable anonymity, and rogue TPM tagging.

Compared to the current construction for DAA (the BCC scheme), our scheme has more efficient sign and verify protocols, thus all computation can

be completed in the TPM alone, making the computation distribution in the BCC scheme unnecessary. Therefore, our scheme is more attractive for embedded devices, such as cell phone, PDA, etc.

Finally, we proved our construction is secure under the strong RSA assumption and the decisional Diffie-Hellman assumption.

References

1. G. Ateniese, J. Camenisch, M. Joye, and G. Tsudik. A practical and provably secure coalition-resistant group signature scheme. In *Advances in Cryptology — Crypto*, pages 255–270, 2000.
2. N. Baric and B. Pfitzmann. Collision-free accumulators and fail-stop signature schemes without trees. In *Advances in Cryptology — Eurocrypto*, pages 480–494, 1997.
3. M. Bellare and P. Rogaway. Random oracles are practical: A paradigm for designing efficient procotols. In *First ACM Conference On computer and Communication Security*, pages 62–73. ACM Press, 1993.
4. D. Boneh, X. Boyen, and H.Shacham. Short group signatures. In *Advances in Cryptology — Crypto'04, LNCS 3152*, pages 41–55, 2004.
5. D. Boneh and H. Shacham. Group signatures with verifier-local revocation. In *Proc. of the 11th ACM Conference on Computer and Communications Security (CCS 2004)*, pages 168–177, 2004.
6. E. Brickell, J. Camenisch, and L. Chen. Direct anonymous attestation. In *ACM Conference on Computer and Communications Security*, pages 132–145, 2004.
7. J. Camenisch and J. Groth. Group signatures: Better efficiency and new theoretical aspects. In *Security in Communication Networks (SCN 2004), LNCS 3352*, pages 120–133, 2005.
8. J. Camenisch and M. Michels. A group signature scheme based on an RSA-variants. Technical Report RS-98-27, BRICS, University of Aarhus, Nov. 1998.
9. J. Camenisch and M. Stadler. Efficient group signature schemems for large groups. In *Advances in Cryptology — Crypto'97, LNCS 1294*, pages 410–424, 1997.
10. J. Camenisch and M. Stadler. A group signature scheme with improved efficiency. In *Advances in Cryptology — ASIACRYPT'98, LNCS 1514*, pages 160–174, 1998.
11. A. Chan, Y. Frankel, and Y. Tsiounis. Easy come - easy go divisible cash. In *K. Yyberg, editor, Advances in Cryptology – Eurocrypt'98, LNCS 1403*, pages 561 – 574. Sringer-Verlag, 1998.
12. D. Chaum and E. van Heyst. Group signature. In *Advances in Cryptology — Eurocrypt*, pages 390–407, 1992.
13. T. ElGamal. A public key cryptosystem and a signature scheme based on discrete logarithms. In *Advances in Cryptology — Crypto*, pages 10–18, 1984.
14. A. Fiat and A. Shamir. How to prove yourself: practical solutions to identification and signature problems. In *Advances in Cryptology — CRYPTO'86, LNCS 263*, pages 186–194. Springer-Verlag, 1987.
15. E. Fujisaki and T. Okamoto. Statistical zero knowledge protocols to prove modular polynomial relations. In *Advances in Cryptology — Crypto*, pages 16–30, 1997.
16. E. Fujisaki and T. Okamoto. A practical and provably secure scheme for publicly verifable secret sharing and its applications. In *Advances in Cryptology – EUROCRYPTO'98*, pages 32–46, 1998.

17. A. Kiayias, Y. Tsiounis, and M. Yung. Traceable signatures. In *Advances in Cryptology—Eurocypt, LNCS 3027*, pages 571–589. Springer-Verlag, 2004.
18. W. Mao. *Modern Cryptography: Theory & Practice*. Prentice Hall PTR, 2004.
19. A. J. Menezes, P. C. Oorschot, and S. A. Vanstone. *Handbook of Applied Cryptography*, pages 613–619. CRC Press, Inc, 1997.
20. A. Shamir. On the generation of cryptograpically strong psedorandom sequences. *ACM Transaction on computer systems*, 1, 1983.
21. TCG. http://www.trustedcomputinggroup.org.
22. TCG. TPM V1.2 Specification Changes: A summary of changes with respect to the v1.1b TPM specification, 2003.