An Anonymous Authentication Scheme for Trusted Computing Platform

He Ge [**]

Abstract. The Trusted Computing Platform is the industrial initiative to implement computer security. However, privacy protection is a critical problem that must be solved in Trusted Computing Platform. In this paper, we propose a simple and efficient method to implement anonymous authentication in such setting. The new scheme is proved to be secure under the strong RSA assumption and decisional Diffie-Hellman assumption.

Keywords: Privacy, Direct Anonymous Attestation, Trusted Computing Platform.

1 Introduction

Trusted Computing Group [17] is an industry standardization body to develop standards for Trusted Computing Platforms. A trusted computing platform is a computing device which is integrated with a cryptographic chip called trusted platform module (TPM). TPM is the root of trust. It is designed and manufactured in specific way such that all other remote parties trust some cryptographic computing results from this TPM. A trusted computing platform implements many security related features based on TPM, for example, secure boot, sealed storage, software integrity attestation, etc. More introduction about TPMs, and trusted computing platform can be found at the website of trusted computing group [17].

However, the deployment of TPM introduces privacy concerns. During a transaction a remote server knows a unique identifier for TPM. Different servers can cooperate with each other to link transactions made by the same TPM. To protect the privacy of a TPM owner, it is desirable to implement anonymous authentication, i.e, a TPM can prove its authenticity to a remote server without disclosing its real identity.

Two solutions have been proposed in the specification of TPM. TPM v1.1 is based on a trusted third party, called Privacy CA. A TPM generates a second RSA keypair called Attestation Identity Key (AIK). TPM sends AIK to Privacy CA to apply for a certificate on AIK. After TPM prove its ownership on a valid EK, Privacy CA issues the certificate on AIK. Later, TPM send the certificate

[**] Department of Computer Science and Engineering, University of North Texas, ge@unt.edu.

for AIK to a verifier and prove it owns such AIK. This way, the TPM hides its identity during the transaction. Obviously, this is not satisfactory solution, since each transaction needs the involvement of Privacy CA, and the compromise of CA will disclose all mapping between AIK's and EK.

The solution in TPM v1.2 is called direct anonymous attestation (DAA) in which TPM can directly proves its authenticity to a remote server with the help of Privacy CA. The current solution for DAA has been introduced by Brickell *et al.* in [4], which we refer to as BCC scheme in this paper. The solution is based on the research results from group signature which has been introduced by Chaum and Heyst in 1991 [11]. More specifically, it is based on the Camenisch-Lysyanskaya signature scheme [5] and ACJT group signature scheme [1]. A direct anonymous attestation can be seen as a group signature without opening capability.

In this paper, we propose a new solution to anonymous attestation method for Trusted Computing Platform. Our method is quite a simple and efficient solution compared to the current solution. The rest of this paper is organized as follows. The next section presents the model for our solution. Section 3 reviews some definitions and cryptographic assumptions. Section 4 introduces the building blocks for our scheme. Section 5 presents the proposed scheme. The security properties are considered in Section 6. The paper concludes in section 7.

2 The Model

We propose the model for our solution to anonymous authentication in Trusted Computing Platform. We adopt a different method for the creation of anonymous credential as in BCC scheme, a way more like the key generation for Endorsement Key EK. We suggest a new keypair for TPM, called Anonymous Authentication Key (AAK) This keypair is created during manufacturing just as EK, and solely used for anonymous authentication.

Definition 1 (The Model). *A manufacturer creates TPMs. After creation, TPMs independently work as expected, and cannot be interfered by the outside. Manufacturer and TPMs forms a group in which manufacturer holds group master key, while TPMs hold their anonymous authentication keypairs (AAK). The system should satisfy the following security requirements.*

1. *(Forgery-resistance) AAKs can only be created using manufacturer's master key.*
2. *(Anonymity) TPM can directly anonymous attest its authenticity to a remote server, without the help of a trusted third party. It is infeasible to extract the real identity of a TPM, or link the transactions by the same TPM.*
3. *(Revocation) TPM should be tamper-proof. On very rare circumstances a TPM may be compromised, and the group member certificate is exposed. In such case, a revocation mechanism should be available for the identification, or exclusion of rogue TPMs.*

3 Definitions and Preliminaries

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

Definition 2 (Special RSA Modulus [5]). *An RSA modulus $n = pq$ is called special if $p = 2p' + 1$ and $q = 2q' + 1$ where p' and q' also are prime numbers. Special RSA modulus is also called safe RSA modulus in some literature [1].*

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 $a^2 \equiv x \,(\mathrm{mod} n)$. 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 some properties about QR_n which will be used in the paper.

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 $g^a \pmod{n}$ is a generator of QR_n if and only if $GCD(a, |QR_n|) = 1$.*

Property 3 *If $x \in_R Z_n^*$ is uniformly distributed over Z_n^*, then $x^2 \pmod{n}$ is uniformly distributed over QR_n.*

The security of our techniques relies on the following security assumptions which are widely accepted in the cryptography literature. (see, for example, [2, 13, 6, 1]).

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 $v^e = u \,(\mathrm{mod} n)$. 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 over 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), $x, y, z \in_R Z_n$, 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 the subset of QR_n in [15], i.e., x, y, z are randomly chosen from some subsets of QR_n. They showed that the Decisional Diffie-Hellman Assumption is still attainable over subset of QR_n with the size down to at most $|QR_n|^{1/4}$. The unlinkability of our construction for direct anonymous attestation will depend on this variation of DDH assumption. Readers refer to their paper for deep discussion.

4 Building Blocks

Our main building blocks are *statistical honest-verifier zero knowledge proofs of knowledge* related to discrete logarithms over QR_n [9,14,8]. They include protocols for things such as the knowledge of a discrete logarithm, the knowledge of the equality of two discrete logarithms, the knowledge of the discrete logarithm that lies in certain interval, etc. Readers refer to the original papers for the protocol details.

4.1 Knowledge of Discrete Logarithm in an Interval

The zero-knowledge proof of the discrete logarithm in certain interval was introduced in [9,14]. Camenisch and Michels proposed a slight modification and provided a security proof in [6]. Here we introduce the version in [6,1].

Definition 4 (Protocol 1). *Let n be a special RSA modulus, QR_n be the quadratic residue group modulo n, and g is a generator of QR_n. α, l, l_c are security parameters that are all greater than 1. X is a constant number. A prover Alice knows x, the discrete logarithm of T_1, and $x \in [X - 2^l, X + 2^l]$. Alice demonstrates her knowledge of $x \in [X - 2^l, X + 2^l]$ as follows.*

- *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.*
- *Bob picks a random $c \in \{0,1\}^{l_c}$ and sends it to Alice.*
- *Alice computes*

$$w = t - c(x - X),$$

and $w \in \pm\{0,1\}^{\alpha(l+l_c)+1}$. Alice sends w to Bob.
- *Bob checks $w \in \pm\{0,1\}^{\alpha(l+l_c)+1}$ and*

$$g^{w-cX} T_1^c =?\ T_2 \pmod{n}.$$

If the equation holds, Alice proves knowledge of the discrete logarithm of T_1 lies in the range $[X - 2^{\alpha(l+l_c)}, X + 2^{\alpha(l+l_c)}]$.

Remark 1. It should be pointed out that Alice knows a secret x in $[X - 2^l, X + 2^l]$, the protocol only guarantees that x lies in the extended range $[X - 2^{\alpha(l+l_c)}, X + 2^{\alpha(l+l_c)}]$.

Remark 2. In the protocol, the generator g is a pre-defined system parameter. This protocol has been generalized to prove the relationship of two arbitrary elements $T_1, T_2 \in QR_n$ such that $T_2 = T_1^x \pmod{n}$, and $x \in [X - 2^l, X + 2^l]$ in [15]. In the generalized protocol, T_1, T_2 are both provided by a prover.

4.2 Knowledge of Equality of Discrete Logarithms with Different Bases

The protocol was first introduced in [10]. It is adopted to the group with unkown order in [6]. In this protocol, Alice proves she knows the discrete logarithm of two elements $T_1, T_2 \in QR_n$ with the generator g, h, respectively. That is, $g^x = T_1 \pmod{n}$, $h^x = T_2 \pmod{n}$. The protocol works as follows.

Definition 5 (Protocol 2). *Let $n, QR_n, g, \alpha, l, l_c, X$ as defined in* protocol 1.

- *Alice picks a random $t \in \pm\{0,1\}^{\alpha(l+l_c)}$ and computes*

  $$T_3 = g^t \pmod{n},\ T_4 = h^t \pmod{n}.$$

  Alice sends (T_1, T_2, T_3, T_4) to a verifier Bob.
- *Bob picks a random $c \in \{0,1\}^{l_c}$ and sends it to Alice.*
- *Alice computes*

  $$w = t - c(x - X),$$

  and $w \in \pm\{0,1\}^{\alpha(l+l_c)+1}$. Alice sends w to Bob.
- *Bob checks $w \in \pm\{0,1\}^{\alpha(l+l_c)+1}$ and*

  $$g^{w-cX}T_1^c =?\ T_3 \pmod{n},\ h^{w-cX}T_2^c =?\ T_4 \pmod{n}$$

  If the equations hold, Alice has proved knowledge of equality of two discrete logarithm of T_1, T_2 with base g, h.

4.3 Knowledge of the Co-primality of Two Discrete Logarithms

In this section, we propose a zero-knowledge protocol to show the co-primality of two discrete logarithms. That is, a prover demonstrates the knowledge of the discrete logarithms of two elements T_1, T_2 in QR_n are relatively prime. The method is based on the following theorem.

Theorem 1. *Let n be an RSA modulus. For a random element $u \in Z_n^*$, if one can find a tuple (T_1, T_2, x, y) such that $T_1^x T_2^y = u \pmod{n}$, then x, y must be relatively prime.*

Proof. By contradiction. If x, y are not co-prime, we assume $GCD(x, y) = e$, $x = k_1 e$, $y = k_2 e$. Then we have $T_1^x T_2^y = (T_1^{k_1} T_2^{k_2})^e = u \pmod{n}$. Thus, we find a pair (v, e) such that $v^e = u \pmod{n}$, where $v = T_1^{k_1} T_2^{k_2} \pmod{n}$, to solve a flexible RSA problem. This contradicts the strong RSA assumption. Therefore x, y must be relatively prime. □

Definition 6. (Sketch) *Suppose Alice knows a, c are relatively prime. She first uses GCD algorithm to compute b, d, such that $ab + cd = 1$. Then Alice computes*

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

$$T_3 = g^d \pmod{n},\ T_4 = T_3^c \pmod{n}.$$

Alice sends (T_1, T_2, T_3, T_4) to Bob, and proves she knows the discrete logarithms of T_2, T_4 with base T_1, T_3, respectively. Finally, $T_2 T_4 = g \pmod{n}$, this shows that the discrete logarithms of T_2, T_4 are relatively prime.

5 The Protocol to Implement Anonymous Authentication

The manufacturer, the producer of TPMs, sets various parameters, the lengths of which depend on a *security parameter*, which we denote by σ.

5.1 System Parameter Setting

The system parameters are set by manufacturer, these values are:

- n, g, h: n is a special RSA modulus such that $n = pq$, $p = 2p' + 1$, and $q = 2q' + 1$, where p and q are each at least σ bits long (so $p, q > 2^{\sigma}$), and p' and q' are prime. $g, h \in_R QR_n$ are random generators of the cyclic group QR_n. n, g, h are public values while p, q are kept secret by the administrator.
- α, l_c, l_s: security parameters that are greater than 1.
- X: a constant value. $X > 2^{\alpha(l_c+l_s)+1}$.

The verification of manufacturer's parameters can be accomplished by certain "independent evaluator". For example, manufacture should prove n is indeed the product of two safe primes through the protocol in [7]. An illustration of the system parameter is the setting of $\sigma = 1024$ (so n is 2048 bits), $\alpha = 9/8$, $X = 2^{520}$, $l_s = 300$ and $l_c = 160$.

5.2 Generation of Anonymous Authentication Key (*AAK*)

The specification for the generation of Endorsement Key (EK) states: "The TPM can generate the EK internally using the TPM_CreateEndorsementKey or by using an outside key generator. The EK needs to indicate the genealogy of the EK generation" [18] [1]. AAK is our proposal for TPM, we would like to follow the same specification for the generation of AAK.

Outside Key Generation. The method for key creation is straightforward. Manufacturer picks a random prime number $s \in [X - 2^{l_s}, X + 2^{l_s}]$ and computes

$$E = g^{s^{-1}} \pmod{n},$$

where s^{-1} is the inverse of s modulo $|QR_n| = p'q'$. (E, s) is TPM's AAK. s must be kept private by TPM, E may also be kept private. Manufacturer feeds (E, s) into TPM, and record E in its database. After that, s should be destroyed by manufacturer.

Internal Key Generation TPM internally generate $s \in [X - 2^{l_s}, X + 2^{l_s}]$ that will never be revealed to outside. $p'q'$ is temporally fed into TPM, and TPM compute s^{-1} modulo $p'q'$ and E. Since TPM is totally a passive chip created by manufacturer, it is surely "trusted" by manufacturer. TPM accomplishes these computation "honestly". After key generation, the copy of $p'q'$ should be destroyed by TPM.

[1] In practice, EK is generally produced internally by TPM.

5.3 Anonymous Authentication

The idea of our method to implement direct anonymous attestation is: TPM generates a random blinding integer b, computes $T_1 = E^b = g^{s^{-1}b} \pmod{n}$, $T_2 = g^b \pmod{n}$. Then TPM sends (T_1, T_2) to a verifier. TPM proves that $T_1^s = T_2 \pmod{n}$ and s lies in the correct interval; $T_2 = g^b \pmod{n}$ and s, b are coprime.

Definition 7 (Protocol 4: Direct Anonymous Attestation).

1. *A TPM Alice picks a random* $b \in_R [X - 2^{l_s}, X + 2^{l_s}]$. *She also picks* $t_1, t_2 \in_R \pm\{0,1\}^{\alpha(l+l_c)}$. *Alice uses GCD algorithm to solve* $2 \times sa + bd = 1$. *That is, TPM should find an even integer* $a' = 2a$ *such that* $sa' + bd = 1$.

$$T_1 = E^b \pmod{n},\ T_2 = g^b \pmod{n},\ T_3 = T_1^{t_1} \pmod{n},$$

$$T_4 = h^a \pmod{n},\ T_5 = (T_4^2)^s \pmod{n},\ T_6 = (T_4^2)^{t_1} \pmod{n},$$

$$T_7 = g^d \pmod{n},\ T_8 = T_7^b \pmod{n},\ T_9 = T_7^{t_2} \pmod{n},$$

$$T_{10} = g^{t_2} \pmod{n}$$

$(T_1, T_2, T_3; T_4, T_5, T_6)$ *are used to prove the equality of the discrete logarithm of* T_2, T_5 *with base* T_1, T_4^2 *respectively. Also, they are served to prove* s *lies in the correct range.* $(g, T_2, T_{10}; T_7, T_8, T_9)$ *are used to prove the equality of the discrete logarithm of* T_2, T_8 *with the base* g, T_7*, respectively. Alice sends* $(T_1, T_2, T_3, T_4, T_5, T_6, T_7, T_8, T_9, T_{10})$ *to a verifier Bob.*
2. *Bob picks number* $c_1, c_2 \in \{0,1\}^{l_c}$*, and sends them to Alice.*
3. *Alice computes*

$$w_1 = t_1 - c_1(s - X),\ w_2 = t_2 - c_2(b - X),$$

Alice sends (w_1, w_2) *to Bob.*
4. *Bob checks* $w_1, w_2 \in \pm\{0,1\}^{\alpha(l_s+l_c)+1}$,

$$T_1^{w_1 - c_1 X} T_2^{c_1} =?\ T_3 \pmod{n},\ (T_4^2)^{w_1 - c_1 X} T_5^{c_1} =?\ T_6 \pmod{n}$$

$$g^{w_2 - c_2 X} T_2^{c_2} =?\ T_{10} \pmod{n},\ T_7^{w_2 - c_2 X} T_8^{c_2} =?\ T_9 \pmod{n},$$

$$T_5 T_8 =?\ h \pmod{n}.$$

The computation of T_4^2 *serves to force it to be the element of* QR_n.

If all these equations hold, this finish the direct anonymous attestation.

Remark 3. Using the Fiat-Shamir heuristic[12], our DAA scheme can be turned into a non-interactive "signature of knowledge" scheme just as BCC scheme, which is secure in the random oracle model [3].

5.4 Rogue TPM Identification

We mentioned TPMs should be built tamper-proof. Otherwise, the whole efforts of trusted computing platform become meaningless. Even though, if in extreme circumstances, a TPM is compromised and its keypair is exposed, verifier should be able to identify an attestation request from rogue TPM. In our construction, it is quite easy to accomplish this task. Suppose a keypair (E, s) is put on revocation list, when a request comes, the verifier check

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

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

6 Security Properties of Proposed Scheme

Before discussing the security of the new scheme, we first introduce a lemma due to Shamir [16] that will be used shortly.

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

Proof. Since $GCD(x, y) = 1$, we can use the Extended GCD algorithm to find a and b such that $ay + bx = 1$, and let $z = v^a u^b$. Thus

$$z^x = v^{ax} u^{bx} = u^{ay+bx} = u \pmod{n}.$$

□

First, we need to address the issue of keypair forgery. In the context of trusted computing platform, TPM is produced tamper-proof. It should be extremely rare that a TPM can be compromised. If this could happen, attacker should not be able to forge new valid keypair. Therefore, we consider an attack model in which 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.

Theorem 2 (Forgery-resistance). *If there exists a probabilistic polynomial time algorithm which takes a list of valid keypairs, $(s_1, E_1), (s_2, E_2), \ldots, (s_k, E_k)$ and with non-negligible probability produces a new keypair (s, E) such that $E^s = g \pmod{n}$ 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 valid TPM 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):

1. First, we check if $\mathrm{GCD}(u, n) = 1$. 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 $\mathrm{GCD}(u, n) = 1$, so $u \in Z_n^*$.
2. We pick random prime numbers $s_1, s_2, \ldots, s_k$ with the required scope range, i.e., $s_i \in (0, X - 2^l)$, 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 with the appropriate length (constrained to be smaller than the lengths of p' and q'), it must be the case that $\mathrm{GCD}(r, |QR_n|) = 1$, 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 member keys, 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 $i = 1, \ldots, k$, $E_i^{s_i} = u^{s_1 s_2 \cdots s_k} = u^r = g \pmod{n}$.
4. We use the algorithm for creating new group member key to calculate (s, E), where s is in the required scope and $E^s = g = u^r \pmod{n}$.
5. If the forgery algorithm succeeded, then s will be different from all the s_i's, but will have the same length. Therefore, it is impossible for s to be an integer multiple of any of the s_i's, and since the s_i's are prime then it follows that $\mathrm{GCD}(s, s_1 s_2 \cdots s_k) = 1$, i.e., $\mathrm{GCD}(s, r) = 1$. Due to *lemma 1*, we can find a pair (y, s) such that
$$y^s = u \pmod{n},$$
which 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 $\mathrm{GCD}(u, n) \neq 1$, so let P_1 represent the probability of this event, which is negligible. When $\mathrm{GCD}(u, n) = 1$, 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.

□

Next, we address the security of our DAA scheme which is described as following theorem.

Theorem 3. *Under the strong RSA assumption, the direct anonymous attestation protocol is a statistical zero-knowledge honest-verifier proof of a keypair (E, s) such that $E^s = g \pmod n$ and s lies in the correct interval.*

Proof (Sketch). Our protocol uses the standard building blocks to accomplish direct anonymous attestation.

In the protocol, $(g, T_2, T_{10}; T_7, T_8, T_9)$ are used to prove the equality of the discrete logarithms of T_2 with base g, and T_8 with base T_7. This is the statistical honest-verifier zero-knowledge protocol that its security has been proved in the literature under the strong RSA assumption.

$(T_1, T_2, T_3; T_4, T_5, T_6)$ are used to prove the equality of the discrete logarithms of T_2 with base T_1, and T_5 with baser T_4^2. Also, they are served to prove this discrete logarithm $s \in [X - 2^l, X + 2^l]$. The squaring computation $(T_4^2 \pmod n)$ forces T_4^2 to be the elements in QR_n. Therefore we can use the generalized statistical honest-verifier zero-knowledge protocol in [15] to complete the proof. The protocol is also secure under the strong RSA assumption.

Since $T_5 T_8 = h \pmod n$, following *theorem 1*, the discrete logarithms of T_5, T_8, s and b, respectively, are co-prime.

Putting above together, the TPM demonstrates it knows s, b such that $T_1^s = g^b \pmod n$, and s, b are relatively prime. Due to *lemma 1*, the TPM can solve this equation and obtain $E^s = g \pmod n$, which is a valid keypair. □

Finally, we address the anonymity property of the scheme. The transaction-unlinkability of a TPM in our scheme relies on the decisional Diffie-Hellman assumption. We propose the following theorem.

Theorem 4 (Anonymity). *Under the decisional Diffie-Hellman assumption over subset of QR_n, the protocol implement anonymous authentication such that it is infeasible to link the transactions by a TPM.*

Proof (Sketch). Since b, b' are randomly selected by TPM with sufficiently large size, it is infeasible to extract b, b' from T_2, T_2' due to discrete logarithm over QR_n. Therefore, to link two arbitrary transactions, one needs to decide whether following equations are produced from the same E.

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

without the knowledge of b, b'. T_1, T_1' are the generators of QR_n due to the property of QR_n. Then the problem of linking these two equations reduces to decide the equality of discrete logarithms of $log_{T_1} T_2$, and $log_{T_1'} T_2'$. The choice of b in the scheme makes it infeasible to extract the discrete logarithms of T_1' with base T_1, and T_2' with base T_2, respectively. Therefore, it is impossible to make such decision under the decisional Diffie-Hellman assumption over subset of QR_n [15]. The same arguments can be applied to following equations

$$T_1,\ T_2 = T_1^s \pmod n,$$

$$T_4'^2,\ T_5' = (T_4'^2)^{s'} \pmod{n}.$$

Therefore, it is infeasible to link the transactions by a TPM. □

7 Conclusion

In this paper, we have presented a simple and efficient construction to implement anonymous authentication in Trusted Computing Platform. Our protocol is proved secure under the strong RSA assumption and the decisional Diffie-Hellman assumption over subset of QR_n.

Theorem 1 is an interesting result in the paper. It is an elegant corollary of the strong RSA assumption. Based on this result, we devised a novel knowledge proof of co-primality of the discrete logarithms. We believe the theorem could be used in other cryptographic construction.

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. E. Brickell, J. Camenisch, and L. Chen. Direct anonymous attestation. In *ACM Conference on Computer and Communications Security*, pages 132–145, 2004.
5. J. Camenisch and A. Lysyanskaya. A signature scheme with efficient protocols. In *SCN'02, LNCS 2576*, pages 268–289, 2002.
6. 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.
7. J. Camenisch and M. Michels. Proving in zero-knowledge that a number n is the product of two safe prime. In *Advances in Cryptology — EUROCRYPT'99, LNCS 1592*, pages 107–122, 1999.
8. J. Camenisch and M. Stadler. A group signature scheme with improved efficiency. In *Advances in Cryptology — ASIACRYPT'98, LNCS 1514*, pages 160–174, 1998.
9. 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.
10. D. Chaum and T. P. Pedersen. Wallet databases with observers. In *Rrnest F. Brickell, editor, Advances in Cryptology — CRYPTO'92, LNCS 740*, pages 89–105. Springer-Verlag, 1993.
11. D. Chaum and E. van Heyst. Group signature. In *Advances in Cryptology — Eurocrypt*, pages 390–407, 1992.
12. 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.

13. E. Fujisaki and T. Okamoto. Statistical zero knowledge protocols to prove modular polynomial relations. In *Advances in Cryptology — Crypto*, pages 16–30, 1997.
14. 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.
15. A. Kiayias, Y. Tsiounis, and M. Yung. Traceable signatures. In *Advances in Cryptology—Eurocypt, LNCS 3027*, pages 571–589. Springer-Verlag, 2004.
16. A. Shamir. On the generation of cryptograpically strong psedorandom sequences. *ACM Transaction on computer systems*, 1, 1983.
17. TCG. http://www.trustedcomputinggroup.org.
18. TCG. TPM Main: Part 1 design principles, 2005.