A Method to Implement Direct Anonymous Attestation

He Ge

Department of Computer Science and Engineering
University of North Texas
Denton, TX 76203

Abstract. In this report we present an efficient anonymous authentication scheme that works in a manner similarly to the so-called Direct Anonymous Attestation in the setting of the Trusted Computing Platform. Our construction implements features such as total anonymity, variable anonymity, and rogue TPM tagging. The new scheme is significantly simpler, and more efficient than the current solution that has been adopted in the standard specification, which implies our scheme might be deployed in the similar settings. We have proved the new scheme is secure under the strong RSA assumption, and the decisional Diffie-Hellman assumption.

Keywords: Privacy, Authentication, Direct Anonymous Attestation, Trusted Computing, Variable Anonymity.

1 Introduction

The Trusted Computing Group [19] is an industry consortium to develop standards for Trusted Computing Platforms. A trusted computing platform is a computing device embedded with a cryptographic chip called trusted platform module (TPM). The 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 the TPM such as secure boot, sealed storage, software integrity attestation, etc. More introduction about TPMs, and trusted computing platforms can be found at the website of the Trusted Computing Group [19].

However, the deployment of the TPM introduces privacy concerns. A TPM holds an RSA keypair called Endorsement Key (EK) which uniquely identifies this TPM. During a transaction a remote server knows public key of EK for a TPM. Therefore, all the transaction by the same TPM can be linked, and analyzed. 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, EK.

Two solutions have been proposed in the specification of the 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). The TPM sends the AIK to the Privacy CA applying for a certificate on this AIK. After the TPM prove its ownership on a valid EK, the Privacy CA issues the certificate on the AIK. Later, the TPM sends the certificate for the 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 satisfactory solution, since each transaction needs the involvement of the Privacy CA, and the compromise of the Privacy CA will disclose all mappings between AIKs and EKs.

The solution in TPM v1.2 is called Direct Anonymous Attestation (DAA). A Privacy CA is not necessary in the new method. A TPM can directly proves its authenticity to a verifier. The current solution is based on the research results from group signature which has been introduced by Chaum and Heyst [11]. More specifically, the current solution [4] is based on the Camenisch-Lysyanskaya signature scheme [5] and the group signature scheme in [1]. DAA can be seen as a group signature without open capability. In the rest of the paper, we refer to the current solution as the BCC scheme.

In this paper, we propose a construction that can efficiently carry out anonymous authentication similar to DAA. Our method is much simpler, and more efficient than the BCC scheme. The rest of this paper is organized as follows. Section 2 analyzes the characteristics of TPMs and our method. Section 3 reviews some definitions and cryptographic assumptions, and building block for our proposed scheme. We introduces our construction in Section 4. Security proofs are provided in Section 5. The paper concludes in Section 6.

2 The Characteristics of TPM

A TPM is a tamper-resistance cryptographic chip. When a TPM is manufactured, a unique RSA keypair, called Endorsement Key (EK), is produced and stored in the protective area of a TPM. The EK might be generated internally inside a TPM, or imported from an outside key generator. The public part of the EK is authenticated by the manufacturer, while private part of the EK will never be revealed to the outside. TPMs independently accomplish cryptographic computation inside themselves. Even the manufacturer should not be able to obtain the knowledge of these computation. TPMs are embedded into computing devices by device manufacturers. These devices (e.g., personal computers) are called trusted computing platforms.

At the heart of trusted computing platforms is the assumption that TPMs should independently work as expected, and be "trusted" by its manufacturer as well as remote parties. If we see TPMs as a group members in group signature schemes, the manufacturer is the group manager. However, in a group signature scheme, the group manager and a member are mutual distrustful. This difference has profound impacts on the protocol design. Essentially, the whole initiative of trusted computing platforms is based on the trust of TPMs. It is a hardware-assisted technique to enhance computer securities.

If the authentications for TPMs are directly based on EKs, all transactions by the same TPM can be linked. Furthermore, if a TPM is associated with a user's identity, the user may suffer from privacy abuse. The current solution to direct anonymous authentication, the BCC scheme, adopts the techniques from group signatures: a TPM applies for a credential from an issuer. Later, the TPM generates a special signature based on 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 authentications, it is called "Total Anonymity".

"Variable Anonymity" is a conditionally linkable anonymous authentication mechanism, 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 different time periods cannot be linked. When time interval becomes short, the method works like perfect unlinkable authentications. 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, it is likely this TPM has been compromised.

Rogue TPM tagging deals with the revocation of corrupted TPMs. When a broken TPM is being identified, its secrets (e.g, EK, credential) should be published on the revocation list. Verifiers can identify and exclude rogue TPMs on the list.

The current solution, the BCC scheme, is quite a complex construction with high computing intensity. To expedite authentication, computation has been distributed among a TPM and the Host into which the TPM is embedded. The TPM finishes the computation related to the signature, while the Host finishes the computation related to anonymity. The BCC schemes works fine on personal computers with high computing capabilities. However, it is still an expensive solution to low end devices.

In this paper, we proposed a scheme which also implements anonymous authentications with features such as total anonymity, variable anonymity, and rouge TPM tagging. We suggest a new keypair/credential for a TPM, called Anonymous Authentication Key (AAK), which serves the purpose of anonymous authentication. AAKs can be produced when manufactured, or issued by a trusted third party after TPMs are shipped through a Join protocol. Our scheme has much lower computation overhead, making it unnecessary the computation distribution among TPMs and Hosts. Therefore, our construction is more attractive for devices with lower computing capability, such as cell phones, PDAs, etc.

To facilitate the later discussion, we abstract the security requirements for the anonymous authentication in the trusted computing platform as follows.

Definition 1 (The Model). *A manufacturer creates TPMs. After creation, TPMs independently work as expected, and cannot be interfered by the outside. The manufacturer and TPMs forms a group in which the manufacturer holds group master key, while TPMs hold their anonymous authentication keypairs (AAK). The scheme includes four protocols:*

- **KeyGen:** *the manufacturer adopts KeyGen protocol to generate system parameters and its master key, and AAKs for TPMs; or, TPMs apply for AAKs from a trusted third party after TPMs are shipped.*
- **Sign:** *A TPM signs an anonymous signature following Sign protocol.*
- **Verify:** *A verifier follows Verify protocol to validate a signature by a TPM.*
- **Rogue TPM tagging:** *A verifier identifies the signatures by corrupted TPMs on the revocation list.*

The scheme should satisfy the following security requirements.

- **Forgery Resistance:** *AAK can only be created using manufacturer's group master key.*
- **Total Anonymity:** *A TPM can directly anonymously prove its authenticity to a remote server, without the help of a trusted third party. It is infeasible to link the transactions by the same TPM.*
- **Variable Anonymity** *The scheme also supports Variable Anonymity authentication.*

3 Definitions and Preliminaries

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

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 $a^2 \equiv x \,(\mathrm{mod} n)$. 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 $|QR_n| = \frac{1}{4}|Z_n^*|$.*

In the rest of the paper, n, QR_n will refer to special RSA modulus and quadratic residue group modulo n without explicitly suggestion. We list some properties which will be used shortly.

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, [2, 14, 8, 9, 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 $v^e = u \, (\bmod\, n)$. 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), $x, y, z \in_R Z_n$, 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 [16], 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.

3.2 Building Block

We review a knowledge protocol which will be used as building block to implement direct anonymous attestation in this paper. It is a zero-knowledge proof of the discrete logarithm in certain interval which was introduced in [10, 15]. It has been proved secure under the strong RSA assumption in the honest-verifier mode.

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 as follows.*

1. *Alice picks a random $t \in \pm\{0,1\}^{\alpha(l+l_c)}$ and computes $T_2 = g^t \; (\bmod\; n)$. 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*
$$w = t - c(x - X),$$
and $w \in \pm\{0,1\}^{\alpha(l+l_c)+1}$. Alice sends w to Bob.
4. *Bob checks $w \in \pm\{0,1\}^{\alpha(l+l_c)+1}$ and*
$$g^{w-cX} T_1^c = ?\; T_2 \; (\bmod\, n).$$
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 needs to 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. Using the Fiat-Shamir heuristic[13], the protocol can be turned into a non-interactive "signature of knowledge" scheme, which is secure in the random oracle model [3]. We will introduce the proposed scheme in the manner of "signature of knowledge" in the next section.

4 The Protocol to Implement Direct Anonymous Attestation

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

4.1 System Parameter Setting

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

- n, g: 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 is a generator of the cyclic group QR_n. n and g are public values while p and q are kept secret by the administrator.
- α, l_c, l_s, l_b: security parameters that are greater than 1.
- X, Y: X, Y are constant integers. $Y > 2^{\alpha(l_c+l_b)+1}$, and $X > Y + 2^{\alpha(l_c+l_b)} + 2^{\alpha(l_s+l_c)+2}$.
- The choice of X, l_s satisfies $X + 2^{l_s} < 2(X - 2^{l_s})$.
- $\mathcal{H} : \{0,1\}^* \to \{0,1\}^{l_c}$: a strong collision-resistant hash function.

4.2 Generation of Anonymous Authentication Key (AAK)

The specification for the generation of the 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" [20] [1]. The AAK is our proposal for TPMs, we would like to follow the same specification for the generation of the AAK.

Outside Key Generation. The method for key creation is straightforward. The 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 a TPM's AAK. s must be kept private by the TPM, E may also be kept private. The manufacturer feeds (E, s) into the TPM, and records E in its database. After that, s should be destroyed by the manufacturer.

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

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

$p'q'$ should be destroyed by the TPM.

Internal Key Generation (Method II) A TPM internally generates $s \in_R [X - 2^{l_s}, X + 2^{l_s}]$, and another random prime number s about the same size as s. The TPM submits $T = ss'$ to the manufacturer. The manufacturer computes $E' = g^{T^{-1}} \pmod n$ and feeds it into the TPM. The TPM computes

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

In this method, there are some restrictions on the choice of s, s' such that it infeasible to factorize T. Readers refer to [6, 12] for the deep discussion.

Remark 3. To ensure that TPMs be correctly produced, we would imagine certain entities, "independent evaluators", may take the responsibilities for the evaluation for TPMs' manufacturing processes. For example, the evaluator could require a manufacturer to prove n is really a special RSA modulus, this can be done by the protocol in [7]. The evaluator may verify whether the manufacturing processes conforms to the specification. It may even deploy some reverse-engineering processes to make sure no backdoors exist in TPMs.

4.3 Join Protocol

The outside key generation, internal key generation method I and II are accomplished in the manufacture process of TPMs. Therefore, TPMs are authenticated by the manufacturer. We can extend the internal key generation method II to a Join protocol as in generic group signature schemes. That is, a TPM can apply for a keypair/credential for direct anonymous attestation from other trusted third party, we may called it AAK issuer.

In a simplified Join protocol, a TPM calculates its signature on $T = ss'$ using its private key of EK, and sends T together with the signature to an issuer. The TPM may also uses the issuer's public key to encrypt the message. The issuer follows the method II to obtain E', and encrypts it by the TPM's public key. The issuer may also calculates its signature on E'. The TPM follows the same way to obtain the keypair/credential as in the method II. In this case, the issuer trusts a TPM should work "correctly" and "honestly".

An AAK issuer may only trust computation related to the endorsement key EK, and doubts other behaviors of TPMs. In such cases, a full Join protocol could be deployed which has been introduced in [6] for a group signature scheme. A TPM sends $T_1 = ss', T_2 = g^s \pmod n, T_3 = g^{s'} \pmod n$ to the issuer. The TPM proves:

- T_1 is the product of two prime numbers.
- The discrete logarithms of T_2, $g^{T_1} \pmod n$ with respect to g, T_3 are equal, and lies in the interval $[X - 2^{l_s}, X + 2^{l_s}]$.

After the issuer is convinced that (T_1, T_2, T_3) are correctly constructed by a real TPM, it computes

$$E = T_3^{T_1^{-1}} = (g^{s'})^{(ss')^{-1}} = g^{s^{-1}} \pmod n,$$

and sends E to the TPM.

The full Join protocol is less efficient than the one in the BCC scheme, due to the inefficiency of the proof for T_1 being the product of two prime numbers. However, a Join protocol runs rather infrequently, therefore it should not affect the system performance at all. We would imagine an AAK/credential should have rather long lifetime. Otherwise, frequently updating AAKs would make the system work in a manner like TPM v1.1, the Privacy CA model. Besides, in the context of trusted computing platform, we believe the simplified Join protocol should satisfy the requirement because TPMs are the "root of trust". If this is the case in practice, the Join protocol is indeed a very efficient one.

4.4 Total Anonymity Authentication

The idea of our method to implement "Total Anonymity" authentication is: A TPM picks a random blinding integer b, 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. The TPM proves that (T_1, T_2) is constructed from a legitimate keypair.

Definition 5 (Sign Protocol). *For a message m, the TPM proceeds the following steps:*

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*
$$T_1 = E^b \pmod{n},\ T_2 = g^b \pmod{n};\ d_1 = T_1^{t_1} \pmod{n},\ d_2 = g^{t_2} \pmod{n};$$
2. *Compute:*
$$c = \mathcal{H}(g||T_1||T_2||d_1||d_2||m);$$
$$w_1 = t_1 - c(s - X),\ w_2 = t_2 - c(b - Y).$$
3. *Output $(c, w_1, w_2, T_1, T_2, m)$.*

Definition 6 (Verify Protocol).

1. *Compute*
$$c' = \mathcal{H}(g||T_1||T_2||T_1^{w_1-cX}T_2^c||g^{w_2-cY}T_2^c||m).$$
2. *Accept the signature if and only if $c = c'$, $w_1 \in \pm\{0,1\}^{\alpha(l_s+l_c)+1}$, and $w_2 \in \pm\{0,1\}^{\alpha(l_b+l_c)+1}$,*

4.5 Variable Anonymity Authentication

We mentioned the Variable Anonymity is actually the pseudo-anonymity in a period of time. Our method is similar to the BCC scheme: in a period of time, a verifier produces a random generator h derived from its base name and other information, for instance, current time. Suppose a random generator h be computed as follows
$$h = (\mathcal{H}(bsn||cnt))^2 \pmod{n}.$$

Due to the property of QR_n, h is a random generator of QR_n. To implement variable anonymity, we add the following computations to *Sign protocol*, i.e., a TPM further computes

$$T_3 = h^s \pmod{n},\ d_3 = h^{t_1} \pmod{n},$$

$$c = \mathcal{H}(g||h||T_1||T_2||T_3||d_1||d_2||d_3||m);$$

and output $(c, w_1, w_2, T_1, T_2, T_3, m)$. Meantime, a verifier computes

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

Since h will be kept unchanged for certain time, therefore the same TPM will always produce the same T_3. The frequency of T_3 can be used by verifiers to identify suspicious attestation requests: if the same T_3 appear too many times in a time period, this could be the indication the TPM has been compromised. Since h changes from time to time, the unlinkability of the same TPM is achieved.

4.6 Rogue TPM Tagging

TPMs should be produced tamper-resistance. Otherwise, the whole efforts of trusted computing platforms become meaningless. Even though, if in extreme circumstances, a TPM is compromised and its keypair is being exposed, verifiers should be able to identify the attestation requests from a rogue TPM. To do so, the AAKs of exposed TPMs should be published on the revocation list. For a keypair (E, s) on the revocation list, a verifier can check

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

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

4.7 Performance Analysis

The computation complexity in the protocol is dominated by the modular squaring and multiplication. To estimate the computation overhead, it is sufficient to count total modular squarings and multiplications in the protocol. For simplicity, we estimate the computation overhead based on the techniques for general exponentiation [17]. Let the bit length of the binary representation of exponent be t_1, and t_2 be the number of 1's in the binary representation, the total computation overhead can be treated as t_1 squarings and t_2 multiplications. For example, if $y = g^x \pmod{n}$, and $x \in_R \{0,1\}^{160}$. We assume half of 160 bits of s will be 1. Then the total computation includes 160 squarings and 80 multiplications.

Suppose we set $\sigma = 1024$, then n is 2048 bits (p, q 1024 bits respectively). We further choose $\alpha = 9/8$, $l_c = 160, l_s = 300, l_b = 240$. We also set $X = 2^{512}$ (64 bytes), $Y = 2^{456}$ (57 bytes). The parameter setting conforms to the requirement of the Decisional Diffie-Hellman Assumption over the subset of QR_n. We should notice that significant part of s, b in binary representation are 0's. The computation with exponent b has 456 squarings and 121 multiplications. For the Total Anonymity authentication, a TPM needs 1880 ($456\times 3+512$) squarings, and 726 ($121 \times 2 + 512/2 + 456/2$) multiplications.

We have counted the total exponent bit-length in the BCC scheme which is 25844 for the Total Anonymity authentication. However, due to computation distribution among a TPM and its Host, some efficient exponentiation algorithm has been used in the Host part. According to their method, a TPM needs 4088 bits exponentiation, and Host needs at least 12098 exponentiation. The total exponent bit-length is 16186, which includes 16186 squarings and 8093 multiplications. If we assume the cost of squaring is equal to that of multiplication [2], our scheme is about 9 (24279/2606) times more efficient than the BCC scheme. Even if we only consider the computation in TPMs, our scheme is still more than 2 (6132/2606) times efficient than the BCC scheme.

It should be noticed that, if we want to, we can also distribute computation in our scheme. T_1, T_2, d_2, w_2 can be calculated by the Host, d_1, w_1 should be computed by the TPM. However, this is unnecessary since all the computation can be done by the TPM alone. Without computation distribution, the system design can be simplified. Thus, our method is more appropriate for mobile devices with low computing capabilities.

5 Security Properties of Proposed Scheme

We prepare two lemmas that will be used shortly in the later proof. The first lemma is due to Shamir [18], the second one can be seen as the first one's generalization.

[2] Squaring computation can at most two times faster than multiplication.

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

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

□

Lemma 2. *Let n be an integer. Given values $u, v \in Z_n^*$ and $x, y \in Z$ such that $GCD(x, y) = r$, 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 $GCD(x, y) = r$, 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 $z^k \equiv u \pmod{n}$. □

We have the following theorem with respect to the security of keypairs.

Theorem 1. *Under the strong RSA assumption, only manufacturer with the knowledge of factors of n can compute a legitimate keypair (E, s) such that $E^s = g \pmod{n}$, and s lies in the correct interval.*

Proof. Direct result from the Strong RSA Assumption. □

However, we need to address the issue of keypair forgery. In the context of trusted computing platform, TPMs are produced tamer-resistance. It should be extremely rare that a TPM can be compromised. If this happens, we should make it infeasible for attackers to forge a new valid AAK. 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, $(E_1, s_1), (E_2, s_2), \ldots, (E_k, s_k)$ 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 $\mathrm{GCD}(u, n) = 1$. 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 $s_1, s_2, \ldots, s_k$ in the required range $[X - 2^{l_s}, X + 2^{l_s}]$, and compute

$$r = s_1 s_2 \ldots s_k,$$

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

   Note that since the s_i values are primes, 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 keypairs, using the s_i values and E_i values calculated as follows:

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

   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 $s \in [X - 2^{l_s}, X + 2^{l_s}]$, 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. Since $X + 2^{l_s} < 2(X - 2^{l_s})$, 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 $\mathrm{GCD}(s, s_1 s_2 \cdots s_k) = 1$. Therefore, due to *lemma* 1, 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.

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.

□

From the step 5 of the above proof, we can obtain a corollary as follows.

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

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

If $s \in (X + 2^{l_s}, (X - 2^{l_s})^2)$, then s can not be the product of any $s_i s_j$, $i, j < k$. Therefore either $GCD(s, s_1 s_2 \cdots s_k) = 1$, or $GCD(s, s_1 s_2 \cdots s_k) = s_i$, $s = c \times s_i$, $c \in (0, X - 2^{l_s})$. In the first case, we can solve a flexible RSA problem. In the second case, we have $E^s = u^r \pmod n$, we can further have

$$(E^{s_i})^c = u^{s_1 s_2 \cdots s_k} \pmod n.$$

Since $GCD(c, s_1 s_2 \cdots s_k) = 1$, due to *lemma* 1, we can find a pair (y, c) such that

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

which means we solve a flexible RSA problem.

Therefore, under the strong RSA assumption, we have the corollary as above. □

Next, we further propose a lemma that will be used for the security proof of our protocol. It also could be seen as the generalization of Shamir's lemma.

Lemma 3. *Let n be an integer. For given values $u, v \in Z_n^*$ and $e, r \in Z_n$ such that $e > r$ and $v^e = u^r \pmod{n}$, there is an efficient way to compute the value (x, y) such that $x^y = u \pmod{n}$.*

Proof. When $e > r$, there are three cases:

1. Suppose $GCD(e, r) = 1$. Due to *lemma* 1, we can find a pair (y, e) such that

$$y^e = u \ (mod\ n).$$

2. Suppose $GCD(e, r) = r$. Since $e > r$, $e = kr$ for $k > 1$. Thus we have

$$v^e = v^{kr} = u^r \,(\mathrm{mod}\, n),\ \ v^k = u \ (mod\ n).$$

3. Suppose $GCD(e, r) = d$ such that $1 < d < r$. Thus we have $e = kd$ for $k > 1$. Due to *lemma* 2, we find

$$y^k = u \,(\mathrm{mod}\, n).$$

□

The restriction on the parameters X, Y, generally speaking, X should be greater than Y, are key to the validity of our protocol. In fact, the protocols would fail if this restriction were not be held. We propose the following theorem to address the security of Sign and Verify protocols.

Theorem 3. *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 a TPM holds an anonymous authentication keypair (AAK) (E, s) such that $E^s = g \pmod{n}$ and s lies in the correct interval.*

Proof (Sketch). The proofs of completeness and statistical zero-knowledge property (simulator) follow the same method as the proof for *protocol 1* (Definition 1) in [6]. Here we only outline the existence of the knowledge extractor.

In Sign protocol, the TPM follows *protocol 1* to prove $T_2 = g^b \pmod{n}$, and $b \in [Y - 2^{\alpha(l_c+l_b)}, Y + 2^{\alpha(l_c+l_b)}]$. 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 a 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

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

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

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

Since we also require $Y + 2^{\alpha(l_c+l_b)} > b$, 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

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

By *lemma* 3, 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 setting, $(w_1' - w_1) + (c - c')X < (X - l_s)^2$. By *corollary 1*, s must be a legitimate keypair in the correct interval. Therefore, (E, s) is a valid keypair. This shows a knowledge extractor can recover a legitimate keypair/credential. □

For Variable Anonymity, $(h, T_3, d_3; T_1, T_2, d_1)$ are used to prove the equality of the discrete logarithms of T_3 with base h, 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.

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

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

Now, an observer obtains a tuple $(T_1,\ T_2(= T_1^s),\ T_1'(= T_1^x),\ T_2'(= T_1^{xs}))$. If the observer can link (T_1, T_2) to (T_1', T_2'), this means he can decide the discrete logarithm of T_2' is the product of the discrete logarithms of T_2, T_1' with base T_1, respectively (the parameter settings of our scheme make it infeasible to extract the discrete logarithm of T_1', x, with base T_1). Thus, under the decisional Diffie-Hellman assumptions over subset of QR_n, it is intractable to link the transactions by a TPM.

The same argument can be applied to Variable Anonymity, in which

$$T_3 = h^s \pmod{n},\ T_3' = h'^s \pmod{n}$$

where h, h' are two random generators of QR_n in the different time period. □

6 Conclusion and Future Work

In this paper, we have presented an anonymous authentication scheme which provides ‘the features such as Total Anonymity, Variable Anonymity and Rouge TPM tagging. Due to its simplicity and efficiency, all computation can be done by a TPM (or a cryptographic chip) alone, making the scheme particularly appropriate for the mobile devices with low computing capabilities. We have proved the new scheme is secure under the strong RSA assumption and the Decisional Diffie-Hellman Assumption.

In this paper, we mainly focused on the presentation of our methodology and security proofs. We have not studied a protocol with implementation details, which would exactly follow the specification of the Trusted Computing Group. The BCC scheme is already an industrial level protocol such that all technical details are in place. In the future, we might discuss with the TCG's standard group the possibility of integrating our method into the TPM's specification, or apply the scheme to other appropriate settings.

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. Efficient group signature schemems for large groups. In *Advances in Cryptology — Crypto'97, LNCS 1294*, pages 410–424, 1997.
9. J. Camenisch and M. Stadler. A group signature scheme with improved efficiency. In *Advances in Cryptology — ASIACRYPT'98, LNCS 1514*, pages 160–174, 1998.
10. 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.
11. D. Chaum and E. van Heyst. Group signature. In *Advances in Cryptology — Eurocrypt*, pages 390–407, 1992.
12. D. Coppersmith. Finding a small root of a bivariate integer equation; factoring with high bits known. In *Ueli Maurer, editor, Advances in Cryptology —EUROCRYPT, LNCS 1070*, pages 178–189. Springer-Verlag, 1996.
13. 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.
14. E. Fujisaki and T. Okamoto. Statistical zero knowledge protocols to prove modular polynomial relations. In *Advances in Cryptology — Crypto*, pages 16–30, 1997.
15. 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.
16. A. Kiayias, Y. Tsiounis, and M. Yung. Traceable signatures. In *Advances in Cryptology—Eurocypt, LNCS 3027*, pages 571–589. Springer-Verlag, 2004.
17. A. J. Menezes, P. C. Oorschot, and S. A. Vanstone. *Handbook of Applied Cryptography*, pages 613–619. CRC Press, Inc, 1997.
18. A. Shamir. On the generation of cryptograpically strong psedorandom sequences. *ACM Transaction on computer systems*, 1, 1983.
19. TCG. http://www.trustedcomputinggroup.org.
20. TCG. TPM Main: Part 1 design principles, 2005.