Sequential Aggregate Signatures Working over Independent Homomorphic Trapdoor One-Way Permutation Domains

Huafei Zhu, Feng Bao, and Robert H. Deng

Department of Information Security, I^{2}R, A-Star, Singapore 119613
{huafei, baofeng}@i2r.a-star.edu.sg
School of Information Systems, Singapore Management University
robertdeng@smu.edu.sg

Abstract. The contribution of this paper has two folds. In the first fold, we propose a generic construction of sequential aggregate signatures from families of certificated trapdoor one-way permutations. We show that our construction is provably secure in the random oracle model assuming that the underlying homomorphic permutations are trapdoor one-way. Compared to Lysyanskaya et al's generic construction that is constructed from a trapdoor one-way permutation family working over the same domain [16], our scheme works over independent trapdoor one-way permutation domains. The flexible choice of the underlying permutation domains benefits our scheme to its applications in the real world where individual user may choose its working domain independently. In the second fold, we instantiate our generic construction with RSA so that the RSA moduli in our scheme can be chosen independently by individual user and thus the moduli is not required to be of the same length. Consequently, our proposed instantiation is the first scheme based on the RSA problem that works for any moduli − this is the most significant feature of our scheme different from the best results constructed from the RSA problem (say, Kawauchi et al's scheme [14], and Lysyanskaya et al's scheme [16]).

Keywords: Homomorphic trapdoor one-way permutation, sequential aggregate signature, signature scheme.

1 Introduction

In [3], Boneh, Gentry, Lynn and Shacham (BGLS) first introduced and formalized a new notion called aggregate signatures, together with a concrete implementation from the bilinear mapping. An aggregate signature scheme is the following cryptographic primitive: each of n users with a public/private key pair (PK_i, SK_i and a message m_i) wishes to attest to a message m_i. Informally the procedure can be stated as follows: user u_i first signs the correspondent message m_i ($1 \leq i \leq n$) and obtains a signature σ_i, and then n signatures can be combined by an unrelated party into an aggregate signature. Aggregate signatures are natural extension of multi-signature schemes. In a multi-signature scheme

S. Qing et al. (Eds.): ICICS 2005, LNCS 3783, pp. 207–219, 2005.

(e.g., [13], [22], [12], [24], [9], [11], [20] and [21]), a collection of users all sign the same message m. The result is a single signature (thus the conception of multi-signature schemes is not so good for practical applications since in certain cases we must be able to aggregate signatures on different messages, e.g., Certificate chaining, Secure Border Gateway Protocol). Recently, Micali, Ohta, and Reyzin [18], presented a clear security model and new constructions for multi-signature schemes from Schnorr's signature scheme. Another interesting construction was presented by Boldyreva [1] from the gap Diffie-Hellman assumption.

Burmester et al [2], Doi, Mambo, and Okamoto [8] (DMO), Mitomi and Miyaji [17] have already mentioned that when multiple entities sign a document (hence a set of users all sign the same message), the signing order often reflects the role of each signer and signatures with different signing orders are regarded as different multi-signature schemes. Thus a multi-signature scheme with message flexibility, order flexibility and order verifiability should be required. Burmester et al's then proposed an interesting order-specified multi-signature scheme from modified ElGamal signature scheme while Doi et al's construction is from the RSA problem. Notice that the protocol presented in [8] requires that the public keys corresponding the signing order have to be registered in advance, but it is not necessary in [17]. Later Kawauchi, Komano, Ohta and Tada [14] studied the possibility of simulation of Mitomi and Miyaji's schemes [17] in order to investigate whether that scheme is secure against active attacks. Unfortunately, they showed that Mitomi and Miyaji's scheme cannot be proved secure against active attacks. Tada [25] then proposed an order-specified multi-signature scheme based on the hardness of the discrete logarithm problem. The scheme allows the signing orders to be formed like any series-parallel graphs, which differs from the scheme [20]. The security is shown by using ID-reduction technique, which reduces the security of multi-signature schemes to those of multi-round identification schemes. Finally they constructed alternative RSA-based multi-signature scheme in order to overcome the problem from which Mitomi and Miyaji's scheme suffers.

Very recently, Lysyanskaya et al. [16] presented a clear security model and new constructions for order-specified multi-signature schemes that allow a collection of users to sign different messages (i.e., sequential aggregate signatures). In their paper [16], a generic construction for sequential aggregative signature schemes based on Full Domain Hash, which again is based on trapdoor permutations, is presented. They also instantiated the underlying trapdoor permutations with RSA. Their work is not trivial for several reasons, one being that their generic construction needs the trapdoor permutation to be over the same domain for each user, which is not the case for RSA.

1.1 Problem Statement

Sequential aggregate signatures are very useful for many network applications. They can be used in secure border gateway protocol (S-BGP) [15] for securing the UPDATE routing messages and in secure Ad Hoc On-Demand Vector Routing protocol. They can be used in hierarchical public key infrastructure to reduce

the certificate chain as well. To the best of our knowledge, all order-specified multi-signature schemes (or sequential aggregate signatures if each users signs different message) proved to be secure against active attacks are based on either the hardness of the discrete logarithm problem (for example, [18], [20] and [25]) or gap Diffie-Hellman problem (e.g., [1], [3] and [4]) while those based on the RSA problem suffer from unnecessary restrictions (e.g., [14] and [16]). Thus any more satisfactory solution to sequential aggregate signature scheme based on the RSA problem is certainly welcome (the main topic of this paper). Before starting our works, we would like to review the problems among the best results based on RSA below which are closely related to our works:

In [14], Kawauchi, Komano, Ohta and Tada proposed their construction from RSA trapdoor one-way permutations [23]: The key-generation step: P_i has RSA, which on input 1^{K_i}, randomly selects two distinct $K_i/2$-bit primes p_i and q_i, and calculates the modulus $N_i = p_i q_i$. It randomly picks up an encryption exponent $e_i \in Z_{\lambda(N_i)}$ and $\lambda(N_i) = \mathrm{LCM}(p_1-1, q_i-1)$. Also each P_i uses two hash functions G_i and H_i. The first one, H_i, called the compressor, maps as H_i: $\{0,1\}^* \rightarrow \{0,1\}^{k_2}$ and the other one, G_i, called the generator, maps as G_i: $\{0,1\}^{k_2} \rightarrow \{0,1\}^{K_{i-1}+k_1}$, where $K_i = 1 + K_{i-1} + k_1 + k_2$, $K_0 = k_0$ [1]. Signature generation step (for the same message m): Suppose that $\mathcal{P}$=$\{ P_1, \cdots, P_n \}$ generates a multi-signature for a message m. P_1 picks up an $r_1 \in_R \{0,1\}^{k_1}$ to compute $\omega_1 = H_1(m, r_1)$. Also P_1 computes $\sigma_1 = z_1^{d_1} \bmod N_1$, where $z_1 = 0||t_1||s_1||\omega_1$, $(t_1||s_1) = (0^{k_0}||r_1) \oplus G_1(\omega_1)$. Then P_1 sends (m, σ_1) as a signature for m to P_2. For each $i \in [2, n]$, the following is executed: P_i receives (m, σ_{i-1}) from P_{i-1}. P_i then checks the validity of received signature which is described below. If it is invalid, halts, otherwise, P_i picks up an $r_i \in_R \{0,1\}^{k_1}$ to compute $\omega_i = H_i(m, r_i, \sigma_{i-1})$. Also P_i computes $\sigma_i = z_i^{d_i} \bmod N_i$, where $z_i = 0||t_i||s_i||\omega_i$, $(t_i||s_i) = (\sigma_{i-1}||r_i) \oplus G_i(\omega_i)$. Then P_i sends (m, σ_i) as a signature for m to P_{i+1}, where P_{n+1} is a verifier. Verification step: Suppose that the verifier V receives (m, σ_n) as a multi-signature for a message m. For each $i = n$ to $i = 2$, the following is executed by the verifier. -First, V computes $z_i = \sigma_i^{e_i} \bmod N_i$, breaks up z_i as $b_i||t_i||s_i||\omega_i$. (That is, let b_i be the first bit of z_i, t_i the next K_{i-1} bits, s_i the next k_1 bits, and ω_i the remaining k_2 bits.) And V calculates $(\alpha_i||\beta_i) = (t_i||s_i) \oplus G_i(\omega_i)$. If $b_i = 0$ and $\omega_i = H_i(m, \beta_i, \alpha_i)$, then V computes $\alpha_i = \sigma_{i-1}$ and goes on the step. Finally V obtains σ_1, computes $z_1 = \sigma_1^{e_1} \bmod N_1$, breaks up z_1 as $b_1||t_1||s_1||\omega_1$, and calculates $(\alpha_1||\beta_1) = (t_1||s_1) \oplus G_1(\omega_1)$. If $b_1 = 0$, $\omega_1 = H_1(m, \beta_1, \alpha_1)$ and $\alpha_1 = 0^{k_0}$, then V returns 1 else return 0.

In [16], Lysyanskaya, Micali, Reyzin, Shacham proposed two approaches to instantiate their generic construction from RSA trapdoor one-way permutations: the first approach is to require the user's moduli to be arranged in increasing order: $N_1 < N_2 < \cdots < N_t$. At the verification, it is important to check that the i-th signature σ_i is actually less than N_i to ensure the signatures are unique if H is fixed. As long as $\log(N_1) - \log(N_t)$ is constant, the range of H is a subset of Z_{N_1} whose size is the constant fraction of N_1, the scheme will be secure; the

[1] Thus the restriction $|N_i|$-$|N_{i-1}| = 1 + k_1 + k_2$ is posted. We see that this unpleasant restriction should be removed from the point of views of practical applications.

second approach does not require the moduli to be arranged in increasing order, however they are required to be of the same length. The signature will expanded by n bits $b_1, \cdots, b_n$, where n is the total number of users. Namely, during signing, if $\sigma_i \geq N_{i+1}$, let $b_i = 1$; else, let $b_i = 0$. During the verification, if $b_i = 1$, add N_{i+1} to σ_i before proceeding with the verification of σ_i. Always, check that σ_i is in the correct range $0 \leq \sigma_i \leq N_i$ to ensure the uniqueness of signatures.

We would like provide the following remarks on KKOT scheme [14] and Lysyanskaya et al's scheme [16]: Lysyanskaya et al's first scheme can be viewed as improvement of of KKOT scheme [14]. The restriction of moduli in Tada's scheme $|N_i|$-$|N_{i-1}| = 1 + k_1 + k_2$ is weakened by the restriction of users's moduli to be arranged in increasing order $N_1 < N_2 < \cdots < N_t$ in Lysyanskaya et al's scheme. The second approach of Lysyanskaya et al's scheme does not require the moduli to be arranged in increasing order, however they are required to be of the same length and the signature size will expanded by n bits $b_1, \cdots, b_n$, where n is the total number of users. Namely, during signing, if $\sigma_i \geq N_{i+1}$, let $b_i = 1$; else, let $b_i = 0$. For applications of sequential aggregate signature schemes in the scenarios discussed above, the choice of N_i of a host is independent on the choice of another host N_j in the Internet (in case the underlying protocol is constructed from RSA). A reasonable assumption should be that the sizes of all moduli are bounded by a fixed size. Since there is no solution to this problem, an interesting research problem can be addressed as:

Research problem: how to construct practical and secure sequential aggregate signatures assuming that all moduli are bounded by a fixed size?

1.2 Our Works

In this paper, we first propose sequential aggregate signatures in which the set of participants is ordered. The aggregate signature is computed by having each signer, in turn, add his signature to it. We propose a generic construction of sequential aggregate signatures from families of certificated trapdoor one-way permutations. We then show that our construction is provably secure in the random oracle model assuming that the underlying homomorphic permutations are trapdoor one-way. Compared to Lysyanskaya et al's generic construction that is constructed from a trapdoor one-way permutation family working over the same domain [16], our scheme works over independent trapdoor one-way permutation domains. The flexible choice of the underlying permutation domains benefits our scheme to its applications in the real world where individual user may choose its working domain independently. Finally, we instantiate our generic construction with RSA that enjoys the following nice features: All three signatures are based on the hardness of RSA problem. Notice that the computation complexity of our scheme for the i-th users signing a message needs one exponent computation while the verification processing needs $(i-1)$ exponent computations. Thus all three schemes mentioned above have approximate computation complexity; In our scheme, the moduli are not required to be of the same length, i.e., each RSA modulus N_i is chosen independently by individual user. Thus our construction is the first scheme from RSA that works for any moduli − the most significant feature of our scheme different from all known schemes available in the literature.

2 Standard Signature Schemes Working over Extended Domains

2.1 Notions

Definition 1: A collection of permutation $F = \{f_i : D_i \rightarrow D_i | i \in I\}$ over some index set $I \subset \{0,1\}^*$ is said to be a family of trapdoor one-way permutations if: there is an efficient sampling algorithm $S(1^k)$ which outputs a random string index $i \in \{0,1\}^k \cap I$, and a trapdoor information sk_i; there is an efficient sampling algorithm which, on input i, outputs a random $x \in D_i$. Notice that there must be a mathematical structure associated with D_i. For simplicity, we assume that G_i is a group (not necessary a cyclic group, e.g., $D_i = Z^*_{n_i}$, if f_i is the RSA function); each f_i is efficiently computable given i and input $x \in D_i$; each f_i is efficiently invertible given the trapdoor information sk_i and output $y \in D_i$; for any probabilistic algorithm $\mathcal{A}$, $\mathcal{A}$ is said $(t(k), \varepsilon(k))$-break F, if $\mathcal{A}$ runs in time at most $t(k)$ and $Adv^F_{\mathcal{A}}(k) \geq \epsilon(k)$, where the advantage of $\mathcal{A}$ is defined as $Adv^F_{\mathcal{A}}(k) = \Pr[x' = x | (i, sk_i) \leftarrow S(1^k), x \leftarrow G_i, y = f_i(x), x' \leftarrow \mathcal{A}(i, y)]$. F is said to be $(t(k), \varepsilon(k))$-secure if no adversary $\mathcal{A}$ can $(t(k), \varepsilon(k))$-break it.

Definition 2: A collection of trapdoor one-way permutation $F = \{f_i : D_i \rightarrow D_i | i \in I\}$ over some index set $I \subset \{0,1\}^*$ is said to be homomorphic if for any $(D_i, f_i) \leftarrow i$ and $x, y \in D_i$, it satisfies $f_i(xy) = f_i(x) f_i(y)$.

We review the well known definition of security of ordinary digital signatures [10]. Existential unforgeability under a chosen message attack in the random oracle [5] for a signature scheme (KG, Sig, Vf) with a random oracle is defined using the following game between a challenger and an *adv* (notice that this security definition can also be applied to the scenario where a collection of random oracles are deployed): the challenger runs KG to obtain a public key pk and private key sk. The adversary *adv* is given pk; Proceeding adaptively, *adv* requests signatures with pk on at most q_{sig} messages of his choice $m_1, \cdots, m_{q_{sig}}$. The challenge responds to each query with a signature σ_i. Algorithm *adv* also adaptively asks for at most q_H queries of the random oracle H; *adv* outputs a pair (m, σ) and wins the game if $m \notin \{m_1, \cdots, m_{q_{sig}}\}$, and $Vf(pk, m, \sigma)$=1 (a valid signature of the message m).

By $\mathsf{AdvSig}_{\mathcal{A}}$, we denote the probability of success of an adversary.

Definition 3: We say a signature scheme is secure against adaptive chosen-message attack if for every polynomial time Turing machine $\mathcal{A}$, the probability $\mathsf{AdvSig}_{\mathcal{A}}$ that it wins the game is at most a negligible amount, where the probability is taken over coin tosses of KG and Sig and $\mathcal{A}$.

2.2 Generic Construction

We show that our signature scheme constructed from the extended domain of homomorphic trapdoor one-way permutations is provably secure against adaptive chosen message attack in the random oracle model [5]. We assume that a

permutation used to construct our sequential aggregate signature scheme must be a certificated one-trapdoor permutation. A certified trapdoor one-way permutation is one such that, for any describing string des, it is easy to determine whether des can have been output by a trapdoor one-way permutation generator, and thereby ensure that $f(\mathsf{des}, \cdot)$ is a permutation.

-Key generation algorithm KG: On input a security parameter l, k, KG specifies two cryptographic hash functions H: $\{0,1\}^* \rightarrow \{0,1\}^l$ and G: $\{0,1\}^t \rightarrow \{0,1\}^k$, $t = l - k$. On input k, KG outputs an instance of homomorphic trapdoor one-way permutation $\{f : D \rightarrow D\}$. We assume that each element in D can be represented by a k-bit string, i.e., $D \subset \{0,1\}^k$. We further assume that there is an efficiently computable mapping from $\{0,1\}^k \setminus D$ to D and given $\tau(x)$ and b, it is easy for one to recover $x \in \{0,1\}^k$, where $\tau(x)$ maps a element $x \in \{0,1\}^k$ to x modulo $|D|$, i.e., $\tau^0(x) = x$ if $x \in D$ while $\tau^1(x) = x \bmod |D|$, if $x \in \{0,1\}^k \setminus D$ (it will be clear in case that the underlying homomorphic one-way trapdoor permutation is instantiated with RSA, see next for more details).

-Signing algorithm: On input a message m, it computes $x = H(m)$ and then separates $x = y||z$, where $y \in \{0,1\}^k$ and $z \in \{0,1\}^t$, $t = l - k$. Finally, it computes $g = f^{-1}(\tau^b(y \oplus G(z)))||z$, where τ is an efficient mapping from $\{0,1\}^k \setminus D$ to D. That is, if $y \oplus G(z) \in D$, then the signature σ of the message m is $(g, 0)$ (b=0); if $y \oplus G(z) \in \{0,1\}^k \setminus D$, then the signature σ of the message m is denoted by $(g, 1)$ (b=1).

-The verification algorithm is given as input a signature $\sigma = (g, b)$, the messages m, and the correspondent public key pk and proceeds as follows: first it computes $x = H(m)$ and separates $x = y||z$ and $g = v||w$ and checks whether w is z, if not, it outputs 0; Else, it checks that pk and f is a certificated permutation. If both conditions are valid, then it computes y from the equation $\tau^b(y \oplus G(z)) = f(v)$. And output 1, if the test $H(m) = y||z$ is valid.

Lemma 1: Let f be a homomorphic trapdoor one-way permutation defined over D, and τ be an efficiently computable mapping from $\{0,1\}^k \setminus D$ to D and given $\tau(x)$, it is easy for one to recover $x \in \{0,1\}^k$, then our signature scheme is secure within the random oracle model.

Proof: We follow Coron's full domain reduction [7]. Let F be a forger that (t, q_{sig}, q_H, q_G, ϵ) breaks our sequential aggregate signature scheme. We assume that F never repeats a hash query and a signature query. We will build an inverter that can (t', ϵ') breaks underlying one-way trapdoor permutation. The inverter receives an input D, f, where D and f are public keys and $Y \in D$ is chosen uniformly at random. The inverter tries to find $X = f^{-1}(Y)$. The inverter starts running F and makes hash oracle queries and signing queries on behalf of the inverter. The inverter will answer hash oracle queries (H-oracle query and G-oracle query) and signing oracle queries. We assume for simplicity that when F requests a signature of the message m, it has already made the corresponding hash queries on m. If not, the inverter goes ahead to make H-oracle query and then to make G-oracle query. The inverter uses counter i initially set to zero. When F makes a H-oracle for m, the inverter increments i, sets $m_i = m$ and picks two random strings $y_i \in \{0,1\}^k$ and $z_i \in \{0,1\}^{l-k}$, where k is the bit length

of D. The output of H-oracle is x_i ($x_i = y_i||z_i$). To make the G-oracle query of the string z_i, the inverter first checks that whether z_i has been made the G-oracle query, and then the inverter will look at the H-oracle table for any query m_i queried to H-oracle with answer (y_i, z_i). It will output the string v_i which is defined below if z_i has already requested (notice that since H is a random oracle, the probability that the H-oracle will output $(*, z_i)$ for any message different than m_i is negligible assuming that the amount $1/2^{(l-t)}$ is negligible). The inverter now picks a random string $r_i \in D$ then returns v_i such that τ^b $(y_i \oplus v_i) = f(r_i)$ with probability p and v_i such that τ^b $(v_i \oplus y_i) = Yf(r_i)$ with probability $(1-p)$. Here p is a fixed probability which will be determined later. When F makes a signing query for m, it has already requested the hash queries of m, so $m = m_i$ for some i. If τ^b $(v_i \oplus y_i) = f(r_i)$, then the inverter returns r_i as the signature, otherwise the process stop and the inverter has failed. Eventually, F halts and outputs a forgery (m, σ). We assume that m has requested H-oracle and G-oracle of m before. If not, I goes ahead and makes the hash oracle queries itself, so that in any case, $m = m_i$ for some i. Then if τ^b $(v_i \oplus y_i) = Yf(r_i)$, we can compute $f^{-1}(Y) = \tau^b(v_i \oplus y_i)$ r_i with probability $p^{q_{sig}}(1-p)$. Setting $p = 1 - \frac{1}{q_{sig}+1}$, it follows that the probability that the inverter can find $y \in D$ such that $f(y) = Y$ with probability $\epsilon = exp(1)q_{sig}\epsilon'$ for sufficiently large q_{sig}.

2.3 Instantiated with RSA

Specifically to RSA instance [23], a certificated permutation could be done by having a trusted certification authority to check that N is a product of two large primes and e is relative prime to $\phi(N)$ before issuing a certificate. This check, however, requires one to place more trust in the authority than usual. Namely, the authority must be trusted not just to verify the identity of a key's purported owner, but also to perform verification of some complicated properties of the key. More precisely,

Lemma 2: suppose $\gcd(e, p-1) = k_1k_2$, $\gcd(e, q-1) = k_2k_3$, $\gcd(k_1, q-1) = 1$, $\gcd(k_3, p-1) = 1$ (i.e., we consider the case where k_1 is a factor of $p-1$, k_2 is a common divisor of $p-1$ and $q-1$, k_3 is a factor of $q-1$ and k_1, k_2 and k_3 are pair wise prime), then the number of set A is $k_1k_2k_2k_3$.

Proof: We consider the following three cases.

- case 1: $\gcd(e, p-1) = k \neq 1$, $\gcd(e, q-1) = 1$; Since $f(x) = x^e \bmod q$ is a permutation from Z_q^* to Z_q^*, we will consider $g(x) = x^e \bmod p$ from Z_p^* to Z_p^*. Let g be a generator of Z_p^* and denote $B = \{g^k : x \in Z_p^*\}$. Since $\gcd(e, p-1) = k \neq 1$, it follows that $\gcd(e/k, p-1) = 1$. Denote $g(x) = x^e \bmod p = g_1(g_2(x))$, where $g_1(x) = x^k \bmod p$ and $g_2(x) = x^{e/k} \bmod p$. Notice that g_2 is a permutation from Z_p^* to Z_p^* but $g_1(x)$ is a homomorphism from Z_p^* to Z_p^*. Since order $\mathrm{ord}(g^k) = \frac{p-1}{\gcd(k,p-1)}$, it follows that the number of elements of B is k and so the number of the set A is also k.
- case 2: $\gcd(e, p-1) = 1$, $\gcd(e, q-1) = k \neq 1$; Same claim as case 1.

– case 3: $k = k_1k_2k_3$, $\gcd(e, p-1) = k_1k_2$, $\gcd(e, q-1) = k_2k_3$, $\gcd(k_1, q-1) = 1$, $\gcd(k_3, p-1) = 1$ (i.e., we consider the case where k_1 is a factor of $p-1$, k_2 is a common divisor of $p-1$ and $q-1$, k_3 is a factor of $q-1$ and k_1, k_2 and k_3 are pair wise prime). Since Z_n^* and $Z_p^* \times Z_q^*$ are isomorphic and $f(x)=x^{e/k} \bmod n$ is a permutation from $Z_p^* \times Z_q^*$ to $Z_p^* \times Z_q^*$, we will only consider the function $g(x)= x^k \bmod n$. Denote $f_i(x) = x^{k_i} \bmod n$, then $g(x)=f_3\ (f_2\ (f_1(x))) = x^k \bmod n$. Denote $B=\{x^k : x \in Z_p^* \times Z_q^*\}$. We know that there are k_1k_2 elements $x_1 \in Z_p^*$ such that $x_1^k = 1$ (this 1 is in Z_p^*). And there are k_2k_3 elements $x_2 \in Z_q^*$ such that $x_2^k = 1$ (this 1 is in Z_q^*). So the number of the set A is $k_1k_2k_2k_3$.

Alternative approach may allow one to choose a large prime number e such that $e > N$, and then to show that e is a prime number by making use of the prime test protocol. This approach is attractive and has been used in [16]. Our sequential aggregate signature scheme will make use of this approach.

We show that our signature scheme described below is provably secure against adaptive chosen message attack in the random oracle model [5] assuming that the RSA problem (on input a randomly chosen $y \in Z_N^*$, and the public key (e, N), outputs $x \in Z_N^*$ such that $y = x^e \bmod N$) is hard.

-Key generation algorithm: On input a security parameter k, it generates an RSA public key (N, e) and secret key (N, d), ensuring that $|N|$=k-bit and that $e > N$ is a prime. Let $f^{-1}(x) = x^d \bmod N$ be the inverse function of RSA function $f(x) = x^e \bmod N$ ($ed \equiv 1 \bmod \phi(N)$). On input l and k, it also specifies two cryptographic hash functions H: $\{0,1\}^* \rightarrow \{0,1\}^l$ and G: $\{0,1\}^t \rightarrow \{0,1\}^k$, $t = l - k$.

-Signing algorithm: On input a message m, it computes $x = H(m)$ and then separates $x = y||z$, where $y \in \{0,1\}^k$ and $z \in \{0,1\}^t$, $t = l - k$. Finally, it computes $g = f^{-1}(y \oplus G(z))||z$. If $y \oplus G(z) > N$, then the signature σ of the message m is (g, b), where $b = 1$; if $y \oplus G(z) < N$, then the signature σ of the message m is denoted by (g, b), where $b = 0$ (in this case $\tau(x) = x - bN$ in the RSA instantiation); Note that the probability that the event $y \oplus G(z) = N$ happens is at most negligible amount, therefore we ignore this event in the following discussions.

-The verification is given as input a signature $\sigma = (g, b)$, the messages m, and the correspondent public key (N, e) and proceeds as follows: first it computes $x = H(m)$ and separates x =$y||z$ and g=$v||w$ and checks whether w is z, if not, it outputs 0; Else, it checks that $e > N$ and e is a prime number. If both conditions are valid, then it checks the validity of the equation $y = \mathcal{B}(f(v) + bN) \oplus G(z)$, where $\mathcal{B}(x)$ is the binary representation of $x \in \mathcal{Z}$. And output 1, if the equation is valid.

At first glance it seems that the adversary may have choice whether to use $b = 0$ or $b = 1$. However, this will result in two values $y \oplus G(z)$ that are guaranteed to be different: one is less than N and the other at least N. Hence uniqueness of σ implies uniqueness of b. Notice that once m is given, the value $y \oplus G(z)$ is determined assuming that $H(m)$ and $G(z)$ have already been queried. Furthermore, since the functionality of bit b defined above is to identify whether

$y \oplus G(z) > N$ or not, and $f(y \oplus G(z) + N) = f(y \oplus G(z))$ (an invariant of RSA function $f(x) = x^e \bmod n$), we can simply assume that $y \oplus G(z) < N$ in the following security argument. As an immediate application of Lemma 1, we have the following statement:

Corollary 1: Under the hardness of the RSA problem, the ordinary signature scheme described above is secure against adaptive chosen message attack in the random oracle model in the sense of [10].

3 Syntax, Security Definition, Construction of Sequential Aggregate Signature Scheme from Ordinary Signatures

3.1 Syntax

A sequential signature scheme (KG, AggSign, AggVf) consists of the following algorithms: Key generation algorithm (KG): On input l and k_i, KG outputs system parameters param (including an initial value $\mathcal{IV}$, without loss of generality, we assume that $\mathcal{IV}$ is a zero strings with length l-bit), on input param and user index $i \in \mathcal{I}$ and k_i, it outputs a public key and secret key pair (PK_i, SK_i) of a trapdoor one-way permutation f_i for a user i. Aggregate signing algorithm (AggSign): Given a message m_i to sign, and a sequential aggregate σ_{i-1} on messages $\{m_1, \cdots, m_{i-1}\}$ under respective public keys $PK_1, \cdots, PK_{i-1}$, where m_1 is the inmost message. All of $m_1, \cdots, m_{i-1}$ and $PK_1, \cdots, PK_{i-1}$ must be provided as inputs. AggSign first verifies that σ_{i-1} is a valid aggregate for messages $\{m_1, \cdots, m_{i-1}\}$ using the verification algorithm defined below (if i=1, the aggregate σ_0 is taken to be zero strings 0^l). If not, it outputs $\perp$, otherwise, it then adds a signature on m_i under SK_i to the aggregate and outputs a sequential aggregate σ_i on all i messages $m_1, \cdots, m_i$. Aggregate verifying algorithm (AggVf): Given a sequential aggregate signature σ_i on the messages $\{m_1, \cdots, m_i\}$ under the respective public keys $\{PK_1, \cdots, PK_i\}$. If any key appears twice, if any element PK_i does not describe a permutation or if the size of the messages is different from the size of the respective public keys reject. Otherwise, for $j = i, \cdots, 1$, set $\sigma_{j-1} = f_j(PK_1, \cdots, PK_j, \sigma_j)$. The verification of σ_{i-1} is processed recursively. The base case for recursion is $i = 0$, in which case simply check that σ_0. Accepts if σ_0 equals the zero strings.

3.2 The Definition of Security

The following security definition of sequential aggregative signature schemes is due to [16]. The aggregate forger $\mathcal{A}$ is provided with a initial value $\mathcal{IV}$, a set of public keys $PK_1, \cdots, PK_{i-1}$ and PK, generated at random. The adversary also is provided with $SK_1, \cdots, SK_{i-1}$; PK is called target public key. $\mathcal{A}$ requests sequential aggregate signatures with PK on messages of his choice. For each query, he supplies a sequential aggregate signature σ_{i-1} on some messages $m_1, \cdots, m_{i-1}$ under the distinct public keys $PK_1, \cdots, PK_{i-1}$, and an additional message m_i to be signed by the signing oracle under public key PK. Finally,

$\mathcal{A}$ outputs a valid signature σ_i of a message m_i which is associated with the aggregate σ_{i-1}. The forger wins if $\mathcal{A}$ did not request (m_i, σ_{i-1}) in the previous signing oracle queries. By $\mathsf{AdvAggSign}_{\mathcal{A}}$, we denote the probability of success of an adversary.

Definition 4: We say a sequential aggregate signature scheme is secure against adaptive chosen-message attack if for every polynomial time Turing machine $\mathcal{A}$, the probability $\mathsf{AdvAggSign}_{\mathcal{A}}$ that it wins the game is at most a negligible amount, where the probability is taken over coin tosses of KG and $\mathsf{AggSign}$ and $\mathcal{A}$.

3.3 Generic Construction from Independent Homomorphic Trapdoor One-Way Permutations

We now propose an interesting method to construct aggregate signature schemes from independent homomorphic trapdoor one-way permutations.

-Key generation: each participant i runs its key generation algorithm KG_i on input l, k_i, KG_i specifies two cryptographic hash functions H: $\{0,1\}^* \rightarrow \{0,1\}^l$ and G_i: $\{0,1\}^{t_i} \rightarrow \{0,1\}^{k_i}$, t_i=$l - k_i$ (notice that the system security parameter l should be shared by all participants). On input k_i, KG_i outputs an instance of homomorphic trapdoor one-way permutation $\{f_i : D_i \rightarrow D_i\}$. We assume that each element in D_i can be represented by a k_i-bit string, i.e., $D_i \subset \{0,1\}^{k_i}$. We further assume that there is an efficiently computable mapping from $\{0,1\}^{k_i} \setminus D_i$ to D_i and given $\tau(x)$ and b, it is easy for one to recover $x \in \{0,1\}^{k_i}$. The public key pk_i is (f_i, D_i, G_i, H); The private key sk_i is the trapdoor information of f_i;

-Aggregate signing: the input is a private key sk_i, a message $m \in \{0,1\}^*$ to be signed, and a sequential aggregate σ_{i-1} =$(g_{i-1}, b_1, \cdots, b_{i-1})$ on messages $\mathbf{m}|_1^{i-1}$, under the public keys $\mathbf{pk}|_1^{i-1}$ (for a vector $\mathbf{x}$, we denote a sub-vector containing $x_a, \cdots, x_b$ by $\mathbf{x}|_a^b$). Verify that σ' is a valid signature on $\mathbf{m}$ under $\mathbf{pk}$ using the verification algorithm below; if not, output $\perp$ indicating error. Otherwise, compute $h_i \leftarrow H(\mathbf{pk}|_1^i, \mathbf{m}|_1^i)$. The signer then rewrites $h_i \oplus g_{i-1} = x_i$: = $y_i||z_i$, and computes g_i =$f_i^{-1}(\tau^{b_i}(y_i \oplus G_i(z_i)))||z_i$. The signature of $\mathbf{m}|_1^i$ under $\mathbf{pk}|_1^i$ id denoted by σ_i =$(g_i, b_1, \cdots, b_i)$;

-Aggregate verification: the input is a sequential aggregate σ_i on message $\mathbf{m}|_1^i$ under $\mathbf{pk}|_1^i$. If any public key appears twice in $\mathbf{pk}|_1^i$, if any element of $\mathbf{pk}|_1^i$ does not describe a valid permutation, reject; Otherwise, for j=$i, \cdots, 1$, the verification algorithm processes the following steps recursively:

- for a given σ_i= (g_i, b_i), setting $v_i||w_i \leftarrow g_i$, $z_i \leftarrow w_i$;
- computing y_i from the equation $\tau^{b_i}(y_i \oplus w_i)$ =$f(v_i)$; and setting $x_i \leftarrow y_i||z_i$;
- computing $h_i \leftarrow H(\mathbf{pk}|_1^i, \mathbf{m}|_1^i)$, and then $g_{i-1} \leftarrow x_i \oplus h_i$.

-Accept if σ_0 is equal to 0^l (the initial value of sequential aggregate signature scheme);

3.4 The Proof of Security

Theorem: Let $\cup_{i\in I} f_i$ be a certificated homomorphic trapdoor permutation family. Then our sequential aggregate signature scheme described above is secure in the random oracle model.

Proof: Suppose *adv* is a forger algorithm that with non-negligible probability ϵ breaks the sequential aggregate signature scheme. We construct an algorithm F that inverts the permutation given by pk_i on a given input $z \in D_i$ which is chosen uniformly at random. Recall that the security definition of a sequential aggregate signature scheme allows an adversary to generate a collection of public keys pk_j $(j = 1, \cdots, i)$, and obtain the correspondent sk_j except for the trapdoor information sk_i of a target permutation. Thus, the security of the sequential aggregate signature scheme can be reduced to that of the underlying (ordinary) signature scheme. Since the simulator knows sk_j for $j \neq i$, it follows that the simulation of the j-th user can be trivially simulated while the simulation of i-th user can be simulated exactly as that described in the proof Lemma 1. Consequently, the proof of security of the theorem follows from an immediate application of Lemma 1.

3.5 Instantiated with RSA

Let H: $\{0,1\}^* \rightarrow \{0,1\}^l$ be a cryptographic hash function and $\mathcal{IV}$ be the initial vector that should be pre-described by a sequential aggregate signature scheme. The initial value could be a random l-bit string or an empty string. Without loss of generality, we assume that the initial value $\mathcal{IV}$ is 0^l. Our sequential aggregate signature scheme is described as follows:

-Key generation: Each user i generates an RSA public key (N_i, e_i) and secret key (N_i, d_i), ensuring that $|N_i| = k_i$ and that $e_i > N_i$ is a prime. Let G_i: $\{0,1\}^{t_i}$ $\rightarrow \{0,1\}^{k_i}$, be cryptographic hash function specified by each user i, $t_i = l - k_i$.

-AggSig: User i is given an aggregate signature g_{i-1} and $(b_1, \cdots, b_{i-1})$, a sequence of messages $m_1, \cdots, m_{i-1}$, and the corresponding keys $(N_1, e_1), \cdots, (N_{i-1}, e_{i-1})$. User i first verifies σ_{i-1}, using the verification procedure below, where $\sigma_0 = 0^l$. If this succeeds, user i computes $H_i = H(m_1, \cdots, m_i, (N_1, e_1), \cdots, (N_i, e_i))$ and computes $x_i = H_i \oplus g_{i-1}$. Then it separates $x_i = y_i||z_i$, where $y_i \in \{0,1\}^{k_i}$ and $z_i \in \{0,1\}^{t_i}$, $t_i = l - k_i$. Finally, it computes $g_i = f_i^{-1}(y_i \oplus G_i(z_i))||z_i$. By $\sigma_i \leftarrow (g_i, b_i)$, we denote the aggregate signature(if $y_i \oplus G_i(z_i) > N_i$, then b_i $=1$, if $y_i \oplus G_i(z_i) < N_i$, then $b_i = 0$; again we do not define the case $y_i \oplus G_i(z_i) = N_i$ since the probability the event happens is negligible), where $f_i^{-1}(y) = y^{d_i}$ mod N_i, the inverse of the RSA function $f_i(y) = y^{e_i}$ mod N_i defined over the domain $Z^*_{N_i}$.

-AggVf: The verification is given as input an aggregate signature g_i, $(b_1, \cdots, b_i)$, the messages $m_1, \cdots, m_i$, the correspondent public keys $(N_1, e_1), \cdots, (N_i, e_i)$ and proceeds as follows. Check that no keys appears twice, that $e_i > N_i$ is a prime. Then it computes:

- $H_i = H(m_1, \cdots, m_i, (N_1, e_1), \cdots, (N_i, e_i))$;
- Separating $g_i = v_i||w_i$;

- Recovering x_i form the trapdoor one-way permutation by computing $z_i \leftarrow w_i$, $y_i = \mathcal{B}_i(f_i(v_i) + b_i N_i) \oplus G_i(z_i)$, and $x_i = y_i || z_i$, where $\mathcal{B}_i(x)$ is the binary representation of $x \in \mathcal{Z}$ (with k_i bits).
- Recovering g_{i-1} by computing $x_i \oplus H_i$. The verification of (g_{i-1}, b_{i-1}) is processed recursively. The base case for recursion is $i = 0$, in which case simply check that $\sigma_0 = 0^l$.

Corollary 2: Our sequential aggregate signature scheme described above is secure in the sense of [16] in the random oracle model.

4 Conclusion

In this paper, a generic construction of sequential aggregate signatures has been constructed from homomorphic trapdoor one-way permutations. We have shown that our generic constructions are provably secure in the random oracle model assuming that the underlying homomorphic permutations are trapdoor one-way. We then instantiate our generic constructions with RSA. Compared the best results in the literature, say Kawauchi et al's scheme, and Lysyanskaya et al's scheme [16], our protocol has nice feature: the moduli are not required to be of the same length in our scheme, i.e., in our scheme N_i is chosen by each user independently. Thus we have proposed the first sequential aggregate signature scheme from RSA that works for any moduli.

References

1. A. Boldyreva. Efficient threshold signature, multisignature and blind signature schemes based on the gap-Diffie-Hellman-group signature scheme. In Y. Desmedt, editor, Proceedings of PKC 2003, volume 2567 of LNCS, pages 31C46. Springer-Verlag, 2003.
2. M. Burmester, Y. Desmedt, H. Doi, M. Mambo, E. Okamoto, M. Tada, Y. Yoshifuji: A Structured ElGamal-Type Multisignature Scheme. Public Key Cryptography 2000: 466-483
3. Dan Boneh, Craig Gentry, Ben Lynn, Hovav Shacham: Aggregate and Verifiably Encrypted Signatures from Bilinear Maps. EUROCRYPT 2003: 416-432.
4. Dan Boneh, Craig Gentry, Ben Lynn, Hovav Shacham: A Survey of Two Signature Aggregation Techniques. In CryptoBytes Vol. 6, No. 2, 2003.
5. M. Bellare and P. Rogaway. Random oracles are practical: A paradigm for designing efficient protocols. In D. Denning, R. Pyle, R. Ganesan, R. Sandhu, and V. Ashby, editors, Proceedings of CCS 1993, pages 62-73. ACM Press, 1993.
6. Jan Camenisch, Markus Michels: Proving in Zero-Knowledge that a Number Is the Product of Two Safe Primes. EUROCRYPT 1999: 107-122.
7. J. Coron: On the Exact Security of Full Domain Hash. CRYPTO 2000: 229-235.
8. H. Doi, M. Mambo, E. Okamoto: On the Security of the RSA-Based Multisignature Scheme for Various Group Structures. ACISP 2000: 352-367.
9. H. Doi, E. Okamoto, M. Mambo, and T. Uyematsu, Multisignature Scheme with Specified Order, Proc. of the 1994 Symposium on Cryptography and Information Security, SCIS94-2A, January 27 -29, 1994.

10. Shafi Goldwasser, Silvio Micali, Ronald L. Rivest: A Digital Signature Scheme Secure Against Adaptive Chosen-Message Attacks. SIAM J. Comput. 17(2): 281-308 (1988).
11. P. Horster, M. Michels, and H. Petersen Meta-multisignature schemes based on the discrete logarithm problem, Information Security -the Next Decade, Proc. of IFIP Sec95, Chapman-Hall pp. 128 -142 1995.
12. T. Hardjono, and Y. Zheng A practical digital multisignature scheme based on discrete logarithms, Lecture Notes in Computer Science 718, Proc. of Auscrypt92, Springer-Verlag, pp. 122-132, 1993.
13. K. Itakura and K. Nakamura. A public key cryptographic suitable for digital multisignatures. NEC Rearch and Development (71), page 1-8, 1983.
14. K. Kawauchi, Y. Komano, K. Ohta and M. Tada: Probabilistic multi-signature schemes using a one-way trapdoor permutation, IEICE transactions on fundamentals, vol.E87-A, no5, pp.1141 -1153, 2004. Previous version: Kei Kawauchi, Mitsuru Tada: On the Extract Security of Multi-signature Schemes Based on RSA. ACISP 2003: 336-349
15. S. Kent, C. Lynn and K. Seo: Secure Border Gateway Protocol (S-BGP). IEEE Journal on Selected Areas in Communicaitons, Vol. 18, No. 4, Apr. 2000.
16. Anna Lysyanskaya, Silvio Micali, Leonid Reyzin, Hovav Shacham: Sequential Aggregate Signatures from trapdoor one-way permutations. EUROCRYPT 2004: 74-90.
17. S. Mitomi and A. Miyaji, "A general model of multisignature schemes with message flexibility, order flexibility, and order verifiability", IEICE Trans., Fundamentals. vol. E84-A, No.10(2001), 2488 - 2499. Previous version: S. Mitomi and A. Miyaji, A multisignature scheme with message flexibility, order flexibility and order verifiability, Information security and privacy-Proceedings of ACISP 2000, Lecture Notes in Computer Science, 1841(2000), Springer-Verlag, p298 - 312.
18. S. Micali, K. Ohta, and L. Reyzin. Accountable-subgroup multisignatures (extended abstract). In Proceedings of CCS 2001, pages 245 -254. ACM Press, 2001.
19. K. Ohta, and T. Okamoto, A digital multisignature scheme based on the Fiat-Shamir scheme, Lecture Notes in Computer Science 739, Advances in Cryptology -Asiacrypt'91, Springer-Verlag, pp. 139-148, 1993.
20. K. Ohta and T. Okamoto. Multisignature schemes secure against active insider attacks. IEICE Trans. Fundamentals, E82-A(1):21C31, 1999.
21. K. Ohta and T. Okamoto: Generic construction methods of multi-signature schemes, Proceedings of The 2001 Symposium on Cryptography and Information Security (SCIS2001), vol.I, pp.31-36, 2001.
22. T. Okamoto. A digital multisignature scheme using bijective public-key cryptosystems. ACM Trans. Computer Systems, 6(4):432C41, November 1988.
23. R. Rivest, A. Shamir, and L.M. Adleman. A Method for Obtaining Digital Signatures and Public-Key Cryptosystems. Communications of the ACM 21(2): 120-126 (1978).
24. A. Shimbo, Multisignature Schemes Based on the ElGamal Scheme, Proc. of The 1994 Symposium on Cryptography and Information Security,January 27 - 29, 1994.
25. M. Tada: A secure multisignature scheme with signing order Verifiability, IEICE transactions on fundamentals, vol.E86-A, no.1, pp.73-88, 2003. Previous version: M. Tada: An Order-Specified Multisignature Scheme Secure against Active Insider Attacks. ACISP 2002: 328-345.