International Journal of Future Generation Communication and Networking
Vol.7, No.3 (2014), pp.229-240
http://dx.doi.org/10.14257/ijfgcn.2014.7.3.21

An Efficient Synchronized Aggregate Signature Scheme From Standard RSA Assumption

Xinshun Guo[1] and Zhiwei Wang[2]

[1]*School of Business Information Management, Shanghai University of International Business and Economics, Shanghai 201620, P.R.China*
[2]*College of Computer, Nanjing University of Posts and Telecommunications, Nanjing 210003, P.R.China*
gxs@suibe.edu.cn, zhwwang@njupt.edu.cn

Abstract

An aggregate is a digital signature that supports aggregation where any one given n signatures on n distinct messages from n different users can aggregate all these signatures into a single one. We proposed an efficient synchronized aggregate signature scheme from the standard RSA assumption. Our construction is based on the Hohenburger and Waters RSA signature (Crypto 09). This scheme can be proved existentially unforgeable against adaptive chosen-message attacks without random oracles.

Keywords: *aggregate signature, HW signature, RSA assumption, standard model*

1. Introduction

The notion of aggregate signature is introduced by Boneh *et al*., [1] , which allows an efficient algorithm to aggregate n signatures of n distinct messages from n different users into one single signature. The result aggregate signature can convince the verifier that n different users did indeed sign n distinct messages. This primitive has many applications where storage or bandwidth is restricted, and thus one wants to reduce the total cryptographic overhead.

Boneh *et al.*, presented the first aggregate signature [1], which was based on BLS siganture[2] in groups with bilinear maps. Subsequently, Lysyanskaya *et al.* presented a sequential RSA-based scheme [3] that, while more limited, could be instantiated using more general assumptions. In Eurocrypt 2006, Lu *et al.*, proposed another sequential aggregate siganture [4] from Waters signature [5]. A sequential aggregate signature is that signatures can only be aggregated by sequentially passing the aggregate from one signer to the next. However, it is in appropriate for many important applications where signers cannot be conventiently arranged in sequence. Furthermore, the schemes in [1-3] are proved secure in the random oracle model, only the scheme in [4] is proved secure in the standard model.

In order to eliminate the need for signer interaction, Gentry et al. proposed an aggregate signature scheme [6] where nointeraction between signers was required, provide that all signers have a global strategy for choosing a value ω that is used during signing. Only the signature with the same ω can be aggregated. This kind of aggregate signature is called **synchronized aggregate signature**, and Gentry *et al.*, scheme is under CDH assumption in the random oracle model. In CCS 2010, Ahn et al. proposed an novel sychronized aggregate signature [11] based on the stateful signature of Hohenburger and Waters (Eurocrypt 2009) [7]. They define ω as a time period. Although their scheme can can be proved secure under

ISSN: 2233-7857 IJFGCN
Copyright ⓒ 2014 SERSC

the CDH assumption in the standard model, the computational overhead is heavy. To verify an aggregate of N signatures, requires $k+3$ pairings, (where k is a security parameter,) plus $N+2$ exponentiations. The computational cost of pairing is usually considered very heavy.

We present a synchronized aggregate signature under Hohenburger and Waters's RSA signature [8](Crypto 09). Our scheme is provably secure based on the standard RSA assumption in the standard model. Compared with Ahn *et al.*, scheme [11], our scheme is efficient and without pairing.

In Section 2, we review the standard RSA assumption and define some notations. In Section 3, we review the secure definition of synchronized signature. Section 4 presents the synchronized signature scheme under the standard RSA assumption. Section 5 presents the security analysis to our scheme. Section 6 gives the conclusion.

2. Preliminary

RSA Assumption. Let κ be the security parameter. Let integer n be the product of two κ -bits, distinct odd primes p, q. Let e be a randomly chosen positive integer less than and relatively prime to $\phi(n) = (p-1)(q-1)$. Given (n, e) and randomly chosen $y \in \mathbf{Z}_n^*$, we say that the (τ_R, ε_R) -RSA assumption holds if for all τ_R -time adversary A

$$\Pr[(x) \leftarrow A(n, y, e), x \in \mathbf{Z}_n^*, x^e = y \bmod n] \leq \varepsilon_R$$

where the probability is over the random choices of y, n, e, and the random coins of A .

Notation. For $B \in \{0,1\}^{l_B}$, let $B = b_1 b_2 \cdots b_{l_B}$ be the binary presentation of B with $b_i \in \{0,1\}$ for $i \in \{1, \cdots, l_B\}$. We denote by $B^{(i)} = 0^{l_B - i} b_1 \cdots b_i \in \{0,1\}^{l_B}$ the prefix of B that consists of $l_B - i$ zeros and the first i bits of B . We use QR_n to denote the set of quadratic residues modulo $n \in \mathbf{N}$, i.e., $QR_n = \{x \mid \exists y \in \mathbf{Z}_n^* : y^2 = x \bmod n\}$. We use $\lambda \in \mathbf{N}$ to indicate the security parameter while 1^λ describes the string that consist of λ ones. Let $l = l(\lambda)$ and $l_B = l_B(\lambda)$ be polynomials with $l_B \leq l$. We define a prime map function called *nearprime*: $\{0,1\}^{l_B} \rightarrow P_l$, P_l is a set of primes belongs to $\{1, \cdots, 2^l\}$. The function *nearprime* maps to the smallest prime that is equal or greater than the input value.

Nearprime Function. We will construct a *nearprime* function: $r : \{0,1\}^{l_B} \rightarrow P_l$, that is the same prime mapping function used by [9]. Firstly, we choose a random key K from a key space K for the pseudo-random function (PRF) $F : \{0,1\}^* \rightarrow \{0,1\}^{l_B}$, and a random value $s \in_R \{0,1\}^{l_B}$. Secondly, we establish a function as $H_K(X) = F_K(X) \oplus s$. Then the final definition of r is $r(X) = nearprime(H_K(X))$.

For convenience, we additionally define a new function $E : \{0,1\}^{l_B} \rightarrow \mathbf{N}$ as $E(B) = \prod_{i=1}^{l_B} r(B^{(i)})$.

 Copyright ⓒ 2014 SERSC

3. Secure Definition of Synchronized Aggregate Signature

In an aggregate signature, Suppose that there are N signers, each chooses a public-private key pair (pk_i, sk_i) in the same system parameters. Signer u_i signs a message M_i to obtain a signature σ_i. Then there is a public aggregation algorithm that takes as input all of individual signatures $\sigma_1, \cdots, \sigma_N$ and outputs a short compressed signature σ_{agg}. Anyone can aggregate the signatures. Moreover, the aggregation can be performed incrementally. There is also an aggregate verification algorithm that takes as input $pk_1, \cdots, pk_N, M_1, \cdots, M_N$ and σ_{agg}, and decides whether the aggregate signature is valid. This is also true for **synchronzied aggregate signature**, except that all signers have a synchronized clock and two restrictions, which can be described as follows:

1. A signer can sign at most one signature in each time period (one value of clock denotes a time period).

2. Only the signatures with the same clock value can be aggregated.

Then, we give the formal definition of synchronized aggregate signature as Gentry et al.[6] and Ahn et al.[11].

Definition 1. (Synchronized Aggregate Signatures) A synchronized aggregate signature scheme is consisted with six algorithms (**ParaGen, Setup, Sign, Verify, Aggregate, AggVerify**).

ParaGen(1^λ**)**: This algorithm generates the public parameters $param$.

Setup(1^λ**,** $param$ **):** This algorithm takes in 1^λ and $param$**,** outputs a keypair (pk, sk).

Sign(sk, M, t **):** This algorithm takes input as a secret key sk, a message M, and the current clock value t, and produces a signature σ.

Verify(pk, M, σ **):** This algorithm takes in a public key pk, a message M, and a corresponding signature σ, and verifies whether σ is a valid signature. If valid, it outputs 1, otherwise 0.

Aggregate($(pk_1, M_1, \sigma_1), \cdots, (pk_N, M_N, \sigma_N)$ **):** On input a sequence of public keys $(pk_1, \cdots, pk_N)$, messages $(M_1, \cdots, M_N)$, and corresponding signatures $(\sigma_1, \cdots, \sigma_N)$, this algorithm outputs an aggregate signature σ_{agg}.

AggVerify(($pk_1, \cdots, pk_N$ **),(** $M_1, \cdots, M_N$ **),** σ_{agg} **):**On input a sequence of public keys $(pk_1, \cdots, pk_N)$, messages $(M_1, \cdots, M_N)$, and corresponding aggregate signature σ_{agg}, this algorithm outputs 1 if σ_{agg} is a valid aggregate signature and 0 otherwise.

Next, we recall Ahn *et al.*'s secure definition of aggregate signature[11].

Exist Unforgeability According to the standard definition of security of Goldwasser , Micali and Rivest [10], a signature scheme S =(**ParaGen, Setup, Sign, Verify**) is existentially unforgeable under an adaptive chosen messages attack if no forger that has q times access to a signing oracle $O(sk,.)$ can produce a new valid signature on a new

message. The security definition of aggregate signature is a extension to the normal digital signature, which can be described as a game between a challenger C and an adversary A .

Setup: The challenger C runs **ParaGen(**1^{λ}**)** algorithm to obtain the public parameters $param$, then it runs the **Setup(**1^{λ}**)** algorithm for N times to obtain N distinct key pairs $(pk_1, sk_1), \cdots, (pk_N, sk_N)$. Finally, C sends $(pk_1, (pk_2, sk_2), \cdots, (pk_N, sk_N))$ to A .

Queries: The adversary A can request a signature on a message of its adaptively choice under sk_1 for each time period 1 to q , provide that at most one query is made per time period. The challenger responses **Sign**(sk_1, M_i, t_i) to a query for message M_i at time period t_i .

Forgery: Finally, the adversary A outputs $((pk_1, \cdots, pk_N), (M'_1, \cdots, M'_N), \sigma^*_{agg})$ and wins the game if

1. A did not make a signature query on M'_1 ;

2. **AggVerify**$((pk_1, \cdots, pk_N), (M'_1, \cdots, M'_N), \sigma^*_{agg})$=1.

We define $AggAdv_{\mathrm{A}}$ to be the probability that the adversary A wins the above game, taken over the coin tosses made by A and the challenger C .

Definition 2. (Exist Unforgeability) *An adversary* A $(\tau, q, N, \varepsilon)$*- breaks an* N *users aggregate signature scheme if* A *runs at most in time* τ *,* A *makes at most* q *signature queries and* $AggAdv_{\mathrm{A}}$ *is at least* ε *. If no forger can* $(\tau, q, N, \varepsilon)$*-break this scheme, we call this aggregate signature scheme is a* $(\tau, q, N, \varepsilon)$ *- existentially unforgeable under an adaptive chosen message attack.*

4. Proposed Synchronized Signature Scheme

4.1. Our Construction

We now define our new synchronized signature scheme. Our construction is based on the Hohenburger and Waters RSA signature[8]. The drapdoor of our scheme is factorization of integer, so N users in our scheme should have the same modulo n and p, q . Thus, N users have the same secret key $sk = (p, q)$, however, their public keys are partially distinct .

ParaGen(1^{λ}**).** Let $l = l(\lambda)$, $l_m = l_m(\lambda)$, and $l_p = l_p(\lambda)$ be polynomials and $l_n = 2l_p$, $l_m \le l \le l_p$. Messages can be interpreted as values in $\{0,1\}^{l_m}$.

Setup(1^{λ}, $param$ **)**.The setup algorithm chooses a RSA modulus $n = pq$. The length of n is l_n , and the length of p, q is l_p . Next, the setup algorithm chooses a generator $g \in QR_n$. Finally, it draws $K \in_R \mathrm{K}$ and $s \in_R \{0,1\}^{l_B}$ and publishes $pk = (n, K, s, g)$ and keeps the secret key $sk = (p, q)$.

 Copyright ⓒ 2014 SERSC

(Remark: N different users have the corresponding secret/ public key pairs as $(sk_1 = (p, q), pk_1 = (n, K, s, g_1)), \cdots, (sk_N = (p, q), pk_N = (n, K, s, g_N))$.)

Sign(sk, M, t): We assume the signer is given the clock value $t = clock() \in \{0,1\}^{l_B}$, as input to the algorithm. It keeps an internal state t_{prev} that denotes the last time period on which it issued a signature. If $t_{prev} = t$ or $t \geq 2^{\lambda}$, then it aborts. Otherwise, it record the current time period as $t_{prev} = t$. The sign algorithm computes a signature on M under secret key sk and time period t as

$$\sigma = (\sigma' = (g^M)^{1/E(t)} \bmod n, t).$$

Verify(pk, M, σ): The verify algorithm firstly parses σ as (σ', t), and checks whether $0 < t < 2^{\lambda}$. If this is false, it rejects. Then, it verifys the signature by checking that

$$\sigma'^{E(t)} = g^M \bmod n.$$

Aggregate($(pk_1, M_1, \sigma_1), \cdots, (pk_N, M_N, \sigma_N)$): This algorithm first parses σ_i as (σ_i', t), and checks the second element of σ_i (for $i = 1, \cdots, N$). If they are distinct, it outputs "failure". Otherwise, if they are the same, the algorithm computes the aggregate signature as

$$\sigma_{agg} = (\gamma = \prod_{i=1}^{N} \sigma_i', t).$$

AggVerify(($pk_1, \cdots, pk_N$),($M_1, \cdots, M_N$),σ_{agg}): This algorithm parses σ_{agg} as (γ, t), and checks that $0 < t < 2^{\lambda}$, if this false, it rejects. This algorithm verifys the signature by checking that

$$\gamma^{E(t)} = \prod_{i=1}^{N} g_i^{M_i} \bmod n.$$

4.2. Efficiency Analysis

Our scheme is practical, since an aggregate signature only requires one element in group of QR_n and a small integer. To verify an aggregate of N signatures, requires only one modular exponentiation and one modular N multi-exponentiations. In general, if N is relatively small, the computaional cost of one modular N multi-exponentiations and one modular exponentiation are almost equal. In the following table, we compare our scheme with other aggregate signature schemes in type, assumption, model, aggregate signature size and the computational cost of aggregate verifying.

Copyright ⓒ 2014 SERSC

Table 1. Summary of Full, Sequential and Sychronized Aggregate Signatures. Let N be the Number of Individual Signatures and k Denotes a Special security Parameter (which could be five in practice). TDP Stands for a Trapdoor Permutation. Size for Aggregate Signatures Count Group Elements and Integers

Scheme	Type	Assumption	Model	Agg Size	Agg Verify
Boneh scheme[1]	full	CDH	ROM	1	$N+1$ pairings
Lysyanskaya scheme[3]	seq	Cert TDP	ROM	-	-
Lu scheme[4]	seq	CDH	Standard	2	2 pairings
Gentry scheme[6]	sync	CDH	ROM	3	3 pairings
Ahn scheme[11]	sync	CDH	Standard	3	$k+3$ pairings
Our scheme	sync	RSA	Standard	2	About 2 modular exponentiations

From the above table, we can see that prior aggregate signatures, Lysyanskaya *et al.*'s scheme and Lu et al.'s scheme are sequential aggregate signatures. Prior synchronized or full schemes, due to Boneh *et al.*'s scheme, Gentry *et al.*'s scheme are only offered heuristic security arguments in the random oracle model. Only one synchronized aggregate signature scheme with a standard security proof exists, due to Ahn *et al.*'s scheme, and its security is based the CDH assumption. Furthermore, it require $k+3$ pairings in the verification of N aggregated signatures.

Our scheme can be proved secure under the standard RSA assumption in the standard model, and only requires About 2 modular exponentiations in the aggregate verification (If N is relatively small, the cost of one modular N multi-exponentiations is almost equal to one modular exponentiation). Thus, our scheme is more practical than theirs.

5. Security Analysis

In this section, we will prove our scheme secure under the standard RSA assumption in the standard model.

Theorem 1. *Assume the* (τ_R, ε_R) *-RSA assumption holds, then our synchronized aggregate signature scheme is* $(q, \tau_w, \varepsilon_w, N)$ *-secure against adaptive chosen message attacks provide that*

$$\varepsilon_w \le \frac{3q2^{l_B}(2^{l_B} - q + l_B)}{2(2^{l_B} - q)}\varepsilon_R + N \cdot 2^{-(l_p - 2)}, \text{ and } \tau_w \approx \tau_R .$$

(l_B is the length of clock value t, and it is relatively small.)

 Copyright ⓒ 2014 SERSC

Proof. Assume that A is forger that $(q, \tau_w, \varepsilon_w, N)$-breaks the exist unforgeability of our scheme. Then we can construct a challenger C that breaks the RSA assumption in time τ_R with probability ε_R. Given the RSA challenge (n, y, e), C 's goal is to compute x such that $x^e = y \bmod n$.

C denotes $1, \cdots, q$ in the binary presentation by $t_1, \cdots, t_q$ (clock values), and all the length of t_i $(i = 1, \cdots, q)$ is l_B. We assume that the adversary makes q queries for each time period 1 to q and receives q responses to a query for message M_i at time period $t_i \in \{t_1, \cdots, t_q\}$. Finally, the adversary returns a forgery of $((pk_1, \cdots, pk_N), (M'_1, \cdots, M'_N), \sigma^*_{agg} = (\gamma^*, t^*))$. We should differentiate two types of forgeries.

- **Type I Forgery**, there exists non-empty set $J \subseteq \{1, \cdots, q\}$ such that all $j \in J$ it holds that $E(t^*) = E(t_j)$. We note that if $r()$ is a collision free function for all t_i $(i \in \{1, \cdots, q\})$ and their prefixs, then $|J| = 1$ and $t^* = t_j$.

- **Type II Forgery**, there exists no $j \in \{1, \cdots, q\}$, such that $E(t^*) = E(t_j)$.

Type I Forgery.

Setup: C guesses $1 \le j \le q$ (such that $E(t^*) = E(t_j)$). Next, C draws $K \in \mathsf{K}$ and $s \in \{0,1\}^{l_B}$ uniformly at random from the set of values with $r(t_j^{(l_B)}) = nearprime(F_K(t_j^{(l_B)}) \oplus s) = e$. Now, C chooses N random numbers $\omega_1, \cdots, \omega_N$ from $Z_{(n-1)/4}$, and computes

$$g_1 = y^{2\omega_1 \prod_{i=1}^{q} E(t_i)} \bmod n$$
$$\vdots$$
$$g_j = y^{2\omega_j \prod_{i=1, i\neq j}^{q} E(t_i)} \bmod n$$
$$\vdots$$
$$g_N = y^{2\omega_N \prod_{i=1}^{q} E(t_i)} \bmod n$$

Obviously, $g_1, \cdots, g_N \in QR_n$. Finally, C sends the N public keys $pk_1 = (n, K, s, g_1), \cdots, pk_N = (n, K, s, g_N)$ to A. Since $sk_1 = sk_2 = \cdots = sk_N$, C can not send $sk_2, \cdots, sk_N$ to the adversary A.

Queries: The adversary A requests a signature on a message of its adaptively choice under (sk_1, pk_1) for each time period 1 to q, provide that at most one query is made per time period. On receiving the signature query on M_j, $j \in \{1, \cdots, q\}$. C computes

$$\sigma'_j = y^{2\omega_1 M_j \prod_{i=1, i\neq j}^{q} E(t_i)} \bmod n,$$

and (σ'_j, t_j) is a correct signature on message M_j in time period j. C sends (σ'_j, t_j) as a response to A.

Forgery: Finally, the adversary A outputs (($pk_1, \cdots, pk_N$), ($M'_1, \cdots, M'_N$), $\sigma^*_{agg} = (\gamma^*, t^*)$). If $t^* \in \{t_1, \cdots, t_q\}$, and $t^* = t_j$ ($j \in \{1, \cdots, q\}$), then C guesses correctly,. From the aggregate verification equation, we have that

$$\gamma^{*E(t^*)} = \prod_{i=1}^{N} g_i^{M'_i} \bmod n$$
$$= y^{2\prod_{i=1}^{q} E(t_i)(\sum_{i=1,i\neq j}^{N} \omega_i M'_i) + 2\omega_j M'_j \prod_{i=1,i\neq j}^{q} E(t_i)} \bmod n .$$

If $GCD(e, 2\prod_{i=1}^{q} E(t_i)(\sum_{i=1,i\neq j}^{N} \omega_i M'_i) + 2\omega_j M'_j \prod_{i=1,i\neq j}^{q} E(t_i)) = 1$, we can compute two values $\alpha_1, \alpha_2 \in Z$ such that

$$\alpha_1 e + 2\alpha_2 (\prod_{i=1}^{q} E(t_i)(\sum_{i=1,i\neq j}^{N} \omega_i M'_i) + \omega_j M'_j \prod_{i=1,i\neq j}^{q} E(t_i)) = 1 ,$$

and find a response to the RSA challenge as

$$y^{1/e} = y^{\alpha_1} \gamma^{*\alpha_2 \prod_{j=1}^{l_B - 1} r(t^*(j))} \bmod n$$

We have completed the simulation of the scheme. In the following step, we will analyze the bounded probability of C.

1) In the **Setup** step, if C guesses correctly, the probability is $1/q$.

2) In the **Setup** step, C chooses $\omega_1, \cdots, \omega_N$ from $Z_{(n-1)/4}$. It holds that $(n-1)/4 = p'q' + (p' + q')/2 > p'q'$, (assume $q' > p'$) which implies that

$$Pr[\chi \in Z_{(n-1)/4}, \chi \notin Z_{p'q'}] < \frac{2}{p' + q'} < \frac{1}{p'} < 2^{-(l_p - 1)} .$$

3) In the **Forgery** step, if C does not abort, which means that $t^* \in \{t_1, \cdots, t_q\}$ and $t^* = t_j$ ($j \in \{1, \cdots, q\}$), the probability is $\frac{1}{2^{l_B}}$.

4) In the **Forgery** step, since $\omega_1, \cdots, \omega_N$ are hidden from the adversary's view, the probability of $GCD(e, 2\prod_{i=1}^{q} E(t_i)(\sum_{i=1,i\neq j}^{N} \omega_i M'_i) + 2\omega_j M'_j \prod_{i=1,i\neq j}^{q} E(t_i)) = 1$ is at least $2/3$.

Putting the above together, we get that

$$\varepsilon_w \leq \frac{3q2^{l_B}}{2} \varepsilon_R + N \cdot 2^{-(l_p - 1)} , \text{ and } \tau_w \approx \tau_R .$$

 Copyright ⓒ 2014 SERSC

Type II Forgery.

Setup: C guesses $1 \le i^* \le q$ (a random number with the longest prefix in common with the random number t^* in the forgery signature) and $1 \le j^* \le l_B$ (the length of the longest common prefix plus one). Let the guessed prefix be $\overline{t}$ and Next, C draws $K \in \mathsf{K}$ and $s \in \{0,1\}^{l_B}$ uniformly at random from the set of values with $r(\overline{t}) = nearprime(F_K(\overline{t}) \oplus s) = e$. Now, C chooses N random numbers $\omega_1, \cdots, \omega_N$ from $Z_{(n-1)/4}$, and computes

$$g_1 = y^{2\omega_1 \prod_{i=1}^{q} E(t_i)} \bmod n$$
$$\vdots$$
$$g_N = y^{2\omega_N \prod_{i=1}^{q} E(t_i)} \bmod n$$

Obviously, $g_1, \cdots, g_N \in QR_n$. Finally, C sends the N public keys $pk_1 = (n, K, s, g_1), \cdots, pk_N = (n, K, s, g_N)$ to A. Since $sk_1 = sk_2 = \cdots = sk_N$, C can not send $sk_2, \cdots, sk_N$ to the adversary A.

Queries: The adversary A requests a signature on a message of its adaptively choice under sk_1 for each time period 1 to q, provide that at most one query is made per time period. On receiving the signature query on M_j, $j \in \{1, \cdots, q\}$. C computes

$$\sigma'_j = y^{2\omega_1 M_j \prod_{i=1, i\neq j}^{q} E(t_i)} \bmod n,$$

and (σ'_j, j) is a correct signature on message M_j in time period j. C sends (σ'_j, j) as a response to A.

Forgery: Finally, the adversary A outputs $((pk_1, \cdots, pk_N), (M'_1, \cdots, M'_N), \sigma^*_{agg} = (\gamma^*, t^*))$. If $t^* \in \{t_1, \cdots, t_q\}$, then C aborts. Otherwise, if $\overline{t} = t^{*(j^*)}$, C guesses correctly. From the aggregate verification equation, we have that

$$\gamma^{*E(t^*)} = \prod_{i=1}^{N} g_i^{M'_i} \bmod n$$
$$= y^{2\prod_{i=1}^{q} E(t_i)(\sum_{i=1}^{N} \omega_i M'_i)} \bmod n$$

If $GCD(e, 2\prod_{i=1}^{q} E(t_i)(\sum_{i=1}^{N} \omega_i M'_i)) = 1$, we can compute two values $\alpha_1, \alpha_2 \in Z$ such that

$$\alpha_1 e + 2\alpha_2 \prod_{i=1}^{q} E(t_i)(\sum_{i=1}^{N} \omega_i M'_i) = 1,$$

and find a response to the RSA challenge as

$$y^{1/e} = y^{\alpha_1} \gamma^{*\alpha_2 \prod_{j=1, j \neq j^*}^{l_B} r(t^{*(j)})} \bmod n$$

We have completed the simulation of the scheme. In the following step, we will analyze the bounded probability of $\mathcal{C}$.

1) In the **Setup** step, if $\mathcal{C}$ guesses correctly, the probability is $\frac{1}{q l_B}$.

2) In the **Setup** step, $\mathcal{C}$ chooses $\omega_1, \cdots, \omega_N$ from $Z_{(n-1)/4}$. It holds that $(n-1)/4 = p'q' + (p'+q')/2 > p'q'$, (assume $q' > p'$) which implies that

$$Pr[\chi \in Z_{(n-1)/4}, \chi \notin Z_{p'q'}] < \frac{2}{p'+q'} < \frac{1}{p'} < 2^{-(l_p-1)}.$$

3) In the **Forgery** step, if $\mathcal{C}$ does not abort, which means that $t^* \notin \{t_1, \cdots, t_q\}$, the probability is $\frac{2^{l_B} - q}{2^{l_B}}$.

4) In the **Forgery** step, since $\omega_1, \cdots, \omega_N$ are hidden from the adversary's view, the probability of $GCD(e, 2\prod_{i=1}^{q} E(t_i)(\sum_{i=1}^{N} \omega_i M_i')) = 1$ is at least $2/3$.

Putting the above together, we get that

$\varepsilon_w \leq \frac{3 q l_B 2^{l_B}}{2(2^{l_B} - q)} \varepsilon_R + N \cdot 2^{-(l_p-1)}$, and $\tau_w \approx \tau_R$.

6. Conclusion

An aggregate signature scheme is a digital signature scheme where anyone given N signatures on n messages from N users can aggregate all these signatures into a single short signature. Thus, it is a good candidate for a variety of communication applications where routing flexibility, speed and low bandwidth are needed. For example, in the wireless sensor network(WSN), sensors often run on battery power and must minimize radio communications. Digital signature offer better security properties, but can add significant bandwidth overhead. Aggregate signature can greatly reduce the bandwidth requirements the message signing imposes on WSN. Rather than carry all signature data, intermediate routing nodes can perform signature aggregation at any point where multiple signatures must be routed towards the collector.

In this paper, we propose the first synchronized aggregate signature from the standard RSA assumption without random oracles. It requires that signers have access to a synchronized clock and only signatures from the same period can be aggregated. Compared with Ahn *et al.*'s construction, our scheme is efficient, and needs not compute pairings in the aggregate verification.

 Copyright ⓒ 2014 SERSC

Acknowledgements

This work was supported by Shanghai 085 Project for Municipal Universities, the Innovation Program of Shanghai Municipal Education Commission under grant No. 14YZ134 and the NSF of China under grants No. 61373006.

References

[1] D. Boneh, C. Gentry, B. Lynn and H. Shacham, "Aggregate and verifiably encrypted signatures from bilinear maps", EUROCRYPT '03, vol. 2656 of LNCS, **(2003)**, pp. 416–432.
[2] D. Boneh, B. Lynn and H. Shacham, "Short signatures from the Wail pairings", Advances in Cryptology-Asiscrypt'01, Gold Coast, Australia, LNCS 2248, Springer-Verlag, Berlin, **(2001)**, pp. 514-532.
[3] A. Lysyanskaya, S. Micali, L. Reyzin and H. Shacham, "Sequential aggregate signatures from trapdoor permutations", In EUROCRYPT '04, vol. 3027 of LNCS, **(2004)**, pp. 74–90.
[4] S. Lu, R. Ostrovsky, A. Sahai, H. Shacham and B. Waters, "Sequential aggregate signatures and multisignatures without random oracles", In EUROCRYPT '06, vol. 4004 of LNCS, **(2006)**, pp. 465–485, Full version at http://cseweb.ucsd.edu/~hovav/dist/agg-sig.pdf.
[5] B. Waters, "Efficient identity-based encryption without random oracles", Ronald Cramer, editor, Advances in Cryptology – EUROCRYPT '05, vol. 3494, **(2005)**, pp. 320–329.
[6] C. Gentry and Z. Ramzan, "Identity-based aggregate signatures", Public Key Cryptography '06, vol. 3958 of LNCS, **(2006)**, pp. 257–273.
[7] S. Hohenberger and B. Waters, "Realizing hash-and-sign signatures under standard assumptions", EUROCRYPT '09, vol. 5479 of LNCS, **(2009)**, pp. 333–350.
[8] S. Hohenberger and B. Waters, "Short and Stateless Signatures from the RSA Assumption", CRYPTO '09, vol. 5677 of LNCS, **(2009)**, pp. 654-670.
[9] S. Schage and J. Schwenk, "A New RSA-Based Signature Schme", AFRICACRYPT 2010, LNCS 6055, **(2010)**, pp. 1-15.
[10] S. Goldwasser, S. Micali and R. L. Rivest, "A digital signature scheme secure against adaptive chosen message attacks", SIAM J. Comput, vol. 17, no. 2, **(1988)**, pp. 281-308.
[11] J. H. Ahn, M. Green and S. Hohenburger, "Synchronized aggregate signatures: new definitions constructions and applications", Ehab Al-Shaer, Angelos D. Keromytis, Vitaly Shmatikov (Eds.): Proceedings of the 17th ACM Conference on Computer and Communications Security, Chicago, Illinois, USA, **(2010)** October 4-8, pp. 473-484.

Copyright ⓒ 2014 SERSC

 Copyright ⓒ 2014 SERSC