A Threshold GQ Signature Scheme

Cheng-Kang Chu, Li-Shan Liu, and Wen-Guey Tzeng
Department of Computer and Information Science
National Chiao Tung University
Hsinchu, Taiwan 30050
Email:{ckchu,gis89566,tzeng}@cis.nctu.edu.tw

Abstract

We proposed the first threshold GQ signature scheme. The scheme is unforgeable and robust against any adaptive adversary if the base GQ signature scheme is unforgeable under the chosen message attack and computing the discrete logarithm modulo a safe prime is hard. Our scheme achieve optimal resilience, that is, the adversary can corrupt up to a half of the players. As an extension of our work, we proposed a threshold forward-secure signature scheme, which is the threshold version of the most efficient forward-secure signature scheme up to now.

Keywords: threshold signature, GQ signature, forward-secure signature.

1 Introduction

A threshold cryptographic protocol involves a set of players together, who each possesses a secret share, to accomplish a cryptographic task via exchange of messages among them. Threshold cryptographic protocols provide strong security assurance and robustness against a number of malicious attackers under a threshold. For example, in a (t, n)-threshold signature scheme, as long as $t + 1$ servers agree, they can jointly produce a signature for a given message even some other servers intend to spoil such process. Also, as long as the adversary corrupts less than $t + 1$ servers, it cannot forge any valid signature.

The signature scheme proposed by Guillou and Quisquater [GQ88], called the GQ signature scheme here, are used in many cryptographic protocols, such as, in [IR01]. To our best knowledge, there are no threshold versions of this important signature scheme in the open literature. Therefore, in this paper we study the threshold signature protocol based on the GQ signature scheme. Our GQ-based threshold signature protocol is secure in the adaptive adversary model. Our scheme achieve optimal resilience, that is, the adversary can corrupt up to a half of the players. We also extend our work to the forward signature paradigm. In particular, we propose a threshold version of the optimal forward signature scheme [IR01], which is based on the GQ signature scheme.

Related work. The first general notion of efficient threshold cryptography was introduced by Desmedt [Des87]. It started many studies on threshold computation models and concrete threshold schemes based on the basic schemes, such as RSA, DSA, ElGamal, etc [CMI93, GJKR96b, CGJ$^+$99, JL00, Jar01, DF91, GJKR96a, Rab98, FMY99, Sho00, DK01]. These protocols develop many techniques for designing secure threshold protocols. We use some of them and develop new ones for our schemes.

There are two types of forward signature schemes. The first type is a general construction based on arbitrary signature schemes. It mainly uses the master public key to certify the public key of the current time period. The second type modifies the concrete signature schemes. Currently, there are three such forward-secure signature schemes [BM99, AR00, IR01], which are based on Fiat-Shamir-, 2^tth-root- and GQ-identification schemes, respectively. The GQ-based forward-secure signature scheme is the most efficient one with respect to the key size, and signing and verification time. There are threshold versions for the first two schemes [TT01, AMN01]. In this paper we propose the threshold version for the third one.

2 Preliminaries

The GQ signature scheme is as follows.

1. Key generation: let $p = 2p' + 1$ and $q = 2q' + 1$ be two safe primes and $n = pq$. The private key of a player is (n, e, s) and the corresponding public key is (n, e, v), where $v = 1/s^e \bmod n$. Note that the user need not know p and q and thus n can be used by all users.

2. Signing: let h be a publicly defined cryptographic strong hash function, such as SHA-1. Given a message M, the signer computes the signature (r, z) as follows:

   - Randomly select a number $r \in Z_n^*$ and compute $y = r^e \bmod n$ and $\sigma = h(y||M)$.
   - Compute $z = r \cdot s^\sigma \bmod n$.

3. Verification: the verifier checks whether $h(M||z^e v^\sigma \bmod n) = r$.

A *threshold signature scheme* consists of the following three components:

1. Key generation: there are two categories in generating the keys and distributing shares of them to the participated players. In the dealer model, a dealer chooses the keys and distributes the shares to the

players. In the distributed key generation model, all players together compute their key shares together.

2. Distributed signing: there are two phases: partial signature generation and signature construction. In the partial signature generation phase, the players communicate with each other and each produces a partial signature for the given message M. Then, in the signature construction, any one who has a number of valid partial signatures over a threshold can compute a valid signature for M.

3. Verification: any one can verify the validity of a signature for a message given the public key.

An *adaptive adversary* is a probabilistic polynomial time Turing machine which can corrupt players dynamically, that is, it can corrupt a player at any time during execution of the protocol. Nevertheless, the total number of players it can corrupt is under the threshold.

Two distribution ensembles $\{X_n\}$ and $\{Y_n\}$ are (computationally) *indistinguishable* if for any probabilistic polynomial-time distinguisher D and any polynomial $p(n)$, there is an integer n_0 such that for any $n \geq n_0$,

$$|\Pr[D(X_n) = 1] - \Pr[D(Y_n) = 1]| < 1/p(n).$$

The *discrete logarithm* over a safe prime problem is to solve $DLog_{\tilde{g}}\tilde{h}$ mod $\tilde{p}$ from given $(\tilde{p}, \tilde{g}, \tilde{h})$, where $\tilde{p} = 2\tilde{p}' + 1$ is prime, $\tilde{p}'$ is also prime, and $\tilde{g}$ is a generator of the quadratic subgroup $G_{\tilde{p}'}$ of $Z^*_{\tilde{p}}$. We assume that no probabilistic polynomial-time Turing machine can solve a significant portion of the input. Let I_n be the (uniform) distribution of the size-n input. Then, for any probabilistic polynomial-time Turing A and polynomial $p(n)$, there is n_0 such that for any $n \geq n_0$,

$$\Pr_{\tilde{p},\tilde{g},\tilde{h} \in I_n}[A(\tilde{p}, \tilde{g}, \tilde{h}) = DLog_{\tilde{g}}\tilde{h} \bmod \tilde{p}] < 1/p(n).$$

3 Threshold GQ signature scheme

In our threshold GQ signature scheme, the dealer generates a public/secret key pair and distributes shares of the secret key to the players. To sign a message distributively, each player produces a partial signature. If there are more than $t + 1$ valid partial signatures, we can construct the signature of the message from the valid partial signatures.

3.1 Generating keys

The *key generation* process is shown in Figure 1. Let $\{P_i\}$ be the set of l participating players and $L = l!$. There are two security parameters k_1 and

Input: security parameters k_1, k_2.
The dealer generates and distributes keys as follows:

1. Choose two $\lceil k_1/2 \rceil$-bit random primes p, q such that $p = 2p' + 1, q = 2q' + 1$, where p', q' are also primes. Let $n = pq, m = p'q'$, and g a generator of G_n.
2. Choose a random polynomial $f(x) = a_0 + a_1 x + \ldots + a_t x^t$ over Z_m of degree t. Let $s = g^{a_0} \bmod n$ be the main secret and hand $s_i = g^{f(i)} \bmod n$ to player P_i secretly.
3. Randomly choose a $(k_2 + 1)$-bit value e, such that $\gcd(e, \phi(n)) = 1$ and $\gcd(e, L^2) = 1$, where $L = l!$. Compute $v = 1/s^e \bmod n$.
4. Let $SK_i = (n, e, g, s_i)$ be the secret key of player P_i, and broadcast the public key $PK = (n, e, g, v)$.

Figure 1: TH-GQ-KeyGen: Generating keys

k_2. The dealer chooses two safe primes $p = 2p' + 1$ and $q = 2q' + 1$, each of length $\lceil k_1/2 \rceil$ bits, where p' and q' are also primes. Let $n = pq$, $m = p'q'$, Q_n the set of all quadratic residues modulo n, and g a generator of Q_n. The order of Q_n is m. Hereafter, all group computations are done in Q_n and the corresponding exponent arithmetic is done in Z_m.

The dealer then chooses a random degree-t polynomial $f(x)$ over Z_m and gives the share $s_i = g^{f(i)}$ to the player P_i. Note that the share given to player P_i is $g^{f(i)}$, instead of $f(i)$. The shared secret key is thus $s = g^{f(0)}$. The dealer then chooses a random (k_2+1)-bit value e with $\gcd(e, \phi(n)) = 1$ and $\gcd(e, L^2) = 1$ and computes $v = 1/s^e \bmod n$. In summary, the public key PK of the scheme is (n, e, g, v) and the secret share of player P_i is $SK_i = (n, e, g, s_i)$.

3.2 Signing messages

The message signing protocol consists of two phases: *distributed signing* and *signature construction*, shown in Figure 2.

3.2.1 Distributed signing

To sign a message M, all players execute INT-JOINT-EXP-RVSS, shown in Figure 7, to jointly generate a random value $y = g^{f_r(0) \cdot e}$, where $f_r(x)$ is the sharing polynomial. Each P_i gets a share $r_i = g^{f_r(i)}$. Then, each player computes $\sigma = H(y, M)$, where H is the chosen one-way hash function.

All players together run INT-JOINT-ZVSS[1] such that they share a zero-constant degree-t polynomial $f_c(x)$ and each player P_i holds a share $c_i = g^{L \cdot f_c(i)} \bmod n$, where $f_c(0) = 0$. Then, all other secret information generated

[1]INT-JOINT-ZVSS is just like INT-JOINT-RVSS in Figure 6, except that each player sets his secret a_{i0}, b_{i0} to be zero.

Signing message
Input: message M;

Distributed Signing:

1. All players perform INT-JOINT-EXP-RVSS to generate $y = g^{f_r(0)e} \bmod n$ such that each player P_i gets his share $f_r(i)$ and computes $r_i = g^{f_r(i)} \bmod n$.
2. All players compute $\sigma = H(y, M)$.
3. All players perform INT-JOINT-ZVSS such that each player P_i gets a share $f_c(i)$ and computes $c_i = g^{Lf_c(i)} \bmod n$. All secret information, except c_i, generated in this step is erased.
4. Each player P_i computes his partial signature $z_i = (r_i s_i^{\sigma})^L c_i \bmod n$.

Signature construction:

5. Compute
$$z' = \prod_{j=1}^{t+1} z_{i_j}^{\lambda_{i_j} L} \bmod n,$$
where $z_{i_1}, z_{i_2}, \ldots, z_{i_{t+1}}$ are $t+1$ valid signatures and λ_{i_j}'s are the corresponding interpolation coefficients.
6. Find integers a, b for $L^2 a + eb = 1$, and compute $z = z'^a \cdot (y/v^{\sigma})^b \bmod n$.
7. The signature of message M is (z, σ).

Verifying signature
Input: (PK, M, SIG) where $PK = (n, e, g, v), SIG = (z, \sigma)$.

1. Compute $y' = z^e v^{\sigma} \bmod n$.
2. Accept the signature if $\sigma = H(y', M)$.

Figure 2: TH-GQ-Sig & TH-GQ-Ver: Signing and verifying message

in INT-JOINT-ZVSS are erased (the erasing technique [CFGN96, CGJ+99]). Finally, each player P_i computes his partial signature $z_i = (r_i s_i^{\sigma})^L c_i \bmod n$.

3.2.2 Signature construction

To compute the signature for M, we choose $t+1$ valid partial signatures $z_{i_1}, z_{i_2}, ..., z_{i_{t+1}}$ and compute their interpolation

$$\begin{aligned} z' &= \prod_{j=1}^{t+1} z_{i_j}^{\lambda_{i_j} L} \bmod n \\ &= (g^{\sum_{j=1}^{t+1} \lambda_{i_j}(f_r(i_j)+\sigma f(i_j)+f_c(i_j))})^{L^2} \bmod n \\ &= (rs^{\sigma})^{L^2} \bmod n, \end{aligned}$$

where λ_{i_j} is the jth interpolation coefficient for the set $\{i_1, i_2, \ldots, i_{t+1}\}$. Since $\lambda_{i_j} \cdot L$ is an integer, we can compute z' even without knowing the factorization of n. Since $\gcd(L^2, e) = 1$, we find integers a and b such that $L^2a + eb = 1$ and compute the signature z for M as:

$$z = z'^a \cdot (y/v^\sigma)^b = (rs^\sigma)^{L^2a}(rs^\sigma)^{eb} = rs^\sigma \bmod n$$

Remark. Since the sharing polynomials $f(x), f_r(x)$, and $f_c(x)$ are over the exponents, the partial signature $z_i = (r_i s_i^\sigma)^L c_i$ is a share of a degree-t polynomial in the exponent. Thus, we need only $t + 1$ shares to interpolate z'.

3.3 Verifying signatures

The verification procedure is straightforward, as defined by the GQ signature scheme, shown in Figure 2.

3.4 Security analysis

Our threshold GQ signature scheme is secure against the chosen message attack in the adaptive adversary model. That is, as long as the adaptive adversary controls less than $t + 1$ players, it cannot forge a valid signature without interacting with un-corrupted players. The adversary cannot interrupt the un-corrupted players to cooperatively obtain a valid signature for a message, either.

We need a simulator $\mathsf{SIM}_{\mathsf{TH-GQ-Sig}}$ to simulate the view of execution of the $\mathsf{TH\text{-}GQ\text{-}Sig}$ scheme producing a signature (σ, z) for a message M. The simulator is shown in Figure 3.

Lemma 1 *If the adaptive adversary corrupts at most t players, its view of an execution of TH-GQ-Sig on input message M and output signature (σ, z) is the same as the view of an execution of $\mathsf{SIM}_{\mathsf{TH-GQ-Sig}}$ on input M and signature (σ, z).*

Proof. Assume that $\mathcal{B}$ is the set of corrupted players and $\mathcal{G}$ is the set of un-corrupted (honest) players up to now. In the beginning (the key generation stage), the simulator emulates the dealer to randomly assign $s_i \in Q_n$ to player P_i, $1 \le i \le l$. These s_i's remain fixed for many rounds of simulation of $\mathsf{TH\text{-}GQ\text{-}Sig}$. Let $y = z^e v^\sigma \bmod n$, which is the correct $r^e \bmod n$. Since y is distributively computed by all players in $\mathsf{TH\text{-}GQ\text{-}Sig}$, the simulator runs $\mathsf{SIM}_{\mathsf{INT-JOINT-EXP-RVSS}}$ (Figure 8) on input y to simulate the execution of $\mathsf{INT\text{-}JOINT\text{-}EXP\text{-}RVSS}$ protocol. In Step 3, the simulator runs $\mathsf{INT\text{-}JOIN\text{-}ZVSS}$ on behalf of honest players and assigns each corrupted player P_i a share $c_i = g^{f_c(i)} \bmod n$, where $f_c(x)$ has a zero constant. Now, the corrupted players P_i get s_i, r_i and c_i. Their partial signatures $z_i = (r_i s_i^\sigma)^L c_i \bmod n$ need be fixed since the adversary corrupts them. Let $\Lambda' \supseteq \mathcal{B}$ be a set of t players. We fix the partial signatures of the players in Λ'. For un-corrupted players $P_j \notin \Lambda'$, we set their partial signatures to be compatible with those

Input: message M and the corresponding signature (σ, z)
Let $\mathcal{B}$ be the set of corrupted players and $\mathcal{G}$ the set of honest players at that time.
Randomly choose $s_i \in Q_n$ for P_i, $1 \leq i \leq l$.

1. Let $y = z^e v^\sigma \bmod n$.
2. Execute $\mathsf{SIM}_{\mathsf{INT-JOINT-EXP-RVSS}}$ on input y.
3. Execute INT-JOINT-ZVSS and assign each corrupted player P_i a share $c_i = g^{f_c(i)L} \bmod n$.
4. Randomly select a set $\Lambda' \supseteq \mathcal{B}$ of t players and for each $P_j \notin \Lambda'$, compute
$$z_j^* = z^{\lambda_{j,0}L} \cdot \prod_{k \in \Lambda'} (r_k s_k^\sigma g^{f_c(k)})^{\lambda_{j,k}L} \bmod n,$$
set
$$c_j^* = z_j^* / (r_j s_j^\sigma)^L \bmod n,$$
and erase c_j.
5. Broadcast all partial signatures z_i^* for $i \in \mathcal{G}$. (Note that $z_i^* = z_i$ for $P_i \in \Lambda' - \mathcal{B}$.)

Figure 3: $\mathsf{SIM}_{\mathsf{TH-GQ-Sig}}$: Simulator of TH-GQ-Sig

in Λ', that is, the shares of any $t+1$ players result in the same signature. This is done by setting their partial signatures as

$$z_j^* = z^{\lambda_{j,0}L} \cdot \prod_{k \in \Lambda'} (r_k s_k^\sigma g^{f_c(k)})^{\lambda_{j,k}L} \bmod n,$$

where $\lambda_{j,k}$'s are the interpolation coefficients for computing the jth share from the set of shares $\{0\} \cup \{k | P_k \in \Lambda'\}$. The simulator also sets the new shares $c_j^* = z_j^* / (r_j s_j^\sigma)^L$ to the players $P_j \notin \Lambda'$ and erases the old shares c_j. These c_j^* make the un-corrupted players have the consistent relation for r_j, s_j, c_j^* and z_j^*.

We see how the simulator produces an indistinguishable distribution for the adversary:

1. The simulator runs $\mathsf{SIM}_{\mathsf{INT-JOINT-EXP-RVSS}}$ which generates y in the proper distribution.

2. The simulator performs INT-JOINT-ZVSS on behalf of honest players. This is the same as what TH-GQ-Sig does in Step 3. Thus, the distribution for c_i's is the same.

3. The partial signatures z_i's of all players are consistent since the shares of any $t+1$ players produces the right signature (σ, z) (by adjusting c_i's). Therefore, they have the right distribution.

4. The erasing technique (for c_i's) is employed. As long as the simulated distribution is indistinguishable for the adversary after it corrupts a

new player, the entire distribution is indistinguishable for the adversary after it corrupts up to t players. There is no inconsistency problem between corruptions of players.

5. The shares c_j's are adjusted to c_j^*'s for the un-corrupted players (up to now) so that even the adversary corrupts it later, the partial signature z_j^* is consistent with the possible check of equation $z_j^* = (r_j s_j^\sigma)^L c_j^*$.

In conclusion, the simulator $\mathsf{SIM}_{\mathsf{TH-GQ-Sig}}$ produces an indistinguishable distribution for the adversary, who corrupts up to t players in an adaptive way. $\square$

We now show that our threshold GQ signature scheme is secure against the adaptive adversary under the chosen message attack.

For unforgeability, let $\mathcal{O}_{sig}$ be the signing oracle that a forger (in the centralized version) queries for signatures of messages. When $\mathcal{O}_{sig}$ returns (σ, z) for a message M, the simulator, on input M and (σ, z), outputs a transcript with an indistinguishable distribution for the adaptive adversary (in the distributed version). Thus, the adversary, who engaged several executions of the TH-GQ-Sig protocol, cannot produce an additional valid signature without cooperation of un-corrupted players.

Theorem 1 *If the underlying GQ signature scheme is unforgeable under the adaptive chosen message attack, the threshold GQ signature scheme in Figures 1 and 2 is unforgeable against the adaptive adversary who corrupts up to t players.*

Proof. (sketch) Assume that the adversary $\mathcal{A}$, who controls up to t players during execution of the TH-GQ-Sig scheme and thus obtains signatures for $M_1, M_2, \ldots$, produces a valid signature for M, $M \neq M_i$ for $i \geq 1$. We construct a forger $\mathcal{F}$ to forge a signature for an un-queried message using the signing oracle $\mathcal{O}_{sig}$ for the underlying GQ signature scheme.

Let (n, e, g, v) be the public key of the underlying GQ signature scheme. This is used in the (simulated) threshold GQ signature scheme also. First, in the key generation stage $\mathcal{F}$ assigns each player P_i a random secret share $s_i \in Q_n$. Then, it simulates all players and the adversary. When the players intend to execute TH-GQ-Sig to produce a valid signature for M_i, $\mathcal{F}$ queries $\mathcal{O}_{sig}$ to obtain a signature (σ_i, z_i) and runs the simulator $\mathsf{SIM}_{\mathsf{TH-GQ-Sig}}$, on input M_i and (σ_i, z_i), to produce a transcript T_i with right distribution (by Lemma 1) for $\mathcal{A}$. Then, $\mathcal{F}$ simulates $\mathcal{A}$, on input of these transcripts T_i's, to produce a valid signature (σ, z) for a new message M, $M \neq M_i, i \geq 1$. Thus, the underlying GQ signature is not unforgeable under the chosen message attack, which is a contradiction. $\square$

Theorem 2 *If computing the discrete logarithm modulo a safe prime is hard, the TH-GQ-Sig scheme in Figure 2 is robust against the adaptive adversary who corrupts up to t players.*

Proof. (Sketch) If there is an adversary $\mathcal{A}'$ who participates the TH-GQ-Sig scheme on the input messages that it selected such that the honest players fail to generate a valid signature for a given message, then we can construct an extractor $\mathcal{E}$ to solve the discrete-log problem modulo a safe prime. That is, on input $(\tilde{p}, \tilde{g}, \tilde{h})$, $\mathcal{E}$ can compute $\mathrm{DLog}_{\tilde{g}}\tilde{h} \bmod \tilde{p}$ as follows, where $\tilde{p} = 2\tilde{p}' + 1$.

First, $\mathcal{E}$ lets the dealer generate the related keys as usual except that the dealer chooses (1) $p = \tilde{p}$ (and thus $p' = \tilde{p}'$) (2) $g = \tilde{g}^2 \bmod n$. Without loss of generality, we assume that $g \not\equiv 1 (\mathrm{mod} q)$. Since $\tilde{g}$ is the generator of $G_{\tilde{p}'}$ and $g \bmod q$ is a generator for $G_{\tilde{q}'}$, g is a generator of Q_n. Then $\mathcal{E}$ simulates h-generation protocol with $\mathcal{A}'$ and outputs $h = \tilde{h}$ [Jar01]. Using the instance (g, h), $\mathcal{E}$ performs TH-GQ-Sig with $\mathcal{A}'$ on behalf of the honest players. Now, we show that if $\mathcal{A}'$ hinders the signing protocol from producing a valid signature, $\mathcal{E}$ can compute $\mathcal{D}$=$\mathrm{DLog}_g h \bmod n$, and then outputs $\mathrm{DLog}_{\tilde{g}}\tilde{h} = 2\mathcal{D} \bmod \tilde{p}$.

Let us consider where the protocol may fail to produce the valid signature. First, in executing the INT-JOINT-RVSS scheme (or INT-JOINT-ZVSS, see Figure 6), if a corrupted player P_i distributes his shares $(f_i(j), f_i'(j))$, $1 \le j \le n$, that pass the verification equation (1), but do not lie on a t-degree polynomial, the extractor $\mathcal{E}$ can solve the system of $t+2$ linearly independent equations of the form $c_0 + c_1 j + c_2 j^2 + \ldots + c_t j^t = f_i(j) + \mathcal{D} f_i'(j)$ with $t+2$ unknown variables $c_0, c_1, \ldots, c_t$ and $\mathcal{D}$. Then, the extractor outputs $\mathcal{D}$.

Another situation that $\mathcal{A}'$ may cheat is on the zero-knowledge proof in executing INT-JOINT-EXP-RVSS. If the corrupted player P_i broadcasts $(A_i^*, B_i^*) \neq (g^{a_{i0}e}, h^{b_{i0}e})$ in Step 2, $\mathcal{E}$ extracts $\mathcal{D}$ =$\mathrm{DLog}_g h$ as follows. Assume that $A_i^* = g^{a'} \bmod n$ and $B_i^* = h^{b'} \bmod n$. After executing Steps 2a-2c, $\mathcal{E}$ gets $R_i = r_i + da'e$ and $R_i' = r_i' + db'e$. Then $\mathcal{E}$ rewinds $\mathcal{A}'$ to run Steps 2b-2c again. This gives $\mathcal{E}$ another two equations $R_i^* = r_i + d^* a'e$ and $R_i'^* = r_i' + d^* b'e$. As long as $d \neq d^*$ (the probability of equality is negligible), $\mathcal{E}$ can solve the equations and get the four unknown variables r_i, r_i', a', b'. Since $\mathcal{E}$ knows the value m, the extractor computes $\mathcal{D}$ from $a_{i0} + \mathcal{D} b_{i0} = a' + \mathcal{D} b' \bmod m$. □

3.5 Achieving optimal resilience

The protocols presented up to now have not yet achieved optimal resilience since in Step 5 of TH-FIG-Sig we need find $t+1$ valid partial signatures. Also, Step 2(b) of INT-JOINT-EXP-RVSS need reconstruct the challenge d. With the simple majority vote, the threshold t is $n/3$ at most.

We describe how to modify them to achieve optimal resilience ($n \geq 2t+1$). The main difference is that each player proves correctness of its partial signature when it is issued. For this proof, in the key generation stage the dealer shares $f(i)$ (instead of $g^{f(i)}$) to player P_i. In addition, the dealer shares another random polynomial $f'(x)$ and broadcasts the coefficient references of $f(x)$ and $f'(x)$ in the unconditionally-secure form, that is, $A_i = g^{f(i)L} h^{f'(i)L}$. When signing a partial signature, each player presents a non-interactive zero-knowledge proof of knowledge of $f(i), f_r(i)$ and $f_c(i)$. Therefore, in signature construction one can verify validity of partial signatures.

To resolve the case for Step 2(b) of INT-JOINT-EXP-RVSS, we can use the non-interactive zero-knowledge proof technique similarly. We can also replace the INT-JOINT-RVSS of jointly generating the challenge d with a coin-flip protocol (e.g. in additive form).

4 Threshold forward-secure signature

Based on our threshold GQ signature scheme and Itkis and Reyzin's forward signature scheme [IR01], we construct a threshold forward-secure signature scheme TH-FS-GQ.

4.1 Key generation

The key generation process is in Figure 4. Let T be the total number of time periods. The first three steps are the same as those of TH-GQ-Sig. Each player P_i gets s_i as his initial share. The dealer chooses primes e_j, $1 \le j \le T$, in the range between $2^{k_2}(1 + (j-1)/T)$ and $2^{k_2}(1 + j/T)$. That is, we divide the interval between 2^{k_2} and 2^{k_2+1} into T buckets, and choose each prime e_j from the j-th bucket. The public key is

$$v = 1/s^{e_1 e_2 \cdots e_T} \bmod n.$$

At time period j, $1 \le j \le T$, the player P_i, $1 \le i \le n$, holds two secrets:

$$\begin{aligned} s_{ij} &= s_i^{e_1 \cdots e_{j-1} e_{j+1} \cdots e_T} \bmod n, and \\ t_{ij} &= s_i^{e_1 e_2 \cdots e_j} \bmod n \end{aligned}$$

where s_{ij} is the signing share and t_{ij} is an auxiliary secret for share update.

In the beginning, the dealer sends the player P_i two shares s_{i1} and t_{i1}. Hence, the secret key for player P_i at time period 1 is $SK_{i1} = (1, T, n, s_{i1}, t_{i1}, e_1)$ and the system public key is $PK = (n, v, T)$.

4.2 Key update

To compute the keys of the $j{+}1$th time period, each player P_i computes the exponents $e_{j+1}, e_{j+2}, \ldots, e_T$ for evolving the secret[2]. Then, P_i computes its

[2]We can also take all e_j's as the part of secret key, but this would increase the storage space linear in T. Therefore, we only put e_j into the secret key for each message signing, and regenerate others for key update. Note that the dealer chooses these primes by a deterministic algorithm such that players can regenerate $e_{j+1}, \ldots, e_T$ based on e_j.

Generating Keys
Input: security parameters (k_1, k_2, T).
The dealer generates and distributes keys as follows:

1. Choose two $\lceil k_1/2 \rceil$-bit random primes p, q such that $p = 2p' + 1, q = 2q' + 1$, where p', q' are also primes. Let $n = pq, m = p'q'$.
2. Choose a random value $g \in Q_n$ and a random polynomial $f(x) = a_0 + a_1 x + \ldots + a_t x^t$ over Z_m of degree t. Let $s = g^{a_0} \bmod n$ be the main secret and hand $s_i = g^{f(i)} \bmod n$ to player P_i secretly.
3. For $j = 1, \ldots, T$, choose primes e_j such that $2^{k_2}(1 + (j-1)/T) \le e_j < 2^{k_2}(1 + j/T)$.
4. Compute the public value $v = 1/s^{e_1 \cdot \ldots \cdot e_T} \bmod n$.
5. Compute the first period share $s_{i1} = s_i^{e_2 \cdot \ldots \cdot e_T} \bmod n$ and $t_{i1} = s_i^{e_1} \bmod n$ for each player P_i.
6. Send the secret share $SK_{i1} = (1, T, n, s_{i1}, t_{i1}, e_1)$ to each player P_i at time period 1 and publish the public key $PK = (n, v, T)$.

Updating Keys
Input: $SK_{ij} = (j, T, n, s_{ij}, t_{ij}, e_j)$ for each player P_i.
Each player P_i updates his share from time period j to $j + 1$.

1. Each player regenerates $e_{j+1}, \ldots, e_T$.
2. Each player P_i updates his share by computing:
$$s_{i(j+1)} = t_{ij}^{e_{j+2} \cdot \ldots \cdot e_T} \bmod n; \quad t_{i(j+1)} = t_{ij}^{e_{j+1}} \bmod n$$
3. The new secret share of player P_i at time period $j + 1$ is $SK_{i(j+1)} = (j + 1, T, n, s_{i(j+1)}, t_{i(j+1)}, e_{j+1})$.

Figure 4: TH-FS-GQ - Generating and Updating Keys

signing share $s_{i(j+1)} = t_{ij}^{e_{j+2}e_{j+3}\cdots e_T} \bmod n$ and key evolving secret $t_{i(j+1)} = t_{ij}^{e_{j+1}} \bmod n$. Thus, P_i's signing share in the $j+1$th period is

$$SK_{i(j+1)} = (j + 1, T, n, s_{i(j+1)}, t_{i(j+1)}, e_{j+1}).$$

4.3 Signing messages

The signing protocol, in Figure 5, is quite similar to TH-GQ-Sig except that we need time parameters and replace the values e and s_i with e_j and s_{ij} for time period j. Note that the time period j and the exponent e_j are put into the hash function in computing σ. Since the verifier doesn't know e_j exactly, the hashing ensures that the signer can't choose e_j arbitrarily after seeing the challenge σ. Finally, the signature for the message M is (z, σ, j, e_j).

Signing Message
Input: message M, public value n
For time period j, players jointly sign the message as follows:
Distributed Signing

1. All players perform INT-JOINT-EXP-RVSS to generate $y = g^{f_r(0)e_j} \bmod n$. Each player P_i sets his share to $f_r(i)$, and computes $r_i = g^{f_r(i)} \bmod n$.
2. All players compute $\sigma = H(j, e_j, y, M)$.
3. All players perform INT-JOINT-ZVSS. Each player P_i gets a share $f_c(i)$ and computes $c_i = g^{Lf_c(i)} \bmod n$. All secret information except c_i is erased.
4. Each player P_i computes his partial signature $z_i = (r_i s_{i,j}^{\sigma})^L c_i \bmod n$.

Constructing Signature

5. Compute $z' = \prod_{i \in S} z_i^{\lambda_i L} \bmod n$, where S is any subset of $t+1$ players.
6. Find integers a, b such that $L^2 a + e_j b = 1$, and compute $z = z'^a \cdot (y/v^{\sigma})^b \bmod n$.
7. The signature of message M is $SIG = (z, \sigma, j, e_j)$.

Verifying Signature
Input: (PK, M, SIG, k_2) where $PK = (n, v, T), SIG = (z, \sigma, j, e')$, k_2 is the security parameter.
The verifier accepts the signature SIG at time period j if the following equations hold:

1. $2^{k_2} \leq e' < 2^{k_2}(1 + j/T)$.
2. $\sigma = H(j, e', y', M)$, where $y' = z^{e'} v^{\sigma} \bmod n$.

Figure 5: TH-FS-GQ - Signing and Verifying

4.4 Signature verification

In order to have a constant size for the public key, the time period exponents $e_1, e_2, \ldots, e_T$ are computed when necessary. They are not part of the public key. Since the verifier gets e' from the signature, it need make sure that it is in between $2^{k_2}(1 + (j-1)/T)$ and $2^{k_2}(1 + j/T)$. Once e' is verified, the remaining verification is the same as that in TH-GQ-Sig.

References

[AMN01] Michel Abdalla, Sara Miner, and Chanathip Namprempre. Forward-secure threshold signature schemes. In *Proceedings of the CT-RSA conference*, pages 441–456. Springer-Verlag, 2001.

[AR00] Michel Abdalla and Leonid Reyzin. A new forward-secure digital signature scheme. In *Proceedings of Advances in Cryptology - ASIACRYPT 2000*, volume 1976 of *LNCS*, pages 116–129. Springer-Verlag, 2000.

[BM99] Mihir Bellare and Sara K. Miner. A forward-secure digital signature scheme. In *Proceedings of Advances in Cryptology - CRYPTO '99*, volume 1666 of *LNCS*, pages 431–448. Springer-Verlag, 1999.

[CFGN96] Ran Canetti, Uriel Feige, Oded Goldreich, and Moni Naor. Adaptively secure multi-party computation. In *Proceedings of the 28th Annual ACM Symposium on the Theory of Computing (STOC '96)*, pages 639–648. ACM, 1996.

[CGJ+99] Ran Canetti, Rosario Gennaro, Stanislaw Jarecki, Hugo Krawczyk, and Tal Rabin. Adaptive security for threshold cryptosystems. In *Proceedings of Advances in Cryptology - CRYPTO '99*, volume 1666 of *LNCS*, pages 98–115. Springer-Verlag, 1999.

[CMI93] Manuel Cerecedo, Tsutomu Matsumoto, and Hideki Imai. Efficient and secure multiparty generation of digital signatures based on discrete logarithms. *IEICE Trans. Fundamentals*, E76-A(4):532–545, 1993.

[Des87] Yvo Desmedt. Society and group oriented cryptography: A new concept. In *Proceedings of Advances in Cryptology - CRYPTO '87*, volume 293 of *LNCS*, pages 120–127. Springer-Verlag, 1987.

[DF91] Yvo Desmedt and Yair Frankel. Shared generation of authenticators and signatures. In *Proceedings of Advances in Cryptology - CRYPTO '91*, volume 576 of *LNCS*, pages 457–469. Springer-Verlag, 1991.

[DK01] Ivan Damgård and Maciej Koprowski. Practical threshold rsa signatures without a trusted dealer. In *Proceedings of Advances in Cryptology - EUROCRYPT 2001*, volume 2045 of *LNCS*, pages 152–165. Springer-Verlag, 2001.

[FGMY97] Yair Frankel, Peter Gemmell, Philip D. MacKenzie, and Moti Yung. Optimal-resilience proactive public-key cryptosystems. In *Proceedings of 38th Annual Symposium on Foundations of Computer Science (FOCS '97)*, pages 384–393. IEEE, 1997.

[FMY98] Yair Frankel, Philip D. MacKenzie, and Moti Yung. Robust efficient distributed rsa-key generation. In *Proceedings of the 30th Annual ACM Symposium on the Theory of Computing (STOC '98)*, pages 663–672. ACM, 1998.

[FMY99] Yair Frankel, Philip D. MacKenzie, and Moti Yung. Adaptively-secure optimal-resilience proactive rsa. In *Proceedings of Advances in Cryptology - ASIACRYPT '99*, volume 1716 of *LNCS*, pages 180–194. Springer-Verlag, 1999.

[GJKR96a] Rosario Gennaro, Stanislaw Jarecki, Hugo Krawczyk, and Tal Rabin. Robust and efficient sharing of rsa functions. In *Proceedings of Advances in Cryptology - CRYPTO '96*, volume 1109 of *LNCS*, pages 157–172. Springer-Verlag, 1996.

[GJKR96b] Rosario Gennaro, Stanislaw Jarecki, Hugo Krawczyk, and Tal Rabin. Robust threshold DSS signatures. In *Proceedings of Advances in Cryptology - EUROCRYPT '96*, volume 1070 of *LNCS*, pages 354–371. Springer-Verlag, 1996.

[GJKR99] Rosario Gennaro, Stanislaw Jarecki, Hugo Krawczyk, and Tal Rabin. Secure distributed key generation for discrete-log based cryptosystems. In *Proceedings of Advances in Cryptology - EUROCRYPT '99*, volume 1592 of *LNCS*, pages 295–310. Springer-Verlag, 1999.

[GQ88] Louis C. Guillou and Jean-Jacques Quisquater. A "paradoxical" indentity-based signature scheme resulting from zero-knowledge. In *Proceedings of Advances in Cryptology - CRYPTO '88*, volume 403 of *LNCS*, pages 216–231. Springer-Verlag, 1988.

[IR01] Gene Itkis and Leonid Reyzin. Forward-secure signatures with optimal signing and verifying. In *Proceedings of Advances in Cryptology - CRYPTO 2001*, volume 2139 of *LNCS*, pages 332–354. Springer-Verlag, 2001.

[Jar01] Stanislaw Jarecki. *Efficient Threshold Cryptosystems*. PhD thesis, MIT, 2001.

[JL00] Stanisław Jarecki and Anna Lysyanskaya. Adaptively secure threshold cryptography: Introducing concurrency, removing erasures. In *Proceedings of Advances in Cryptology - EUROCRYPT 2000*, volume 1807 of *LNCS*, pages 221–242. Springer-Verlag, 2000.

[Ped91] Torben P. Pedersen. A threshold cryptosystem without a trusted party. In *Proceedings of Advances in Cryptology - EUROCRYPT '91*, volume 547 of *LNCS*, pages 522–526. Springer-Verlag, 1991.

[Rab98] Tal Rabin. A simplified approach to threshold and proactive rsa. In *Proceedings of Advances in Cryptology - CRYPTO '98*, volume 1462 of *LNCS*, pages 89–104. Springer-Verlag, 1998.

[Sha79] Adi Shamir. How to share a secret. *Communications of the ACM*, 22(11):612–613, 1979.

[Sho00] Victor Shoup. Practical threshold signatures. In *Proceedings of Advances in Cryptology - EUROCRYPT 2000*, volume 1807 of *LNCS*, pages 207–220. Springer-Verlag, 2000.

[TT01] Zhi-Jia Tzeng and Wen-Guey Tzeng. Robust forward signature schemes with proactive security. In *Proceedings of the Public-Key Cryptography (PKC '01)*, pages 264–276. Springer-Verlag, 2001.

Input: (n, g, h), where n is the product of two large primes

1. Each player P_i chooses two random polynomials of degree t:

$$f_i(x) = a_{i0} + a_{i1}x + \ldots + a_{it}x^t, \quad f_i'(x) = b_{i0} + b_{i1}x + \ldots + b_{it}x^t$$

   where

   (a) $a_{i0} = L^2 s, s \in_R \{0, \ldots, \lfloor n/4 \rfloor - 1\}$.

   (b) $b_{i0} = L^2 s', s' \in_R \{0, \ldots, n^2(\lfloor n/4 \rfloor - 1)\}$.

   (c) $a_{ik}, b_{ik} \in_R \{0, L, 2L, \ldots, L^3 n^2\}$, $k = 1, \ldots, t$.

2. Each player P_i broadcasts $C_{ik} = g^{a_{ik}} h^{b_{ik}} \bmod n$, $1 \leq k \leq t$, and hands $f_i(j), f_i'(j)$ to player P_j, $j \in \{1, \ldots, l\}$.
3. Each player P_j verifies his shares received from each other P_i by checking:

$$g^{f_i(j)} h^{f_i'(j)} \equiv \prod_{k=0}^{t} C_{ik}^{j^k} \bmod n \tag{1}$$

   If the check fails for an index i, P_j broadcasts a *complaint* message against P_i.

4. Each player P_i who received a complaint from player P_j broadcasts the corresponding shares $f_i(j), f_i'(j)$.
5. Each player marks as *disqualified* any player that
   - received more than t complaints, or
   - answered to a complaint with values that falsify Eq. 1.
6. Each player then builds a common set of non-disqualified players $QUAL$. The secret x is now defined as $x = \sum_{i \in QUAL} a_{i0}$, and each player P_i holds a share $x_i = \sum_{j \in QUAL} f_j(i)$.

Figure 6: INT-JOINT-RVSS

A INT-JOINT-RVSS protocol

The INT-JOINT-RVSS protocol is similar to the JOINT-RVSS protocol, except that we use unconditionally-secure VSS over integers [FGMY97, FMY98, FMY99] instead.

B INT-JOINT-EXP-RVSS protocol

In addition to INT-JOINT-RVSS, players also need to share a random value on the exponent in some applications. For example, the distributed key generation for discrete-log based cryptosystem [Ped91, GJKR99, CGJ+99] shares the secret key x and outputs the public key $y = g^x \bmod p$. Our INT-JOINT-EXP-RVSS (Figure 7) is based on an adaptively-secure distributed key generation protocol [CGJ+99] except for the composite modulus and an additional constant exponent. In the protocol, players first jointly perform INT-JOINT-RVSS to share a random secret x. To compute $y = g^{xe} \bmod n$,

Input: (n, e, g, h), where n is the product of two large primes

1. Each player P_i performs INT-JOINT-RVSS and gets the following secret outputs:
   - Polynomials $f_i(x) = a_{i0} + a_{i1}x + \ldots + a_{it}x^t, f_i'(x) = b_{i0} + b_{i1}x + \ldots + b_{it}x^t$ he generated randomly.
   - The shares $f_j(i), f_j'(i)$ sent from player j, where $j = 1, \ldots, l$.
   - The share $f(i) = \sum_{j \in QUAL} f_j(i)$ of the secret x.

   The public outputs are $C_{ik} = g^{a_{ik}} h^{b_{ik}} \bmod n$ where $i = 1, \ldots, l$, $k = 0, \ldots, t$, and the set $QUAL$ of non-disqualified players.
2. Each player $P_i, i \in QUAL$ broadcasts $A_i = g^{a_{i0}e} \bmod n$ and $B_i = h^{b_{i0}e} \bmod n$ such that $A_i B_i = C_{i0}^e$, and prove the knowledge of $a_{i0}e, b_{i0}e$:
   (a) P_i chooses $r_i, r_i' \in_R \{0, \ldots, \lfloor n/4 \rfloor - 1\}$ and broadcasts $T_i = g^{r_i} \bmod n$, $T_i' = h^{r_i'} \bmod n$.
   (b) All players jointly execute INT-JOINT-RVSS and then publicly reconstruct the random secret d.
   (c) Each player P_i broadcasts $R_i = r_i + d \cdot a_{i0}e$ and $R_i' = r_i' + d \cdot b_{i0}e$.
   (d) Each player P_j checks that $g^{R_i} \equiv T_i \cdot A_i^d \bmod n$ and $h^{R_i'} \equiv T_i' \cdot B_i^d \bmod n$ for $i = 1, \ldots, l$. If the check fails for some index i, P_j complains against P_i.
3. For each player P_i receives more than t complaints, P_j broadcasts $f_i(j)$. All players reconstruct a_{i0} and compute $A_i = g^{a_{i0}e} \bmod n$.
4. All players then compute $y = \prod_{j \in QUAL} A_j \bmod n$

Figure 7: INT-JOINT-EXP-RVSS

each player P_i broadcasts $A_i = g^{a_{i0}e}, B_i = g^{b_{i0}e}$ where $\sum_{i \in QUAL} a_{i0} = x$, and proves the knowledge of $a_{i0}e, b_{i0}e$ by simultaneous proof technique [CGJ+99, Jar01].

We also provide a simulation for INT-JOINT-EXP-RVSS in Figure 8. On input $y = g^{xe} \bmod n$, the simulator constructs the same adversarial view as in the real protocol running. The simulator performs most steps on behalf of the un-corrupted players except for a random selected player P_u, which broadcasts A_u^* to fit the input y rather than computes A_u as others. For the later proof, the simulator chooses a challenge d^* and computes the appropriate commitment for P_u in advance. Then, let all players reconstruct the challenge d^*, P_u will pass the verification. However, since the adaptive adversary corrupts players dynamically, if P_u is corrupted in the simulation, the simulator rewinds to the step 2 and executes all the following steps again.

Input: the result value y and parameters (n, e, g, h)

1. Perform INT-JOINT-RVSS on behalf of honest players.
2. Choose an uncorrupted player P_u, and do the following computation:
   - Compute $A_i = g^{a_{i0}e} \bmod n$ and $B_i = h^{b_{i0}e} \bmod n$ for $i \in QUAL \backslash \{u\}$.
   - Set $A_u^* = y \cdot \prod_{i \in QUAL \backslash \{u\}} A_i^{-1} \bmod n$ and $B_u^* = C_{u0}^e / A_u^* \bmod n$.
   - Choose $d^*, R_u, R'_u \in_R \{0, \ldots, \lfloor n/4 \rfloor - 1\}$ and set $T_u^* = g^{R_u} \cdot (A_u^*)^{-d^*} \bmod n$, $T_u'^* = g^{R'_u} \cdot (B_u^*)^{-d^*} \bmod n$
3. Broadcast A_i for player P_i,$i \in QUAL \backslash \{u\}$ and A_u^* for player P_u.
4. Perform Step 2a in the protocol on behalf of each player P_i,$i \in QUAL \backslash \{u\}$, and broadcast T_u^* and $T_u'^*$ for player P_u.
5. Perform INT-JOINT-RVSS on behalf of honest players as Step 2b in the protocol, and each P_i gets a share d_i. Pick a random t-degree polynomial $f_d^*(x)$ over integers such that $f_d^*(0) = d^*, f_d^*(i) = d_i$ for $i \in \mathcal{B}$. Erase all other secret information generated in INT-JOINT-RVSS.
6. Broadcast $f_d^*(i)$ for each honest player P_i.
7. Broadcast R_i, R'_i computed as Step 2c for player P_i, $i \in \mathcal{G} \backslash \{u\}$. Broadcast $R_u^*, R_u'^*$ for player P_u.
8. Verify the proof as in the protocol. If players in $\mathcal{B}$ receive more than t complaints, all other players reconstruct their secrets.
9. Erase all secret information except $f(i)$.

Figure 8: $\mathsf{SIM}_{\mathsf{INT-JOINT-EXP-RVSS}}$