ISSN 1000-9825, CODEN RUXUEW
Journal of Software, Vol.18, No.1, January 2007, pp.168−176
DOI: 10.1360/jos180168
© 2007 by *Journal of Software*. All rights reserved.

E-mail: jos@iscas.ac.cn
http://www.jos.org.cn
Tel/Fax: +86-10-62562563

基于分布式密钥生成方法的 RSA 可验证签名分享设计*

吕克伟+

(信息安全国家重点实验室(中国科学院 研究生院),北京 100049)

RSA Verifiable Signature Sharing Scheme Based on Secure Distributed Key Generations

LÜ Ke-Wei+

(State Key Laboratory of Information Security (Graduate School, The Chinese Academy of Sciences), Beijing 100049, China)

+ Corresponding author: Phn: +86-10-88256432 ext 62, Fax: +86-10-88258713, E-mail: conwaylu@tom.com

Lü KW. RSA verifiable signature sharing scheme based on secure distributed key generations. *Journal of Software*, 2007,18(1):168−176. http://www.jos.org.cn/1000-9825/18/168.htm

Abstract: This paper studies the Verifiable Signature Sharing ($V\Sigma S$) introduced by Franklin and Reiter, which enables the recipient of a signature to share it among *n* proxies so that a subset of them can reconstruct it later. By the use of secure distributed key generation based on discrete-log, threshold cryptosystems and verifiable secret sharing scheme, new protocols for RSA $V\Sigma S$ are presented. The protocols are efficient and provable secure and can tolerate the malicious behavior of up to half of the proxies.

Key words: signature; verifiable secret sharing scheme; protocol; threshold cryptosystem

摘 要: 主要研究由 Franklin 和 Reiter 提出的可验证签名分享($V\Sigma S$).它可以允许一个签名的接受者在 *n* 个代理之间分享该签名,使得代理者的一些子集以后可以重构该签名.利用安全的分布式密钥生成方式、门限密码系统以及可验证秘密分享,给出了一个 RSA $V\Sigma S$ 的新协议.该协议是有效的、可证安全的,并且可以容忍至多一半代理的恶意行为.

关键词: 签名;可验证秘密分享;协议;门限密码系统
中图法分类号: TP309 文献标识码: A

1 Introduction

A $V\Sigma S$ protocol, which was introduced by Franklin and Reiter[1], is divided into *sharing* phase and *recovering* phase. At the end of sharing phase, each proxy can verify that a valid signature for the given document can be reconstructed. At the end of recovering phase, such signature is reconstructed no matter what a malicious subset of proxies may do. It could be widely applied in cash escrow, secure distributed auction and distributed cryptosystems etc. For the instance of a secure distributed auction, bidders at an auction may be required to verifiably share

* Supported by the National High-Tech Research and Development Plan of China under Grant No.2006AA01Z434 (国家高技术研究发展计划(863)); the President's Foundation of Graduate School, the Chinese Academy of Sciences under Grant No.yzjj2003010 (中国科学院研究生院校长基金)

Received 2004-09-30; Accepted 2005-07-28

signatures on checks for the amount of their bids. In this way, it will be impossible for the winner of the bid to default (since the proxies can reconstruct his check), while the payments of the loser will never be recovered. Some efficient protocols were given for RSA, Rabin, ElGamal, Schnorr and DSS signatures respectively. But some of them were broken later. In Ref.[2], based on the key generation protocol from Feldman's verifiable secret sharing protocol (**VSS**) and threshold cryptosystems as we will show later, the new protocols for these signatures were presented, which were efficient and provably secure and can tolerate a malicious sharer and the malicious behavior of up to half of the proxies during sharing and reconstruction time. But Feldman-VSS assumes that the dealer is never to be corrupted by the attacker. A distributed solution to this problem is the run of n parallel executions of the Feldman-VSS as follows, called **Joint-Feldman**: For prime p and q with $q|p-1$, each player P_i selects a random secret $z_i \in Z_q$ and shares it among the n players using Feldman-VSS. This defines the set $QUAL$ of players that share their secrets properly. The secret key x is set to be the sum of the properly shared secrets and each player can compute his share of x by locally summing up the shares he received. The public key y can be computed as the product of the public values $y_i = g^{z_i} (\bmod p)$. But **Joint-Feldman** was insecure (see Ref.[3]).

Our contribution is that we first modify the secure distributed key generation protocol to work over a composite modulus, then construct a threshold cryptosystem and take advantage of the cryptosystem to obtain a new RSA $V\Sigma S$ protocol, which is efficient and provable secure and can tolerate the malicious behavior of up to half of the proxies during sharing and reconstruction. The rest of the paper is organized as follows: In Section 2, we present the communication and adversarial models and some primitive tools. In Section 3, we present a new ElGamal-based threshold cryptosystem over a composite modulus. In Section 4, we give a new RSA $V\Sigma S$ scheme.

2 Preliminaries

2.1 The model

We assume there are three entities: the signer (usually called Bob), the recipient (Alice) and a set of n proxies, $P_1,\dots,P_n$. The $V\Sigma S$ protocol will be between Alice and the proxies and must not involve Bob. Each proxy P_i has a opened value, say i, to show his identity. We assume that Alice and the proxies are connected by a full network of the private channels and a broadcast channel. All communications are synchronous. Moreover, there exists a static adversary $\boldsymbol{A}$ who can corrupt Alice and at most t of the proxies. Once corrupting one, $\boldsymbol{A}$ can read his memory and cause him to deviate arbitrarily from the protocol. The computational power of the adversary is specified by PPT (probabilistic polynomial time) Turing machine.

2.2 Tools and cryptographic assumptions

In the following, we assume $N>>n$ to be a composite, product of two large safe primes p and q. We say p and q are safe, if there exist two primes p' and q' such that $p=2p'+1$ and $q=2q'+1$. We denote with $\phi(N)=(p-1)(q-1)=4p'q'$ the order of multiplicative group Z_N^* of the integers modulo N, relatively prime to N.

Lemma 2.1. Let $N=pq$, where $p<q$, $p=2p'+1$, $q=2q'+1$, and p,q,p',q' are all prime numbers. Then, given an element $\omega \in Z_N^* \setminus \{-1,1\}$ such that $ord(\omega)<p'q'$, either $gcd(\omega-1,N)$ or $gcd(\omega+1,N)$ is a prime factor of N.

By the lemma, we can assume in our protocols that any value ω found by a player who does not know the factorization of N must be of order at least $p'q'$ except for $\{-1,1\}$. And given an element ω of $ord(\omega)\in\{p'q',2p'q'\}$, then $m^4\in\langle\omega\rangle$ for any $m \in Z_N^*$. Let G_0 be a random element in Z_N^*, $G = G_0^{L^3} (\bmod N)$, where $L=n!$. Then G has order $p'q'$ since L is even.

Pederson-VSS over Z_N. Let G be an element in Z_N^* as above, and H be a random element in the subgroup generated by G. It is assumed that the adversary cannot find $\log_G H$. Similar to that in Ref.[3], the dealer will share the secret σ over the integers since he does not know $\phi(N)$. The coefficients of the sharing polynomial must be chosen large enough to statistically hide information. Given a secret $\sigma\in[-N^2,\ldots,N^2]$, the dealer chooses at randomtwo polynomials $f(z)=\sigma+\sum_{j=1}^{t}a_j z^j$ and $f'(z)=\sum_{j=0}^{t}b_j z^j$ with coefficients in $[-L^2N^3,\ldots,L^2N^3]$, and gives player P_i a share $\sigma_i=f(i)$ and the value $\tau_i=f'(i)$. Then he commits to each coefficient of the polynomials f and f' as follows: He broadcasts the values $\beta_j=G^{a_j}H^{b_j}(\bmod N)$. This allows the players to check the received shares by verifying that $\prod_j \beta_j^{i^j}=G^{\sigma_i}H^{\tau_i}(\bmod N)$. At reconstruction time, the players are required to reveal both σ and τ, and the above equation is used to validate the shares. Note that the value of the secret is unconditionally protected since the only value opened is $\beta_0=G^{\sigma}H^{b_0}(\bmod N)$. Similar to the proof of Feldman's **VSS** (see Ref.[1]), we have the following result on **Pederson-VSS** over Z_N.

Lemma 2.2. **Pederson-VSS** over Z_N is a **VSS** of fault-tolerance t for any t, n such that $n>2t$.

Decisional Diffie-Hellman Assumption (**DDH**). Let N be as above, G a random element of Z_N^* and $\mathfrak{I}=\langle G\rangle$. Consider the two probability distributions on $\mathfrak{I}\times\mathfrak{I}\times\mathfrak{I}$ defined as $DH=(G^a,G^b,G^{ab})(\bmod N)$ and $\mathfrak{R}=(G^a,G^b,G^c)(\bmod N)$ for $a,b,$ and c chosen randomly and uniformly in Z_N. We assume that the two distributions are computationally indistinguishable. It is obvious that this assumption relies on the hardness of computing discrete-logs.

ElGamal over a composite. We are going to use the following variation of ElGamal encryption scheme[4,5] over a composite modulus. The public encryption key is $EK=(N,G_0,G,Y)$, where N, G_0 and G are as the preceding. $Y=G^X(\bmod N)$ where $X\in_R Z_N$ is the secret decryption key. A message M is encrypted under EK by choosing a random $K\in_R Z_N$ and computing $A=G_0^K(\bmod N)$ and $B=Y^K M(\bmod N)$. The ciphertext is the pair (A,B). Decryption of a pair (A,B) is computed by taking $M=B/A^{XL^3}(\bmod N)$.

2.3 Verifiable signature sharing

$V\Sigma S$ consists of a pair of protocols ($\Sigma Share$,$\Sigma Recover$) for Alice and the proxies. The input of $\Sigma Share$ for all the players is a message m and the public verification key **VK** of the signer. The secret input for Alice is a signature S of m under the signer's key. The output of $\Sigma Share$ for each proxy P_i is a value S_i, which can assume the special value $S_i=\omega$ denoting that the proxy has rejected the sharing. The protocol $\Sigma Recover$ is then run on the output of $\Sigma Share$ by the proxies.

Definition 2.1. We say that $V\Sigma S$=($\Sigma Share$,$\Sigma Recover$) is a verifiable signature sharing protocol with fault -tolerance t if, for any adversary $\boldsymbol{A}$ that can corrupt Alice and at most t proxies, the following conditions are met:

- **Completeness**: If Alice is not corrupted, then the output of $\Sigma Share$ is a signature S on m under the signer's key **VK**.
- **Soundness**: If a good proxy P_i outputs $S_i=\omega$ at the end of $\Sigma Share$, then each good proxy P_j outputs $S_j=\omega$. If $S_i\neq\omega$ for good proxies, then the output of $\Sigma Recover$ is a signature S on m under the signer's key **VK**.
- **Security**: Define the view $\boldsymbol{V}$ of the adversary $\boldsymbol{A}$ as the set of messages (including the broadcasted ones) sent and received by the bad players during the $\Sigma Share$ protocol. Then there exists an algorithm ***Sim*** called the simulator which, on input m and **VK** and with black-box access to $\boldsymbol{A}$, produces output strings with a distribution which is computationally indistinguishable from $\boldsymbol{V}$.

Remark. We accept a negligible that these conditions are violated. Moreover, completeness means that if Alice really shares the right signature, then, whatever the corrupted proxies do, the signature will be recovered at the end. Soundness means that if Alice is malicious, then either she will be caught trying to cheat or she will share a valid signature anyway. Finally, security says that a run of *ΣShare* gives the adversary no information he could not compute on his own from the message and the public key. In particular, no information about the signature S is revealed unless the scheme is not secure.

2.4 Threshold cryptosystem

With the preceding notations, a *public key encryption scheme E* is defined by three algorithms:

- **Key Generation** is a randomized algorithm that takes a security parameter as input and returns a pair (Y,X) where Y is the public encryption key and X is the secret decryption key.
- **Encryption** takes as input a message M and public key Y and returns a ciphertext $C=E_Y(M)$.
- **Decryption** takes as input a ciphertext $C=E_Y(M)$ and the private decryption key X and returns M.

Threshold cryptosystems. A threshold cryptosystem T_E for E distributes the operation of key generation and decryption among a set of n parties $P_1,\ldots,P_n$. That is, T_E is defined by three protocols:

- **T-Key-Gen**: A randomized protocol that returns as public output the public encryption key Y and as private output for player P_i a value X_i such that $X_1,\ldots,X_n$ constitute a t-out-of-n threshold secret sharing of X.
- **Secret-Key-Gen**: Each player P_i takes as secret input his share X_i, following communication with the other players (who hold the remaining shares of X), and generates X as public output.
- **T-Decrypt**: Each player P_i takes as public input a ciphertext $C=E_Y(M)$ and X and returns as public output the message M.

We say that threshold cryptosystem T_E=(**T-Key-Gen**,**T-Decrypt**) is secure with fault-tolerance t, if for any adversary $\boldsymbol{A}$ that corrupts at most t players the following conditions must be met:

- **correct key generation**: **T-Key-Gen** generates keys with a probability distribution which is computationally indistinguishable from **Key Generation**. Both **T-Key-Gen** and **Secret-Key-Gen** must satisfies the following requirements:

  ***C*1**. All subsets of $t+1$ shares from the honest define the same unique secret key X.

  ***C*2**. All honest players have the same value of public key Y determined by X.
- **correct decryption**: On input $C=E_Y(M)$, **T-Decrypt** returns as output M.
- **simulatability**: Let $\boldsymbol{V}$ be the view of the adversary $\boldsymbol{A}$ during that protocol, which consists of the set of messages sent and received by the corrupted players during a run of that protocol. Then there exists a **simulator *Sim*** with black-box access to $\boldsymbol{A}$ which produces output strings with a distribution which is computationally indistinguishable from $\boldsymbol{V}$.

3 On the Threshold Cryptosystem

In this section, we present a new ElGamal-based threshold cryptosystem over a composite modulus N, which will be used in our RSA *VΣS* later, with the techniques appearing in Refs.[1,6].

3.1 Key generation protocol

We are now ready to show the distributed key generation (DKG) protocol for the later threshold scheme. The general idea follows Gennaro *et al.*[3] for the case of discrete-log cryptosystem over a composite modulus N. We start by running a commitment stage where each player P_i commits to two t-degree polynomials (t is the scheme's threshold) $f_i(z)$, $f_i'(z)$ which shares a random value z_i, z_i' contributed by P_i to the jointly generated secret X and X'. So the following properties from this commitment stage are required: First, the attacker cannot force a commitment by a (corrupted) player P_j to depend on the commitment(s) of any set of honest players. Second, for any non-disqualified player P_i during this stage, there are unique polynomials f_i, f_i' committed to by P_i and these polynomials are recoverable by the honest. Finally, for each honest player P_i and non-disqualified player P_j,P_i holds the value $f_i(j)$, $f_i'(j)$ at the end of this commitment stage.

To realize the above commitment stage, we use the information-theoretic **VSS** protocol due to Pederson, i.e. **Pederson-VSS**[7]. At the end of this commitment stage, the secret key X is determined and no later misbehavior can change it. Most importantly, this guarantees that no bias in the output X or Y of the protocol is possible, and it allow us to present a full proof of security based on simulation. Once X is fixed, the players could compute $Y=G^X(\bmod N)$. The protocol **Key-Gen** appears in below:

Algorithm 1. Protocol Key-Gen.

Input for all players: A composite N as above. An element $G\in Z_N^*$, constructed by taking a random element $G_0\in Z_N^*$ and setting $G=G_0^{L^3}(\bmod N)$. For an element $H\in\langle G\rangle$, assume that it is impossible for the adversary to find $\log_G H$.

Generating *X*:

1. Each player P_i performs an unconditionally secure **VSS** of a random number $z_i\in_R[-N^2,\dots,N^2]$ as a dealer:
   (a) P_i chooses two random polynomials $f_i(z)$ and $f_i'(z)$ over $[-L^2N^3,\dots,L^2N^3]$ of degree t: $f_i(z)=a_{i0}+a_{i1}z+\dots+a_{it}z^t$ and $f_i'(z)=b_{i0}+b_{i1}z+...+b_{it}z^t$. Let $Lz_i=a_{i0}=f_i(0)$ and $z_i'=b_{i0}=f_i'(0)$. P_i broadcasts $C_{ik}=G^{a_{ik}}H^{b_{ik}}(\bmod N)$, where $k\le t$. P_i computes the shares $s_{ij}=f_i(j)$ and $s_{ij}'=f_i'(j)$ for $j=1,2,\dots,n$ and sends s_{ij} and s_{ij}' to player P_j.
   (b) Each player P_j verifies the shares he received from the other players. For each $i=1,2,\dots,n$, P_j check if

$$G^{s_{ij}}H^{s_{ij}'}=C_{i0}\prod_{k=1}^{t}C_{ik}^{j^k}(\bmod N) \tag{1}$$

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

   (c) Each player P_i who, as a dealer, received a complaint from player P_j broadcasts the values s_{ij} and s_{ij}' that satisfy Eq.(1).
   (d) Each player marks as disqualified any player that either
      - Received more than t complaints in Step 1(b); or
      - Answered to a complaint in Step 1(c) with values that falsify Eq.(1).
2. Each player then builds the set of non-disqualified players $QUAL$ (In fact, all honest players build the same set $QUAL$).
3. The distributed secret value X is not explicitly computed by any player, but it equals $X=\sum_{i\in OUAL}z_i$. Each player P_i sets his share of the secret as $x_i=\sum_{j\in QUAL}s_{ij}$; $x_i'=\sum_{j\in QUAL}s_{ij}'$ and $\left\{C_i=\prod_{j\in QUAL}G^{s_{ji}}H^{s_{ji}'}\,|\,1\le i\le n\right\}$.

Extracting *Y*=*G*^*X*(mod*N*):

4. Each player P_i, $i\in QUAL$, exposes $Y_i=G^{z_i}(\bmod N)$.
   (a) Each player P_i, $i\in QUAL$, broadcasts $A_{ik}=G^{a_{ik}}(\bmod N)$ for $k=0,1,\dots,t$.
   (b) Each player P_j verifies the values broadcast by the other players in $QUAL$, namely, for each $i\in QUAL$, P_j checks if

$$G^{s_{ij}}=\prod_{k=0}^{t}A_{ik}^{j^k}(\bmod N) \tag{2}$$

If the check fails for an index i, P_j complains against P_i by broadcasting the values s_{ij} and s_{ij}' that satisfy Eq.(1) but do not satisfy Eq.(2).

(c) For players P_i who receives at least one valid complaint, i.e. values which satisfy Eq.(1) but do not satisfy Eq.(2), the other players run the reconstruction phase of **Pederson-VSS** to compute $Lz_i, f_i(z)$ and A_{ik} for $k=0,1,\ldots,t$. For all players in $QUAL$, set $Y_i = A_{i0} = G^{z_i} (\bmod N)$. Compute $Y = \prod_{i\in QUAL} Y_i (\bmod N)$.

So the public key is set to be Y.

3.2 Secret decryption key generation protocol

We now show the secret decryption key generation protocol in below. The approach is the same as the reconstruction phase of **Pederson-VSS**.

Algorithm 2. Protocol Secret-Decrypt-Key-Gen.

Input for all players: The public input and output of **Key-Gen**, $\{C_i|1\le i\le n\}$, $G_0^{L^2} = G_1$ and H.

Private Input for player P_i: The secret output of **Key-Gen**, i.e., $(x_i; x_i')$.

1. Each player $P_i\in QUAL$ broadcasts $(G_1^{x_i}; H^{x_i'})(\bmod N)$ and proves $L\cdot \log G_1^{x_i} + \log H^{x_i'} = \log C_i$ to each player $P_i\in QUAL$ in **Zero Knowledge** (**ZK**) (see Ref.[7]). Set $A_i = G_1^{x_i} (\bmod N)$.
2. If the proof fails for an index j, P_j broadcasts a complaint against P_i.
3. Each player P_j accepts those for which at most t complaints are broadcast. Take t+1 accepted value A_i and evaluate interpolation coefficients λ_{1i} to compute $A^X = G_1^{LX} = \prod_i A_i^{\lambda_{1i}} (\bmod N)$. So secret decryption key $A^X(\bmod N)$ is obtained.

Theorem 3.1. **TEG**:=(**Key-Gen**,**Secret-Decrypt-Key-Gen**) is a secure key generation protocol for threshold cryptosystem over a composite with fault-tolerance t for any t and n such that $n>2t$.

Proof: We first prove that the distribution of the public key generated by the protocol is “almost” the same as if it was generated by a centralized user. The distribution of Y is induced by that of $X(\bmod\phi(N))$. In the centralized case, X is chosen in Z_N with uniform probability. This results in a distribution statistically close to uniform for Y. So we need to prove that, when X is generated by the protocol, $X(\bmod\phi(N))$ has a distribution which is also statistically close to uniform. Note that $X = \sum_{i\in QUAL} z_i$. Since some of the z_i’s are under the control of adversary, we can set $X=X_A+X_H(\bmod\phi(N))$, where X_A is chosen by the adversary while X_H is determined by the honest. Note that X_A can follow any distribution, but it is independent of X_H since the adversary decided on it at the end of step 1(c) when she has no information about that of the honest. Thus we can consider X_A as a constant. Now it is enough for us to prove that X_H is distributed almost uniform over $Z_{\phi(N)}$. W.l.o.g., we assume the first t+1 players are honest, then $X_H = \sum_{i=1}^{t+1} z_i (\bmod \phi(N))$. For any values u_1 and u_2 in $Z_{\phi(N)}$, we can assume that there exists vector $(z_1,\ldots,z_{t+1})$ such that $u_1 = \sum_{i=1}^{t+1} z_i (\bmod\phi(N))$, where $z_i\in[-N^2,\ldots,N^2]$, then $(z_1,\ldots,z_{t+1}-u_1+u_2)$ can generate u_2. It is easy to verify that this vector is legal if and only if $\Pr[|z_{t+1}|\ge N^2-\phi(N)] < 1/2N$. Note that we can fix any components of the vector, we get $|\Pr[X_H=u_1]-\Pr[X_H=u_2]| < (1/2N)^{t+1}$. So the difference between the distribution of X_H and the uniform over $Z_{\phi(N)}$ is at most $(1/2N)^t$, which is negligible.

For **T-Key-Gen** and **Secret-Key-Gen**, similar to the general t-out-of-n threshold secret sharing scheme (see Ref.[3] and therein references), we know that it satisfies ***C*1** and ***C*2**. Now we show simulation of the protocol. Assume w.l.o.g. that ***A*** corrupts players $P_1,\ldots,P_t$, $B=\{1,\ldots,t\}$ and $\Im=\{t+1,\ldots,n\}$, the indices of the honest. The simulator ***Sim*** works as follows.

*The simulation of a run of **Key-Gen***. During the run of **Key-Gen**, ***A*** sees the following probability distribution of data produced by the uncorrupted players:

- Values $\{s_{ij}, s'_{ij} \mid i \in \Im, j \in B\}$ uniformly chosen in Z_N.
- Values $C_{ik}, A_{ik}, i \in \Im, k=0,\dots,t$ corresponding to coefficients of randomly chosen polynomials and for which Eq.(1) and Eq.(2) are satisfied for all $j \in B$, and $\left\{ C_i \prod_{j \in QUAL} G^{s_{ji}} H^{s'_{ji}} \mid 1 \le i \le n \right\}$.

The simulator ***Sim*** with input Y performs Step 1(a)~Step 1(d), Step 2 on behalf of the uncorrupted players $P_{t+1},\dots,P_n$ exactly as in **Key-Gen**. This includes receiving and processing the information sent privately and publicly from the corrupted to the honest. At the end of Step 2, the following holds in addition to what ***A*** sees as above:

- The set $QUAL$ is well defined. Note that $\Im \subseteq QUAL$ and that polynomials $f_i(z)$, $f'_i(z)$ for $i \in \Im$ are chosen at random.
- ***Sim*** knows all polynomials $f_i(z)$, $f'_i(z)$ for $i \in QUAL$. In particular, he knows all the shares s_{ij}, s'_{ij}, the coefficients a_{ik}, b_{ik} and the public values C_{ik}.

So simulator ***Sim*** performs the following computations:

- Compute $A_{ik} = G^{a_{ik}}$ for $i \in QUAL \backslash \{n\}$, $k=0,\dots,t$
- Set $A^*_{n0} = Y^L \prod_{i \in QUAL \backslash \{n\}} A^{-1}_{i0} (\bmod N)$ and assign $s^*_{nj} = s_{nj} = f_n(j)$ for $j=1,\dots,t$
- Compute $A^*_{nk} = (A^*_{n0})^{\lambda_{k0}} \prod_{i=1}^{t} (G^{s^*_{ni}})^{\lambda_{ki}}$ for $k=1,\dots,t$, where λ_{ki}'s are the Lagrange interpolation coefficients.

  Here we must note that all exponents are integer since $G = G_0^{L^3} (\bmod N)$ and $L=n!$.

Then ***Sim*** performs Step 4(a)~Step 4(c). But we must note that the above distribution of values is characterized by the choice of polynomials $f_i(z)$, $f'_i(z)$ for $i \in \Im \backslash \{n\}$ and $f'_n(z)$ as random independent t-degree polynomials over Z_N, and of $f_n(z)$ as a uniformly chosen polynomial from the family of t-degree polynomials over Z_N satisfying

$$f_n(0) = L \cdot \log_G Y - \sum_{i \in QUAL \backslash \{n\}} f_i(0) = \log_G A^*_{n0} (\bmod N) \qquad (3)$$

Now we show that the probability distribution output by ***Sim*** is identical to the above distribution of ***A***. Note that the above distribution depends on the set $QUAL$ defined at the end of Step 2 of the protocol, since all ***Sim***'s actions performing Step 1, Step 2 are identical to the actions of the honest interacting with ***A*** in a real run of the protocol, we know that the set $QUAL$ is defined in this simulation step identically to that in the real. Now we describe the output distribution of ***Sim*** by modifying some notations as follows:

For $i \in \Im \backslash \{n\}$, set f^*_i to f_i and f'^*_i to f'_i. For $i=n$, define f^*_i via the values $f^*_n(0) = \log_G A^*_{n0}$ and $f^*_n(j) = s^*_{nj} = f_n(j)$, $j=1,\dots,t$. And f'^*_n is defined via the equation: $f^*_n(z) + d f'^*_n(z) = f_n(z) + d f'^*_n(z) (\bmod N)$, where $d=\log_G H$. By this definition, we can see that all the values of these polynomials evaluated at $j \in B$ coincide with that in Step 1. Also, the coefficients of these polynomials agree with exponentials C_{ik} published by the simulated honest in Step 1 as well as with A_{ik}, $i \in \Im \backslash \{n\}$ and A^*_{nk} published by the simulator on behalf of the honest corresponding to that in Step 4a. Hence, all these values pass the verifications of Eq.(1) and Eq.(2) as in the real. So, we only need to prove that polynomials f^*_i and f'^*_i belong to the right distribution. Indeed, for $i \in \Im \backslash \{n\}$, it is immediate by the definition. For f^*_n, at points $j=1,\dots,t$, it evaluates to random value s_{nj}, while at 0, it evaluates $\log_G A^*_{n0}$ satisfying Eq.(3). Moreover, by the definition of f'^*_n as above, and note that f'_n is chosen as a random and independent polynomial in Step 1, so is f'^*_n. So the output of the simulation is clearly Y, and the simulated view of the adversary is identically distributed to that of the real for **Key-Gen**.

The simulation of a run of ***Secret-Decrypt-Key-Gen***. With G_1 and H as input and output $A^X(\bmod N)$, ***Sim*** evaluates interpolation coefficients $\{\lambda_{1i}|i=1,\ldots,t+1\}$. For P_{t+1}, he broadcasts the values $G_1^{\hat{x}_{t+1}}=A^X\Big/\prod_{i\le t}G_1^{x_i\lambda_{1i}}(\operatorname{mod}N)$. Repeat this operation, for each player P_i, $i=t+2,\ldots,n$, ***Sim*** can broadcast $G_1^{\hat{x}_i}$. Hence, for constant λ_{1i}, we have $G_1^{\hat{x}_i}=G_1^{x_i\lambda_{1i}}$. By the same method, ***Sim*** can broadcast $H^{x_i'}(\operatorname{mod}N)$ for $i=t+1,\ldots,n$ and have the similar property. So it is easy to see that the simulated view is identically distributed to that of the real. This completes the proof.

4 RSA Verifiable Signature Sharing Scheme

In this section, we will present a RSA *VΣS* scheme in below, where we take full advantage of the distributed key generation of the preceding protocol. The key is generated distributively by proxies instead of Alice. This will also allow for a very efficient verification that the ciphertext contains the required signature. Indeed, the proxies can verify that the signature is contained in the pair (A_S,B_S) in the following protocol correctly by checking that

$$B_S^{v_B}\Big/(A_S^{v_B})^{XL^3}=(Y^{KS})^{v_B}\Big/(G^K)^{XL^3v_B}=Y^{Kv_B}S^{v_B}\Big/Y^{Kv_B}=m(\operatorname{mod}N) \tag{4}$$

Algorithm 2. Protocol RSA *VΣS*.

Input for Alice: The message m, Bob's **RSA** public key (N,v_B), the signature S on m, i.e. a value such that $m=S^{v_B}(\operatorname{mod}N)$.

RSA-*ΣShare*:

1. Alice sends to proxies the message m, Bob's **RSA** public key (N,v_B) and a random value $r\in Z_N$.
2. The proxies run **Key-Gen** on input N and the bases $G_0=m^r(\operatorname{mod}N)$, $G=G_0^{L^3}(\operatorname{mod}N)$ and $H\in\langle G\rangle$. They return to Alice the public key $Y=G^X(\operatorname{mod}N)$.
3. Alice encrypts S using the **ElGamal** encryption scheme with public key (N,G_0,G,Y). That is, she generates a random number $K\in Z_N$ and computes $A_S=G_0^{KL^2}(\operatorname{mod}N)$ and $B_S=Y^{KS}(\operatorname{mod}N)$. Alice sends (A_S,B_S) to all proxies.
4. The proxies run the **Secret-Decrypt-Key-Gen** to get A_S^{LX}. Then, on the pair $(A_S^{LXv_B},B_S^{v_B})$, they compute $B_S^{v_B}\Big/A_S^{LXv_B}$. If the output is m, they accept; otherwise reject.

RSA-*ΣRecover*: The proxies run **Secret-Decrypt-Key-Gen** on the pair (A_S,B_S) to get A_S^{LX}. Then $S=B_S\Big/A_S^{XL}$.

Theorem 4.1. Under the Decisional Diffie-Hellman assumption modulo a composite, the protocol **RSA** *VΣS* is a secure *VΣS* protocol for **RSA** with fault-tolerance t for any n, t with $n>2t$.

Proof: It is easy to see that the correctness of **Secret-Decrypt-Key-Gen** results in **RSA** *VΣS*'s correctness and soundness. And Eq.(4) is a necessary and sufficient condition for (A_S,B_S) to decrypt to the signature. So either all the proxies reject if Eq.(4) is not satisfied, or they will all accept when the signature will be decrypted successfully at the end of **RSA**-*ΣRecover*.

Now we only need to prove the security of this scheme. W.l.o.g., assume that adversary ***A*** corrupts proxies $P_1,\ldots,P_t$. The simulator ***Sim*** works on input m and (N,v_B), but not the signature S:

1. ***Sim*** just sends m, (N,v_B) and a randomly chosen $\hat{r}$ to the proxies.

2. ***Sim*** runs **Key-Gen** for the good proxies, where the bases are set to be $\hat{G}=m^{\hat{r}}(\operatorname{mod}N)$ and $\hat{H}\in\langle\hat{G}\rangle$. At the end, the values $\hat{C}_i=\hat{G}^{\hat{x}_i}\hat{H}^{x_i'}$ are public and ***Sim*** knows the shares $(\hat{x}_i,\hat{x}_i')$ of the secret key of all proxies.

3. ***Sim*** encrypts the value 1 by choosing a random $\hat{K}\in Z_N$ since he does not know S and broadcasting $\hat{A}_S=\hat{G}^{\hat{K}}(\operatorname{mod}N)$ and $\hat{B}_S=\hat{Y}^{\hat{K}}(\operatorname{mod}N)$.

4. At this point the proxies runs **Secret-Decrypt-Key-Gen** on $A^*=\hat{A}_S$ and $\hat{H}$. In order to get m as the

result, ***Sim*** has to cheat as follows:

Note that he knows $A_i^* = (A^*)^{\hat{x}_i}$ for $1 \le i \le t$ and interpolation coefficients λ_{ij} for all i and j, and each proxy broadcasts A_i^* and prove in **ZK** (see Ref.[6]) that it is correct with respect to $\hat{C}_i$. For any t+1 proxies $P_{i_1},\dots,P_{i_{t+1}}$, the simulator will broadcast A_j^* for $j \in \{i_1,\dots,i_t\}$ and prove their correctness in **ZK**. Then for $P_{i_{t+1}}$, he broadcasts $A_{i_{t+1}}^* = \hat{B}_S^{v_B} \Big/ m \prod_{i_1 \le j \le i_t} (A_j^{*v_B})^{\lambda_{1j}}$, whose correctness will be proved in **ZK** either.

So the differences between the simulated view and the real are as follows:

i) In the real execution (A_S,B_S) is an encryption of S, while in the simulation, $(\hat{A}_S, \hat{B}_S)$ is an encryption of 1.

ii) In the real execution, $(G_1^K, G_1^{x_{t+1}} = A_{t+1}, A_S^{x_{t+1}} = G_1^{x_{t+1}K})$ is a Diffie-Hellman triplet. In the simulated execution, $(\hat{A}_S = \hat{G}^{\hat{K}}, \hat{G}^{\hat{x}_{t+1}}, A_{i_{t+1}}^*)$ is one.

If we distinguish between the real view and the simulated view, then we could distinguish either of the above two cases. It is easy to see that this contradicts the DDH assumption. This completes the proof.

5 Conclusions

We present a new, efficient and provably secure *VΣS* protocol for **RSA** signature scheme against static active adversary with a negligible probability, which substantially puts forward both theory and practice in this field. Indeed it could be widely applied in cash escrow, secure distributed auction, and distributed cryptosystems etc., and achieves best-possible robustness at present. Although there are some efficient protocols given for RSA, Rabin, ElGamal, Schnorr and DSS signatures, their RSA and Rabin *VΣS* protocols were subsequently broken[2]. Catalano et give a new scheme based on key generation protocol from Feldman's verifiable secret sharing protocol (**VSS**) and threshold cryptosystems, but as we had shown that Feldman's verifiable secret sharing protocol is not secure. In view of this, our protocol is more secure and as practical as their. Of course, our scheme is complicated and unfavorable to application to some extent, so it may be an interesting problem to find more simple and practical, secure *VΣS* protocol.

References:

[1] Franklin m, Reiter M. Verifiable signature sharing. In: Proc. of the Eurocrypt'95. LNCS 921, Springer-Verlag, 1995. 50−63.
[2] Catalano D, Gennaro R. New efficient and secure protocols for verifiable signature sharing and other applications. In: Proc. of the CRYPTO'98. LNCS 1462, Springer-Verlag, 1998. 105−120.
[3] Gennaro R, Jarecki S, Krawczyk H, Rabin T. The secure distributed key generation for discrete-log based cryptosystems. In: Proc. of the EUROCRYPT'99. LNCS 1592, Springer-Verlag, 1999. 295−310.
[4] Diffie W, Hellman ME. New directions in cryptography. IEEE Trans. on Information Theory, 1976,IT-22(6):644-654.
[5] ElGamal T. A public-key cryptosystem and a signature scheme based on discrete logarithms. IEEE Trans. on Information Theory, 1985, IT-31(4):469−472.
[6] Gennaro R, Jarecki S, Krawczyk H, Rabin T. Robust and efficient sharing of RSA functions. In: Proc. of the EUROCRYPT'96. LNCS 1109, Springer-Verlag, 1996. 157−172.
[7] Pederson T. Non-interactive and information-theoretic secure verifiable secret sharing. In: Feigenbaum J, ed. Advances in Cryptology CRYPTO'91. LNCS 576, Berlin: Springer-Verlag, 1991. 129−140.

LÜ Ke-wei was born in 1970. He is an associate professor at the Graduate School of Chinese Academy of Sciences. His current research areas are complexity theory, secure protocols, signature, and zero knowledge proof.