INFORMATICA, 2012, Vol. 23, No. 3, 443–460
© 2012 *Vilnius University*

Tightly Secure Non-Interactive Multisignatures in the Plain Public Key Model

Haifeng QIAN[1,2], Xiangxue LI[1] *, Xinli HUANG[1]
[1]*Department of Computer Science and Technology, East China Normal University Shanghai 200241, China*
[2]*Hangzhou Key Lab of E-business and Information Security, Hangzhou Normal University Hangzhou 310036, China*
e-mail: {hfqian, xxli}@cs.ecnu.edu.cn

Received: April 2011; accepted: March 2012

Abstract. Multisignature scheme allows a group of signers to generate a compact signature on a common document that certifies they endorsed the message. However, the existing state of the art multisignatures often suffers from the following problems: impractical key setup assumptions, loose security reductions and inefficient signature verification. In this paper, we propose a non-interactive multisignature scheme with tight security reduction in the random oracle model. Our proposed multisignatures address the above three problems by achieving: provable security in the *plain* public key model; *tight* security reduction under the standard Computational Diffie–Hellman (CDH) assumption and $\mathcal{O}(1)$ computational time for signature verification through pre-computation. Hence, our non-interactive multisignatures are of great use in routing authentication of networks.

Keywords: rogue key attacks, plain public key model, provable security, multisignatures, tight security.

1. Introduction

A multisignature scheme enables multiple signers to jointly authenticate a message and produce a digital signature of compact size on behalf of all signers. Multisignatures provides efficient batch verification of several signatures of the same message under different public keys. This primitive now are widely used in contract signing, authentication of routes in mobile networks, distribution of a certificate authority (CA), aggregation of acknowledgements in multicast (Micali *et al.*, 2001; Bellare and Neven, 2006; Boldyreva, 2003; Kim and Tsudik, 2005; Bagherzandi *et al.*, 2008; Bagherzandi and Jarecki, 2008; Castelluccia *et al.*, 2006; Lu *et al.*, 2010), etc.

However, prior multisignatures achieve provable security at the cost of (1) imposing complex key setup assumptions on the public key infrastructures (PKIs); (2) efficiency on multisignature generation and verification and (3) loose security reduction which might imply large security parameters.

*Corresponding author.

For practical reasons of application, we desire that a multisignature scheme might have the following features: (1) the resulting signature is of constant size for ℓ signers; (2) multisignature generation and verification (even key generation) are very fast (efficient); (3) the communication overhead in multisignature generation should be as small as possible (even reduced to a minimum); (4) trust on the trusted third party (e.g., Certificate Authority) should be reduced as less as possible. All these specific aspects are important in the real life applications of multisignatures.

1.1. *Rogue Key Attacks*

The homomorphic properties of arithmetic operations involved in standard signatures enable aggregation of signatures into multisignatures of constant size. However, these homomorphic properties used in generating multisignatures often incur rogue key attacks for multisignature schemes. In such an attack, the adversary chooses its public key as a function of those of honest signers through which it can forge multi-signatures easily (Bellare and Neven, 2006; Bagherzandi *et al.*, 2008).

For example, rogue key attack succeeds if the verification key for multisignature has a fixed formula as $PK = \prod_i^{\ell} pk_i$ (generated from the public keys of signers). In such a scheme, the adversary may choose $pk_a = g^s \cdot (\prod_i^{\ell-1} pk_i)^{-1}$ for a known random s. Then, the private key for $PK = pk_a \prod_i^{\ell-1} pk_i = g^s$ is s. Finally, the adversary can easily mount rogue key attacks (e.g., Itakura and Nakamura, 1983; Ohta and Okamoto, 1999; Boldyreva, 2003; Lu *et al.*, 2006; Ristenpart and Yilek, 2007) are vulnerable to such attacks in the plain setting). Actually rogue-key attack is considered as a main menace for discrete logarithm based multi-signature schemes.

To prevent rogue-key attacks for multisignatures, many proposals have been put forward, but either at the cost of complexity and expense, or imposing unrealistic and complicate key setup assumptions on the public-key infrastructure (PKI; Micali *et al.*, 2001). These key setup operation assumptions include *dedicated key generation* (DKG), *knowledge of Secret Key* (KOSK), *proof of possesion of private key* (POP).

The *first* effort to prevent rogue key attacks (called DKG) is due to Micali *et al.* (2001). However, the DKG is impractical because of expensive interactions of key generation, complex and large public keys, and static group of signers. The *second* approach, KOSK assumption needs a party to prove knowledge of its secret key, during public key registration with a certificate authority (CA). The requirement of handing over the secret keys leads to obtaining simple constructions and proofs of security (Boldyreva, 2003; Lu *et al.*, 2006). However, the existing public key infrastructures (PKIs) do not require proofs of knowledge of secret keys (Adams *et al.*, 2005). The *third* one is contributed by Ristenpart and Yilek (2007), Bagherzandi *et al.* (2008) and Bagherzandi and Jarecki (2008) where users are required to provide proofs of procession of secret keys in order to prevent rogue key attacks. Ristenpart and Yilek (2007) showed that the schemes in Boldyreva (2003), Lu *et al.* (2006) by using the Key Registration (KR) model can be improved more secure without reducing efficiency. A similar idea named the Key Verification (KV) model was later proposed in Bagherzandi *et al.* (2008) and Bagherzandi and Jarecki (2008), but having the multisignature receiver verify the POP message (together with verification of PKI certificates), instead of the CAs during the key registration. While none are initialized in the real life applications (Schaad, 2005).

We note that either the KR model or the KOSK assumption needs non-standard trust on the CAs because it requires the CAs must perform specific verifications. If a CA is corrupted then the adversary can easily forge multisignature through rogue key attacks. On the other hand, the interaction and verification during key registration also causes additional computational burden for the CAs. While the KV model causes additional computational cost of the verifier (linear to the number of signers) during the verification of a multisignature.

Obviously, it is highly interesting and desirable to provide multisignature schemes which are secure in the plain setting where no special registration process is assumed for public keys registration. Bellare and Neven formalized such a security model, called the plain public key (PPK) model and presented a scheme in such a model (Bellare and Neven, 2006), followed by Bagherzandi *et al.* (2008), Ma *et al.* (2009), Boneh *et al.* (2003). In the plain (public-key) model, there is no dedicated key generation (DKG) procedures, or well-formedness proofs accompanied to the public keys. Namely a party can obtain a certificate on an arbitrary key.

1.2. *Multisignatures in the PPK Model*

Up to date there are very few multisignature schemes with provable security in the plain public key model. The first multisignature secure in the PPK model was proposed by Boneh *et al.* (2003) (for brevity BGLS), which is implied by the construction so-called aggregate signature (Bellare *et al.*, 2006) in the random oracle model (Bellare and Rogaway, 1993). The main drawback of the BGLS scheme is the high cost of verification. In fact verification of a single multisignature of the BGLS scheme requires $\mathcal{O}(\ell)$ pairings where ℓ is the number of signers participating in signing, that makes the BGLS scheme extremely impractical.

Under the DL assumption Bellare and Neven (2006), provided a concrete multisignature scheme (denoted BN) with rather an efficient verification in the PPK model. Followed the idea of Bellare and Neven (2006), a lot of *interactive* multisignature schemes are proposed (Bagherzandi *et al.*, 2008; Ma *et al.*, 2009) in the PPK model. However, the interaction is quite expensive in many important application because communicating even one bit of data may use significantly more power than executing one 32-bit instruction (Barr and Asanović, 2003) and also in many settings, communication is not reliable, and so the fewer interactions, the better.

The only known non-interactive multisignature scheme with efficient verification in the PPK model (the QX scheme) is proposed by Qian and Xu quite recently in Qian and Xu (2010). Comparing with the BGLS scheme, the QX scheme improves verification efficiency by reducing pairing computation complexity $\mathcal{O}(\ell)$ to $\mathcal{O}(1)$. However, security proof of the QX scheme is even looser than that of the BN scheme (Bellare and Neven, 2006). Intuitively, a tight security (proof) means that the scheme is almost as hard to break as the underlying cryptographic problem to solve. Therefore, it is always welcome to find a non-interactive multisignature scheme in the PPK model with more tighter security proof.

REMARK 1. For each scheme, we summarize the underlying cryptographic assumption; number of rounds of the signing protocol ("1" means non-interactive); the computational complexity for each signer (Sign); the communication cost required for the signing (Comm. Cost) ; the computational complexity for verifying a multisignature (Verify); the size of multisignature (σ Size). For CDH-based schemes we assume the symmetric pairing e: $\mathbb{G} \times \mathbb{G} \longrightarrow \mathbb{G}_T$ for convenient comparison. For DL-based schemes, we assume we work over a 160-bit elliptic-curve (EC) group $\mathbb{G}'$. We assume the order of $\mathbb{G}$, $\mathbb{G}_T$ and $\mathbb{G}'$ are equal (i.e., $p = q$). We denote by exp an exponentiation in group $\mathbb{G}$ (or $\mathbb{G}'$ and $\mathbb{G}_T$) whose order is q (or p), and by $\mathsf{mexp}^{(t)}$ a multi-exponentiation with t exponentiation coefficients (e.g., $\mathsf{mexp}^{(2)}$ corresponds to $g^{k_1}h^{k_2}$ for some g, h, k_1, k_2), by ℓ the number of signers, by l_0 the length of hash value, by pr a bilinear pairing, by $|\mathbb{G}|$ the bit length of the representation for elements in group $\mathbb{G}$, and by $|p|$ the bits length of p. DL stands for Discrete Logarithm, CDH stands for Computational Diffie–Hellman. "$\ll$" and "$<$" means very loose security and loose security, respectively. "$\approx$" means close security.

1.3. *Our Contributions*

We present a non-interactive multisignature scheme, which operates in the plain public-key model and is proven tightly-secure based on the standard CDH assumption in the random oracle model. Compared with the BGLS scheme, our scheme minimize the verification cost, by reducing $(\ell+1)$ pairings to four pairings through pre-computation. While the BGLS scheme needs the signers' public keys to be used as prefixes to the corresponding message in the BGLS scheme, implying that verification of the BGLS multisignature needs $(\ell + 1)$ pairings necessarily. Our improvement in verification time is substantial because one pairing costs about 6–20 exponentiations (Bellare and Neven, 2006). Given ℓ signers, the verification key is fixed (consisting of ℓ partial verification keys that can be derived from the public keys independently), which means that we can compute once and for all, the verification key before signing or verification.

In particular, our scheme also enjoys a tight security proof, comparing with both the BN scheme and the QX scheme. Then security parameters of our scheme could be smaller than those of both the BN scheme and the QX scheme, while preserving the same security level. Actually, our security proof shows that an adversary can at most with probability (roughly) $\varepsilon/2$ break our multisignature scheme where ε is the upper bound of probability for breaking the underlying cryptographic problem (the CDH problem). While in the BN scheme the corresponding upper bound is roughly $\sqrt{q_h \cdot \varepsilon}$. Let $q_h = 2^{80}$, $q_s = 2^{40}$, $\varepsilon = 2^{-80}$, our scheme ensures 79 bits of security level, while the BN scheme (or the QX scheme) ensures at most 1 bits of security level (which is of no means in practice). Comparing with the QX scheme, our scheme also saves the mult-exponentiation in verification of a multisignature and achieves tight security as stated, but at the price of a little bit of signature expansion. Our multisignatures double that of the QX's. Detailed comparisons

Table 1

Comparison of Multisignatures in the Plain Public Key Model

Scheme	Assump.	Rounds	Sign	Comm. cost	Verify	σ size
BN	$<$ DL	3	1 exp	$\lvert\mathbb{G}'\rvert + \lvert q\rvert + l_0$	1 mexp$^{(\ell+1)}$	$\lvert\mathbb{G}'\rvert + \lvert q\rvert$
BGLS	$<$ CDH	1	1 exp	$\lvert\mathbb{G}\rvert$	$(\ell+1)$ pr	$\lvert\mathbb{G}\rvert$
QX	$\ll$ CDH	1	1 exp	$\lvert\mathbb{G}\rvert$	2pr + mexp$^{(\ell+1)}$	$\lvert\mathbb{G}\rvert$
Ours	$\approx$ CDH	1	2 exp	2 $\lvert\mathbb{G}\rvert$+1	4 pr	4 $\lvert\mathbb{G}\rvert$

amongst our scheme, the BN scheme, the BGLS scheme and the QX scheme are depicted in Table 1.

The technique for dealing with the rogue key attack in the plain public-key model may be of independent interest since it is quite different from those of Bellare and Neven (2006). Instead of using a dynamic key with respective to messages (Bellare and Neven, 2006) as the verification key for the multisignatures, we use a combined key derived from the public keys of signers, irrelevant to messages. Therefore, we can pre-compute the verification key for any group of signers before knowing the signed messages. Such pre-computation could be done when the certificates of public keys are verified. This technique also reduces the communication rounds of multisignatures to optimal since our scheme is non-interactive. The computational cost and communication cost (the amount of data transmit in generating a single multisignature) are the same as the WMS scheme (Lu *et al.*, 2006) whose security holds under the KOSK assumption, but not secure in the PPK model. Moreover, our multisignature reaches high level of security, in fact our scheme is as secure as the Computational Diffie–Hellman (CDH) problem.

1.4. *Organization*

The rest of the paper is organized as follows. In Section 2 we review the definition and security model of multisignatures. In Section 3 we present our multisignature scheme, and Section 4 we analyze security of our multisignatures. In Section 5, we discuss some important features on our multisignatures. Finally, in Section 6 we conclude the paper.

2. Preliminaries

We recall the basic definitions for multisignatures, then review the cryptographic complexity assumption in this section.

Before proceeding, we explain the notations as follows: If s is a binary string, then $|s|$ denotes its length. If G is a group, then $|G|$ denotes the bit size of its elements. If $s_1, s_2, \ldots$ are strings, then $s_1\|s_2\|\ldots$ denotes their concatenation. If S is a (multi)set, then $s \xleftarrow{R} S$ denotes the operation of selecting s uniformly distributed in S. We use $L = (pk_1, \ldots, pk_\ell)$ to represent the (multi)set $L = \{pk_1, \ldots, pk_\ell\}$ hereafter.

2.1. *Definitions of Multisignatures*

The definition of interactive multisignatures with ℓ signers, each having as input its own public and private keys as well as the public keys of the other signers in the plain public-key model was first formalized in Bellare and Neven (2006). The signers interact via a protocol to generate a multisignature.

In this paper, we consider a more general case, the non-interactive variant: given the same inputs as in an interactive scheme, each signer contributes a partial signature *without* interacting with each other, and the partial signatures can be "assembled" into a multisignature by any one finally.

Formally, a non-interactive multisignature scheme $\mathsf{MS} = (\mathsf{Setup}, \mathsf{Gen}, \mathsf{MSign}, \mathsf{MVf})$ consists of three algorithms and one protocol; adapted from Bellare and Neven (2006):

- $\mathsf{Setup}(1^\lambda)$: This is a randomized algorithm that takes as input a security parameter λ and produces a set of global public parameters pp. (This algorithm should be run by a trusted party and pp can also be viewed as a common reference string.)
- $\mathsf{Gen}(pp)$: This is a randomized algorithm that, on input the public parameters pp, outputs (an honest) signer i's private/public key pair (sk_i, pk_i).
- MSign is a multisignature generation protocol executed by a group of players L who intend to sign the same message M (note that $pk_i = pk_j$ for some $i \neq j$ is allowed in the plain public-key model because one can simply claim another's public key as its own). Each signer P_i executes this protocol on public inputs pp, message M and private input sk_i, his secret key. The output of the protocol is a multisignature denoted σ (that can be verified under the public keys of the group L). Actually, given a partial σ_j (generated by P_j) for $j = 1, \dots, \ell$, any one can obtain a multisignature σ with respect to the public keys on L if it is a non-interactive multisignature scheme.
- $\mathsf{MVf}(pp, M, \sigma, L)$: Given $L = (pk_1, \dots, pk_\ell)$, pp, a message M, a multisignature σ, this deterministic algorithm outputs 0 (reject) or 1 (accept).

We require a multisignature scheme to be *correct*, meaning that every multisignature σ obtained from the partial signatures of legitimate signers (according to MSign) is always accepted as valid.

Security of multisignature scheme requires that it is impossible for any adversary $\mathcal{A}$ to forge a valid multisignature with respect to a new message that extends the classical security notion of digital signature scheme known as existential unforgeability under adaptively chosen-message attacks (Goldwasser *et al.*, 1988). Following Bellare and Neven (2006) and Bagherzandi *et al.* (2008), we assume that there is a single honest signer.

Unforgeability of multisignature in the plain public-key model allows the adversary to corrupt all other signers (except the honest signer) and choose their public keys arbitrarily (even to register the public key of the honest user as their own public keys), and to interact with the honest signer in any number of concurrent signing instances before outputting its forgery eventually.

Formally, we define the advantage of $\mathcal{A}$ against multisignature scheme MS as the probability that the experiment $\mathrm{Exp}^{\mathsf{MS}}_{\mathsf{uu.cma}}(\mathcal{A})$ in Fig. 1 outputs 1. We say the adversary

Experiment $\text{Exp}^{\mathsf{MS}}_{\mathsf{uu.cma}}(\mathcal{A})$:

1. $pp \longleftarrow \mathsf{Setup}(1^\lambda)$; $(pk^\star, sk^\star) \longleftarrow \mathsf{Gen}(pp)$;
2. Run $\mathcal{A}(pp, pk^\star)$ as follows:

   $\mathcal{A}$ can choose arbitrary public key for any user, possibly as a function of the honest user's public key $pk^\star$

   To obtain a multisignature, $\mathcal{A}$ can invoke the execution of $\mathsf{MSign}(\cdot,\cdot,\cdot,\cdot)$ (concurrently) by presenting a message M and a (multi)set $L = (pk_1, \dots, pk_n)$ for any n, as long as $pk^\star$ appears at least once in L. $\mathcal{M} \longleftarrow \phi$ where $\mathcal{M}$ is the set of messages previously queried for signatures.

   If the multisignature scheme uses hash functions that are treated as random oracles, $\mathcal{A}$ can submit strings and obtain their corresponding hash values.
3. Eventually, $\mathcal{A}$ outputs an alleged multisignature $\sigma^\star$ on a message $M^\star$ with respect to $L^\star = (pk_1, \dots, pk_\ell)$. If

$$\mathsf{MVf}(pp, M^\star, \sigma^\star, L^\star) = 1$$

and

$$(pk^\star \in L^\star) \wedge \quad M^\star \notin \mathcal{M} = 1$$

then return 1, otherwise 0.

Fig. 1. Experiment for security definition.

$(t, q_s, q_h, \ell, \varepsilon)$-breaks multisignature scheme MS, if it in time t, after q_s signature queries or q_s invocations of $\mathsf{MSign}(\cdot,\cdot,\cdot)$, and optionally q_h queries to the hash functions (if any) that are treated as random oracles, has an advantage at least ε in forging a multisignature co-signed by ℓ signers, namely

$$\Pr[\text{Exp}^{\mathsf{MS}}_{\mathsf{uu.cma}}(\mathcal{A}) = 1] > \varepsilon.$$

If there is no such adversary that $(t, q_s, q_h, \ell, \varepsilon)$-breaks multisignature scheme MS, we say the multisignature scheme is $(t, q_s, q_h, \ell, \varepsilon)$-secure.

2.2. *Cryptographic Complexity Assumption*

Let $\mathbb{G}$ and $\mathbb{G}_T$ be two (multiplicative) cyclic groups of prime order p where the group action on $\mathbb{G}$ and $\mathbb{G}_T$ can be computed efficiently, g be a generator of $\mathbb{G}$, e: $\mathbb{G} \times \mathbb{G} \longrightarrow \mathbb{G}_T$ be an efficiently computable map (i.e., pairing) with the following properties:

- bilinear: for all $(u, v) \in \mathbb{G} \times \mathbb{G}$ and $a, b \in \mathbb{Z}_p$, $e(u^a, v^b) = e(u, v)^{ab}$;
- non-degenerate: $e(g, g) \neq 1$.

For specific applications, we recommend the asymmetric setting, namely $\mathbb{G}_1 \neq \mathbb{G}_2$ for bilinear maps (i.e., e: $\mathbb{G}_1 \times \mathbb{G}_2 \longrightarrow \mathbb{G}_T$), that allows for short signatures without side-effect. Our scheme is also adaptable for such a setting. For more details, refer to Boneh *et al.* (2001), Galbraith *et al.* (2008).

We define the computational Diffie–Hellman problem (with pairings) as follow.

DEFINITION 1. Given $(g, g^a, h) \in \mathbb{G} \times \mathbb{G} \times \mathbb{G}$ for some random $a \longleftarrow \mathbb{Z}_p$ and random $h \longleftarrow \mathbb{G}$, find $h^a \in \mathbb{G}$.

Define the success probability of an algorithm $\mathcal{A}$ solving the CDH problem as

$$\mathrm{Adv}_{\mathcal{A}}^{\mathrm{cdh}} \overset{\mathrm{def}}{=} \Pr\bigl[h^a \longleftarrow \mathcal{A}(g, g^a, h)\colon g \overset{R}{\longleftarrow} \mathbb{G}, a \overset{R}{\longleftarrow} \mathbb{Z}_p, h \overset{R}{\longleftarrow} \mathbb{G}\bigr].$$

The probability is taken over the uniform random choice of g from $\mathbb{G}$, of a from $\mathbb{Z}_p$, of h from $\mathbb{G}$, and the coin tosses of $\mathcal{A}$. We say the algorithm $\mathcal{A}$ (t, ε)-solves the CDH problem if $\mathcal{A}$ runs in time at most t and $\mathrm{Adv}_{\mathcal{A}}^{\mathrm{cdh}}$ is at least ε. We say the CDH problem is (t, ε)-intractable if there is no algorithm $\mathcal{A}$ that can (t, ε)-solve it.

3. Our Construction

Our scheme uses a Waters-like signature (e.g., $\sigma = (sk \cdot H(m)^r, g^r)$) scheme to construct multisignatures, however this scheme is different from the WMS multisignature scheme in Lu *et al.* (2006) since security of our scheme is proved in the plain public key model (with random oracles) while the WMS multisignature (whose security is proved in the KOSK model) must impose additional requirement on the traditional PKIs to ensure security (e.g., it requires the CAs and users to perform specific protocols to get public key certificated).

Note that security of multisignatures in the KOSK model relies on the trust of the CAs. While security of multisignatures in the plain public key model does not. Therefore, our scheme reduces the trust of the third party (e.g., CA) because even a malicious CA can't do any harm to the honest users in our system.

Our scheme consists of the following algorithms (or protocols):

$\mathsf{Setup}(1^\lambda)$**:** On input a security parameter λ, select global public parameters $pp = (\mathbb{G}, \mathbb{G}_T, p, g, e, H, H_m)$, where $H_m\colon \{0,1\}^* \longrightarrow \mathbb{G}$, $H\colon \mathbb{G} \longrightarrow \mathbb{G}$ and $H_b\colon \{0,1\}^* \longrightarrow \{0,1\}$ are secure hash functions (viewed as random oracles here). This algorithm may be run by a trust party.

$\mathsf{Gen}(pp)$**:** On input pp, an honest user i selects random $x_i \longleftarrow \mathbb{Z}_p$, then sets its private/public key pair (sk_i, pk_i) as where

$$sk_i = H(pk_i)^{x_i}, pk_i = g^{x_i}.$$

$\mathsf{MSign}(pp, sk_i, M)$**:** On input pp, message M, user i $(i = 1, \ldots, \ell)$ executes the following:

1. Pick a random bit $b_i = H_b(sk_i \| M)$ and a random $r \overset{R}{\longleftarrow} \mathbb{Z}_p$ and compute

$$s_i \longleftarrow sk_i \cdot H_m(M \| b_i)^r \quad \text{and} \quad t_i \longleftarrow g^r.$$

2. Broadcast $\sigma_i = (s_i, t_i, b_i)$ as the partial signature for message M (which is a standard signature already).

Given partial signatures $\sigma_1, \ldots, \sigma_\ell$, any one can compute the multisignature for group $L' = (pk_1, \ldots, pk_\ell)$ as follows:

1. Set $L_0 = (pk_{i_1}, \ldots, pk_{i_k})$, where $b_{i_c} = 0$ for $c = 1, \ldots, k$ and $L_1 = (pk_{j_1}, \ldots, pk_{j_n})$, where $b_{j_d} = 0$ for $d = 1, \ldots, n$ ($\ell = n + k$).
2. Compute

$$\sigma^0 = \bigotimes_{c=1}^{k} \sigma_{i_c} = \left(\prod_{c=1}^{k} s_{i_c}, \prod_{c=1}^{k} t_{i_c} \right)$$

and

$$\sigma^1 = \bigotimes_{d=1}^{n} \sigma_{j_d} = \left(\prod_{d=1}^{n} s_{j_d}, \prod_{d=1}^{n} t_{j_d} \right).$$

3. Output $\sigma = (\sigma^0, \sigma^1)$ (and as well as $L = (L_0, L_1)$, M) as the multisignature.

$\mathsf{MVf}(pp, M, \sigma, L)$**:** Given pp, $L = (L_0, L_1)$, message M, and an alleged multisignature $\sigma = (\sigma^0, \sigma^1)$ where $\sigma^0 = (s^0, t^0)$, $\sigma^1 = (s^1, t^1)$, a verifier accepts the multisignature if both

$$e(s^0, g) = e\big(H_m(M\|0), t^0\big) \cdot \prod_{pk_i \in L_0} A_i,$$

and

$$e(s^1, g) = e\big(H_m(M\|1), t^1\big) \cdot \prod_{pk_i \in L_1} A_i,$$

where $A_i = e(H(pk_i), pk_i)$ and rejects otherwise.

The scheme is *correct* because both

$$\begin{aligned}
e(s^0, g) &= e\left(\left(\prod_{pk_i \in L_0} H(pk_i)^{x_i} \right) \cdot H_m\big(M\|0\big)^r, g \right) \\
&= e\big(H_m\big(M\|0\big)^r, g\big) \cdot \prod_{pk_i \in L_0} e\big(H(pk_i), g^{x_i}\big) \\
&= e\big(H_m(M\|0), t^0\big) \cdot \prod_{pk_i \in L_0} e\big(H(pk_i), pk_i\big) \\
&= e\big(H_m(M\|0), t^0\big) \cdot \prod_{pk_i \in L_0} A_i,
\end{aligned}$$

and

$$\begin{aligned} e(s^1, g) &= e\Big(\Big(\prod_{pk_i \in L_1} H(pk_i)^{x_i}\Big) \cdot H_m(M\|1)^r, g\Big) \\ &= e\big(H_m(M\|1)^r, g\big) \cdot \prod_{pk_i \in L_1} e\big(H(pk_i), g^{x_i}\big) \\ &= e\big(H_m(M\|1), t^1\big) \cdot \prod_{pk_i \in L_1} e\big(H(pk_i), pk_i\big) \\ &= e\big(H_m(M\|1), t^1\big) \cdot \prod_{pk_i \in L_1} A_i \end{aligned}$$

hold.

REMARK 2. In our scheme, we divide the group of signers L into two sub groups (L_0, L_1) that indicates the random bit used by each group of signers. While in verification of a multisignature, it seems to compute $A_i = e(H(pk_i), pk_i)$ necessary which needs one pairing computation. While this computation is once for all and can be finished when checking the validity of the public key certificates of signers. Therefore this computation of pairing can be saved through pre-computation before performing multisignature verification since A_i is independent from the content of messages. Comparing with those in the plain public key model (Bellare and Neven, 2006; Bagherzandi *et al.*, 2008), our multisignature scheme is one of the most efficient scheme in verification since our scheme achieves $\mathcal{O}(1)$-verification (respective to pairing computations) through precomputation. Surely, by using such a technique, our multisignatures can be verified online.

REMARK 3. As in many applications, signers might not know who (included L) are going to sign the message M, it is also interesting to find a proper multisignature scheme that can be applied to this situation. Our scheme achieve such a useful feature indeed. Therefore, it security only prevents forgery on a new message (Lu *et al.*, 2006; Ristenpart and Yilek, 2007), not a pair of message/signers as those in Bellare and Neven (2006), However, it can be realized if we replace the message 'M' by '$M\|L'$ as note by Bagherzandi *et al.* (2008).

4. Security of Our Multisignatures

In this section we first explain the techniques of our proof and then present our proof in the random oracle model for the construction.

4.1. *Our Proof Technique*

Our construction results in a tight security reduction by using the approaches from Katz and Wang (2003). In the following proof we assume that the adversary attacks the first signer with pk_1 without loss of generosity.

In the construction of our scheme, each signer participating in the multisigning protocol $\mathsf{MSign}(pp, sk_i, M)$ only generates a signature related to a *fixed* bit b (determined by the secret key sk and the message M) if he/she is given a message M. Thus we should simulates such a multi-signing oracle to respond the queries, namely we *only* respond with a partial signature σ_1 with $b = 1 - \beta$ for message M where β is determined when the adversary queries the random oracle $H_m(\cdot \| M)$. However, for the random oracle $H_m(\cdot)$, we must answer the queries for both $H_m(0\|M)$ and $H_m(1\|M)$. Therefore, we provide perfect simulations for all the oracles.

Finally, when an adversary outputs a forgery that with a multisignature $\sigma = (\sigma^0, \sigma^1)$ for L_0 and L_1, then either $pk_1 \in L_0$ or $pk_1 \in L_1$ must hold. Without loss of generosity we assume that $pk_1 \in L_{b'}$ for $b' \in \{0, 1\}$. Thus we can conclude that either $b' = \beta$ or $b' = 1 - \beta$ and each case happens with the same probability $\frac{1}{2}$ since β is perfectly hidden from the adversary (not determined by any previous queried information) as stated in Katz and Wang (2003). If $b' = \beta$, we will show that one can extract h^a for given the forgery and (g, g^a, h), solving the Computational Diffie–Hellman problem, otherwise not.

4.2. *Security*

We state the result of security for our multisignature scheme in the following theorem.

Theorem 1. *If there is an algorithm $\mathcal{A}$ in the random oracle model that $(t, q_s, q_h, \ell, \varepsilon)$-breaks our scheme, then there is an algorithm $\mathcal{B}$ that (t', ε')-solves the CDH problem, where*

$$t' = t + \mathcal{O}(2q_h + 2q_s + 2\ell + 1) \cdot T_e,$$

and

$$\varepsilon' = \frac{\varepsilon}{2}.$$

T_e is the running-time of exponentiation in $\mathbb{G}$, q_h and q_s are the bound of two hash queries to H_m, H and signature queries, respectively.

Proof. The strategy of our proof is to construct algorithm $\mathcal{B}$ that solves the computational Diffie–Hellman problem, by utilizing algorithm $\mathcal{A}$ which $(t, q_s, q_h, \ell, \varepsilon)$-breaks our multisignature scheme where $q_h = q_H + q_{H_m}$ is the total number of hash queries.

Suppose $\mathcal{B}$ is given $(g, g^a, h) \in \mathbb{G} \times \mathbb{G} \times \mathbb{G}$ for random numbers $a \longleftarrow \mathbb{Z}_p$, $h \longleftarrow \mathbb{G}$, and asked to find h^a. Without loss of generality, assume user 1 is the honest signer. $\mathcal{B}$ simulates the random oracles $H_m(\cdot)$ and $H(\cdot)$, the signature oracle $\mathcal{O}_{\mathsf{MSign}}(pp, pk^\star, M, L)$ for providing user 1's partial signatures valid under the public key $pk^\star \overset{\text{def}}{=} pk_1 = g^{x_1} \overset{\text{def}}{=} g^a$ with $x_1 \overset{\text{def}}{=} a$ unknown to $\mathcal{B}$.

Setup. $\mathcal{B}$ gives $\mathcal{A}$ the public key $pk^\star = g^a$ and other public parameters $(\mathbb{G}, \mathbb{G}_T, e, H_m(\cdot), g, H(\cdot))$.

$H_m(M\|b)$-*Queries.* $\mathcal{B}$ responds to queries to random oracle $H_m(\cdot)$ as follows: If there is a tuple $(M, \beta, r, r', H_m(M\|0), H_m(M\|1))$ already in the H_m-list which is initially empty, return $H_m(M\|b)$; otherwise, execute the following.

1. Choose two random numbers $r, r' \longleftarrow \mathbb{Z}_p$, a random bit $\beta \longleftarrow \{0,1\}$;
2. Set $H_m(M\|\beta) = g^r$ and $H_m(M\|1-\beta) = h \cdot g^{r'}$;
3. Add $(M, \beta, r, r', H_m(M\|0), H_m(M\|1))$ to the H_m-list;
4. Return $H_m(M\|b)$.

$H(X)$-*Queries.* $\mathcal{B}$ initializes an H-list which only has $(pk^\star, h \cdot g^k, k)$ for $k \xleftarrow{R} \mathbb{Z}_p$, by setting $H(pk^\star) = h \cdot g^k$ and then executes as follows: If $(X, H(X), k)$ has been defined, return $H(X)$; otherwise choose $k \xleftarrow{R} \mathbb{Z}_p$, return $H(X) = g^k$ and add $(X, H(X), k)$ to the H-list.

Note that if the argument of the query cannot be parsed as $X \in \mathbb{G}$, $\mathcal{B}$ simply returns a random element of $\mathbb{G}$, while preserving consistency if the same query has been asked before.

$\mathcal{O}_{\mathsf{MSign}}(pp, pk^\star, M)$-*Queries.* $\mathcal{B}$ proceeds as follows:

1. Find $(M, \beta, r, r', H_m(M\|0), H_m(M\|1))$ in the H_m-list (where $H_m(M\|\beta) = g^r$ and $H_m(M\|1-\beta) = h \cdot g^{r'}$).
   Without loss of generosity, we assume that $\mathcal{A}$ has asked the corresponding hash values; otherwise $\mathcal{B}$ just acts as if it is responding to the hash queries to $H_m(\cdot)$.
2. Randomly choose α from $\mathbb{Z}_p$ and compute

$$\begin{aligned}
s_1 &= (g^a)^{k-r'} \left(hg^{r'}\right)^\alpha \\
&= \frac{g^{ak}}{g^{ar'}} \left(hg^{r'}\right)^\alpha \\
&= \frac{(hg^k)^a}{(hg^{r'})^a} \cdot \left(hg^{r'}\right)^\alpha \\
&= \left(hg^k\right)^a \cdot \left(hg^{r'}\right)^{\alpha-a} \\
&= \left(hg^k\right)^{x_1} \left(hg^{r'}\right)^{\alpha-x_1} \\
&= H(pk_1)^{x_1} H_m(M\|1-\beta)^{\alpha-x_1} \\
&= H(pk_1)^{x_1} H_m(M\|1-\beta)^{\gamma}, \\
t_1 &= g^\gamma = g^\alpha \left(g^{x_1}\right)^{-1} = g^\alpha \left(g^a\right)^{-1}, \\
b_1 &= 1-\beta,
\end{aligned}$$

   where $\gamma = \alpha - a = \alpha - x_1$.
3. Output $\sigma_1 = (s_1, t_1, b_1)$.

Note that $\sigma_1 = (s_1, t_1, b_1)$ is valid because γ is random (due to the randomness of α).

Output h^a. Eventually $\mathcal{A}$ outputs a multisignature forgery $\sigma = (\sigma^0, \sigma^1)$ where $\sigma^0 = (s^0, t^0)$ and $\sigma^1 = (s^1, t^1)$, on message $M^\star$ with respect to $L_0 = (pk_{i_1}, \ldots, pk_{i_k})$ where $b_{i_c} = 0$ for $c = 1, \ldots, k$ and $L_1 = (pk_{j_1}, \ldots, pk_{j_n})$ where $b_{j_d} = 0$ for $d = 1, \ldots, n$ ($\ell = n + k$). Since it is a valid forgery we know $M^\star \notin \mathcal{M}$ (the set of previously queried messages for partial signatures from user 1). Then, it follows that for some $\gamma^0, \gamma^1 \in \mathbb{Z}_p$,

$$s^0 = \left(\prod_{c=1}^{k} H(pk_{i_c})^{x_{i_c}}\right) \cdot H_m\big(M^\star \| 0\big)^{\gamma^0}, \quad t^0 = g^{\gamma^0},$$

and

$$s^1 = \left(\prod_{d=1}^{n} H(pk_{j_d})^{x_{j_d}}\right) \cdot H_m\big(M^\star \| 1\big)^{\gamma^1}, \quad t^1 = g^{\gamma^1}.$$

Then, $\mathcal{B}$ executes the following to compute h^a:

1. Find $(M^\star, \beta, r^\star, r', H_m(M^\star \| 0), H_m(M^\star \| 1))$ in the H_m-list,
2. Let $L_\beta = (pk_1, pk_{f_2}, \ldots, pk_{f_w})$ where $w \in [1, \ell]$ (and $pk_{f_1} = pk_1$).
3. If $pk_1 \notin L_\beta$, abort; otherwise perform additional queries $H(pk_{f_i})$ for $i = 1, \ldots, w$, making sure that $H(pk_{f_i})$ (for $i = 2, \ldots, w$) are defined.
4. Let Δ be the number of public keys $\{pk_{f_i}\}$ such that

   $$pk_{f_i} = pk_1$$

   for $i = 1, \ldots, w$.
5. Compute and output

$$h^a = \left(\big(s^\beta\big) \cdot \big(t^\beta\big)^{-r^\star} \cdot \prod_{pk_{f_i} \in L_\beta \wedge pk_{f_i} \neq pk_1} pk_{f_i}^{-k_{f_i}} \cdot \big(g^a\big)^{-k_1 \Delta}\right)^{\Delta^{-1}}. \tag{1}$$

This is correct because

$$\begin{aligned}
&\big(s^\beta\big) \cdot \big(t^\beta\big)^{-r^\star} \cdot \left(\prod_{pk_{f_i} \in L_\beta \wedge pk_{f_i} \neq pk_1} pk_{f_i}^{k_{f_i}}\right)^{-1} \cdot \big(g^a\big)^{-k_1 \Delta} \\
&= \big(s^\beta\big) \cdot \big(g^{-r^\star}\big)^{\gamma^\beta} \cdot \left(\prod_{pk_{f_i} \in L_\beta \wedge pk_{f_i} \neq pk_1} \big(g^{k_{f_i}}\big)^{x_{f_i}}\right)^{-1} \cdot \big(g^a\big)^{-k_1 \Delta} \\
&= \big(s^\beta\big) \cdot H_m\big(M^\star \| \beta\big)^{-\gamma^\beta} \left(\prod_{pk_{f_i} \in L_\beta \wedge pk_{f_i} \neq pk_1} H(pk_{f_i})^{x_{f_i}}\right)^{-1} \cdot \big(g^a\big)^{-k_1 \Delta} \\
&= \prod_{pk_{f_i} = pk_1} H(pk_{f_i})^{x_{f_i}} \cdot \big(g^a\big)^{-k_1 \Delta}
\end{aligned}$$

$$= \left(\left(hg^{k_1}\right)^{x_1}\right)^{\Delta} \cdot \left(g^{x_1}\right)^{-k_1\Delta}$$
$$= h^{a\Delta}, \tag{2}$$

where $H(pk_{f_i}) = g^{k_{f_i}}$ for $pk_{f_i} \neq pk_1$ and $x_1 = a$.

In the simulation, $\mathcal{B}$ perfectly simulates the random oracle $H_m(\cdot)$, $\mathcal{O}_{\mathsf{MSign}}(\cdot,\cdot,\cdot)$. Therefore, $\mathcal{A}$'s view is identical to that in the real world with random oracles. However, when the adversary $\mathcal{A}$ finally outputs the valid forgery, $\mathcal{B}$ can succeed in outputting h^a solving the Diffie–Hellman problem with probability $\frac{1}{2}$. This is true because the probability of $pk_1 \in L_\beta$ is $\frac{1}{2}$ since β is perfectly hidden from $\mathcal{A}$ as stated in Katz and Wang (2003).

Therefore, $\mathcal{B}$ succeeds in solving the CDH problem (i.e., outputting h^a) with probability

$$\varepsilon_{\mathcal{B}} = \frac{\varepsilon_{\mathcal{A}}}{2}. \tag{3}$$

Finally, for the running-time of $\mathcal{B}$, we take into account the running-time t of $\mathcal{A}$, the exponentiations on hash queries $\mathcal{A}$ made, and the linear number of exponentiations in each signing query and $2\ell + 1$ exponentiation on extracting h^a. This takes time at most $t + \mathcal{O}(2q_h + 2q_s + 2\ell + 1) \cdot T_e$, where T_e is running-time of exponentiation and q_h, q_s are the number of hash queries to H_m and signature queries $\mathcal{O}_{\mathsf{MSign}}(\cdot,\cdot,\cdot)$, respectively.

5. Interesting Feature and Application

An interesting feature of our multisignatures is that our scheme achieves the round optimality of communication. Namely, the generation of multisignatures is non-interactive, this feature is also very useful for applications in some special networks. In Bellare and Neven's multisignature scheme (Bellare and Neven, 2006) and those of Bagherzandi *et al.* (2008), Ma *et al.* (2009), the multisignature generation requires a network to be a complete graph where any two nodes of the network are bidirectionally connected (messages can be sent in either direction). In fact, assuming the signers are connected to each other via point-to-point links over which they can send messages actually is necessary when multisignature generation needs more than one round of interaction.

On the contrary, our schemes can work on the networks with incomplete graphs because the way we generate multisignatures 'non-interactively'. Thus, our scheme does not need to assume that the signers are connected to each other via point-to-point links bidirectionally.

Consider the following scenario described in Fig. 2, suppose there are five signers over the network and each can only send messages to other signers according to the arrows' direction with $A \longrightarrow C$, $B \longleftrightarrow C$, $D \longrightarrow C$, $F \longrightarrow C$, $F \longleftrightarrow G$ and $E \longrightarrow D$. Our question is how A, B, C, D, E, F and G can *generate a multisignature over such a network*. Obviously, those multisignature schemes with multiple rounds of

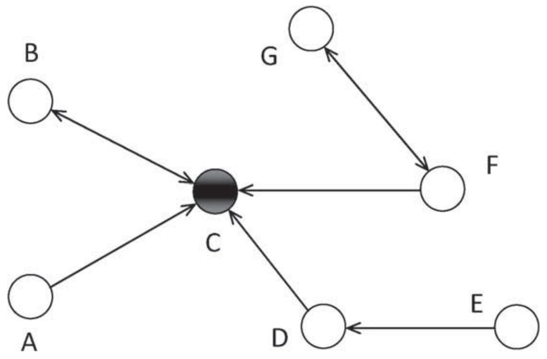

Fig. 2. Network with incomplete graph.

communication can't generate a multisignature over such a network, because after some signers send their messages, they can't get back the responses from the receivers which leads to the failures of executing the interactive protocols. However, by using our scheme the multisignature can be finally output by C (could be a sink in a network) since all the information can finally flow to C over such a network and there are no needs for the information to flow back because our schemes are a non-interactive one.

5.1. *Additional Related Work*

The BGLS scheme was originally proposed for the purpose of aggregate signatures (Boneh *et al.*, 2003), and was later shown Bellare *et al.* (2006) to yield a secure multisignature scheme. However, the BGLS scheme is extremely inefficient in multisignature verification. In principle, sequential aggregate signatures (Lysyanskaya *et al.*, 2004; Neven, 2008) can be used to construct multisignatures as well. However, this approach has drawbacks such as *interactive* signing (signers cannot contribute their partial signatures independently) and expensive verification time. Other aggregate signatures impose special strong assumption on time *synchronization* (Gentry and Ramzan, 2006; Ahn *et al.*, 2010) and strong registration operational assumption for multisignatures as well.

6. Conclusion

Multisignature is an ad hoc signature scheme that is of great use in secure routings. In this paper, we presented an efficient and tightly-secure non-interactive multisignature scheme in the plain public key model. Our scheme reduces the trust assumption on the third party and achieves optimal rounds of communication simultaneously. This scheme, to our best knowledge is the *first* construction which achieves tight security reduction (to the CDH problem) without interactions in the plain public key model. Furthermore, our scheme only needs $\mathcal{O}(1)$ (pairings) in verification through pre-computation. We believe it is one of the most practical schemes currently available in many realistic application scenarios.

Since our scheme works in the random oracle model, it is more interesting to design a non-interactive scheme without random oracles in the plain public key model with tight reduction. We leave it as an open problem.

Acknowledgements. The authors would like to thank anonymous referees for their helpful suggestions on improving the paper. Besides, This work has been supported by the National Natural Science Foundation of China, Grant numbers 61172085, 11061130539, 61103221, 61271118, 61272536 and 61021004.

References

Adams, C., Farrell, S., Kause, T., Monen, T. (2005). *Internet x.509 Public Key Infrastructure Certificate Management Protocol* (cmp).

Ahn, J. H., Green, M., Hohenberger, S. (2010). Synchronized aggregate signatures: new definitions, constructions and applications. In: *ACM Conference on Computer and Communications Security (CCS'10)*, pp. 473–484.

Bagherzandi, A., Jarecki, S. (2008). Multisignatures using proofs of secret key possession, as secure as the Diffie–Hellman problem. In: *Proceedings of the 6th International Conference on Security and Cryptography for Networks (SCN'08)*, pp. 218–235.

Bagherzandi, A., Cheon, J., Jarecki, S. (2008). Multisignatures secure under the discrete logarithm assumption and a generalized forking lemma. In: *Proceedings of the 15th ACM Conference on Computer and Communications Security (CCS'08)*, pp. 449–458.

Barr, K., Asanović, K. (2006). Energy-aware lossless data compression. *ACM Transactions Computer Systems*, 24, 250–291.

Bellare, M., Neven, G. (2006) Multisignatures in the plain public-key model and a general forking lemma. In: *ACM Conference on Computer and Communications Security (CCS'06)*, pp. 390–399.

Bellare, M., Rogaway, P. (1993). Random oracles are practical: a paradigm for designing efficient protocols. In: *ACM Conference on Computer and Communications Security (CCS'03)*, pp. 62–73.

Bellare, M., Namprempre, C., Neven, G. (2006). Unrestricted aggregate signatures. In: *ICALP'07*. Springer, Berlin, pp. 9–13.

Boldyreva, A. (2003). Threshold signature, multisignature and blind signature schemes based on the gap–Diffe–Hellman-group signature scheme. In: *Public Key Cryptograpy-PKC 2003*. Springer, Berlin, pp. 31–46.

Boneh, D., B.Lynn, Shacham, H. (2001). Short signatures from the Weil pairing. In: *Advances in Cryptology-Asiacrypt'01*, LNCS, Vol. 2248. Springer, Berlin, pp. 514–532.

Boneh, D., Gentry, C., Lynn, B., Shacham, H. (2003). Aggregate and verifiably encrypted signatures. In: *EUROCRYPT'03*, pp. 416–432.

Castelluccia, C., Jarecki, S., Kim, J., Tsudik, G. (2006). Secure acknowledgment aggregation and multisignatures with limited robustness. *Computer Network*, 50, 1639–1652.

Galbraith, S., Paterson, K., Smart, N. (2008). Pairings for cryptographers. *Discrete Applied Mathematics*, 156, 3113–3121.

Goldwasser, S., Micali, S., Rivest, R. (1988). A digital signature scheme secure against adaptive chosen-message attacks. *SIAM Journal of Computing*, 17, 281–308.

Gentry, C., Ramzan, Z. (2006). Identity-based aggregate signatures. In: *PKC'06*, pp. 257–273.

Itakura, K., Nakamura, K. (1983). A public key cryptosystem suitable for digital multisignatures. *NEC Research & Development*, 71, 1–8.

Katz, J., Wang, N. (2003). Efficiency improvements for signature schemes with tight security reductions. In: *Proc. 10th ACM Conf. Computer and Communications Security*, pp. 155–164.

Kim, J., Tsudik, G. (2005). Srdp: securing route discovery in dsr. *MobiQuitous*, 247–260.

Lu, R., Lin, X., Shen, X. (2010). Spring: a social-based privacy-preserving packet forwarding protocol for vehicular delay tolerant networks. In: *Proc. IEEE INFOCOM'10*, pp. 14–19.

Lu, S., Ostrovsky, R., Sahai, A., Shacham, H., Waters, B. (2006). Sequential aggregate signatures and multisignatures without random oracles. In: *EUROCRYPT'06*, pp. 465–485.

Lysyanskaya, A., Micali, S., Reyzin, L., Shacham, H. (2004). Sequential aggregate signatures from trapdoor permutations. In: *EUROCRYPT'04*, pp. 74–90.

Micali, S., Ohta, K., Reyzin, L. (2001). Accountable-subgroup multisignatures: extended abstract. In: *ACM Conference on Computer and Communications Security 2001*, pp. 245–254.

Ma, C., Weng, J., Li, Y., Deng, R. (2010). Efficient discrete logarithm based multi-signature scheme in the plain public key model. *Design, Codes and Cryptography*, 54, 121–133.

Neven, G. (2008). Efficient sequential aggregate signed data. In: *EUROCRYPT'08*, pp. 52–69.

Ohta, K., Okamoto, R. (1999). Multisignature schemes secure against active insider attacks. In: *IEICE Transactions on Fundamentals*, E82-A, pp. 21–31.

Qian, H., Xu, S. (2010). Non-interactive multisignatures in the plain public-key model with efficient verification. *Information Processing Letters*, 111, 82–89.

Ristenpart, T., Yilek, S. (2007) The power of proofs-of-possession: securing multiparty signatures against rogue-key attacks. In: *EUROCRYPT'06*, pp. 228–245.

Schaad, J. (2005). *Internet x.509 Public Key Infrastructure Certificate Request Message Format.*

H.F. Qian was awarded a BS degree and a master degree (on algebraic geometry) in Mathematic Department from East China Normal University, China, in 2000 and 2003, respectively, and received the PhD degree in the Department of Computer Science and Engineering, Shanghai Jiao Tong University in 2006. He is currently an associate professor of the Computer Science and Technology Department in East China Normal University, Shanghai. His main research interests include network security, cryptography and algebraic geometry.

X. X. Li received the BS degree in mathematics from Nanjing Normal University, Nanjing, China, in 1997, the MS degree in mathematics from Nanjing University, Nanjing, in 2000, and the PhD degree in computer science and engineering from Shanghai Jiao Tong University, Shanghai, China, in 2006. He is currently an associate professor in the School of Information, East China Normal University, Shanghai. He is also with the State Key Laboratory of Integrated Service Networks, Xidian University, Xi'an, China. His current research interests include lightweight cryptography, applied cryptography, coding theory, disaster recovery, and information security. Dr. Li is involved in several program committees of international conferences.

X. L. Huang received his PhD in computer science from Shanghai Jiao Tong University, Shanghai, China in 2007. He is now an associate professor with the Department of Computer Science and Technology in East China Normal University. He is a member of the IEEE and the ACM. He also serves as a paper reviewer of multiple international journals and academic conferences. Currently, his research interests mainly focus on large-scale distributed systems, peer-to-peer computing and network security.

Viešojo rakto modelio visiškai saugūs neinteraktyvūs bendrieji parašai

Haifeng QIAN, Xiangxue LI, Xinli HUANG

Bendrojo parašo schema įgalina grupę pasirašančių asmenų sukurti bendrą parašą viename dokumente. Tačiau šiuo metu bendrieji parašai turi tokius trūkumus: nepraktiškos raktų derinimo prielaidos, sumažėjęs saugumas ir neefektyvus parašo tikrinimas. Straipsnyje pasiūlytas bendrasis parašas neturi paminėtų trūkumų: jis užtikrina visišką rakto saugumą, išsaugo standartinio Diffie–Hellman'o metodo privalumus ir garantuoja parašo patikrinimo minimalų laiką $\mathcal{O}(1)$.