Non-interactive CDH-Based Multisignature Scheme in the Plain Public Key Model with Tighter Security

Yuan Zhou[1,2], Haifeng Qian[1,⋆], and Xiangxue Li[1]

[1] Department of Computer Science and Technology,
East China Normal University, China
hfqian@cs.ecnu.edu.cn

[2] Network Emergency Response Technical Team/Coordination Center, China

Abstract. A multisignature scheme allows an ad hoc set of users to sign a message so that the resulting single signature certifies that the users endorsed the message. However, all known multisignatures are either at the price of complexity and additional trust of Certificate Authority (CA), or sacrificing efficiency of computation and communication (including both bandwidth and round). This paper proposes a new multisignature scheme with efficient verification in the plain public key model. Our multisignatures enjoys the most desired features: (1) Our plain public key model-based multisignatures do not impose any impractical key setup or PKI requirements; (2) Our multisignature scheme is non-interactive, which saves computation and communication in signature generation; (3) Through pre-computation, our scheme achieves $\mathcal{O}(1)$ verification in the plain public key model; (4) Provable tighter security under the standard CDH assumption ensures high level of security in both practice and theory. Hence, our non-interactive multisignatures are of great use in authentication of routes in networks.

1 Introduction

Multisignatures extend standard digital signatures to allow an ad hoc set of ℓ users to jointly sign a message which are very useful in applications such as contract signing, authentication of routes in mobile networks, distribution of a certificate authority (CA), aggregation of acknowledgements in multicast and secure vehicular communications (refer to [1,2,3,4,5,6,7,8,9,10] for detailed application scenarios).

Due to many practical reasons for applications, we desire that a multisignature scheme might have the following features: (1) the resulting signature is shorter than ℓ separate signatures, even constant for ℓ; (2) Multisignature generation and verification (even key generation) are very fast; (3) The communication overhead including rounds and messages transmitted in multisignature generation should be as fewer as possible (even reduced to a minimum); (4) Trust on

⋆ Corresponding author.

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

In order to generate multisignatures of constant size, the homomorphic properties of arithmetic operations involved in standard signatures are often used. However, these homomorphic properties that enable aggregation of signatures into multisignatures often incurs rogue key attacks for multisignature schemes. In this attack, the adversary chooses its public key as a function of those of honest signers in such a way that it can forge multi-signatures easily[2,5]. For example, rogue key attack succeeds if the verification key (combination of public keys of signers) for multisignature has a fixed formula as $PK = \prod_i^\ell pk_i$ where the adversary can choose $pk_a = g^s \cdot (\prod_i^{\ell-1} pk_i)^{-1}$ and know the corresponding private key s for $PK = pk_a \prod_i^{\ell-1} pk_i = g^s$ (e.g., [11,12,3,13,14] 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.

Early approaches prevent rogue-key attacks for multisignatures at the cost of complexity and expense in the scheme, or using unrealistic and complicate key setup assumptions on the public-key infrastructure (PKI)[1]. These key setup operation assumptions include *dedicated key generation* (DKG), *knowledge of Secret Key* (KOSK), *proof of possesion of private key* (POP) and *the plain public key model* (PPK).

The first effort to prevent rogue key attacks (called DKG) is due to Micali et al. [1]. 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 [3,13]. However, the existing public key infrastructures (PKIs) do not require proofs of knowledge of secret keys [15].

The third one is contributed by [14] and [5,6] where users are required to provide proofs of procession of secret keys (during key registrations) in order to prevent rogue key attacks. Ristenpart and Yilek in [14] showed that the schemes in [3,13] 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 [5,6], but having the multisignature receiver verify the POP message (together with verification of PKI certificates)[16], instead of the CAs during the key registration.

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 provided such a scheme in the plain (public-key) model [2], the others are shown in [5,17,18]. In the plain (public-key) model, there is no dedicated key generation (DKG) procedures, or well-formedness proofs accompanied to the public keys and a party can obtain a certificate on any arbitrary key without any proof.

1.2 Multisignatures in the PPK Model

So far as we know, there are only very few multisignature schemes with provable security in the plain public key model and all these multisignatures are proved secure in the random oracle model [19].

The first multisignature in such a model was proposed by Boneh et al [18] (for brevity BGLS), which is implied by the construction so-called aggregate signature [20]. 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, making it extremely impractical.

Under the DL assumption Bellare and Neven [2], provided a concrete multisignature scheme (denoted BN) with rather an efficient verification in the PPK model. Followed the idea of [2], a lot of *interactive* multisignature schemes are proposed [5,17] 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 [21] 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 [22]. 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 reduction of the QX scheme is even looser than that of the BN scheme[2]. 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 reduction.

1.3 Our Contributions

We present a non-interactive multisignature scheme, which operates in the plain public-key model and is proven 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 two pairings through

Table 1. Multisignature Comparison

Scheme	Assump.	Rounds	Sign	Comm. Cost	Verify	σ Size
BN [2]	$<$ DL	3	1 exp	$\lvert\mathbb{G}'\rvert + \lvert q\rvert + l_0$	1 $\mathsf{mexp}^{(\ell+1)}$	$\lvert\mathbb{G}'\rvert + \lvert q\rvert$
BGLS [18,20]	$\approx$ CDH	1	1 exp	$\lvert\mathbb{G}\rvert$	$(\ell+1)$ pr	$\lvert\mathbb{G}\rvert$
QX[22]	$<<$ CDH	1	1 exp	$\lvert\mathbb{G}\rvert$	$2\mathsf{pr} + \mathsf{mexp}^{(\ell+1)}$	$\lvert\mathbb{G}\rvert$
Ours	$\approx$ CDH	1	2 exp	2 $\lvert\mathbb{G}\rvert$	2 pr	2 $\lvert\mathbb{G}\rvert$

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 $e : \mathbb{G} \times \mathbb{G} \to \mathbb{G}_T$ 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. " $<<$ " and " $<$ " means very loose security and loose security, respectively. " $\approx$ " means close security.

pre-computation (cf. Table 1)[1]. The improvement in verification time is substantial because one pairing costs about 6-20 exponentiations [2]. 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.

On the other hand, our scheme also enjoys a tighter security proof, compared 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 $q_s \cdot \varepsilon$ break our multisignature scheme where ε is the upper bound of probability for breaking the underlying cryptographic problem. While in the BN scheme the corresponding upper bound is roughly $\sqrt{q_h \cdot \varepsilon}$. Let $q_h = 2^{60}$, $q_s = 2^{30}$, $\varepsilon = 2^{-80}$, our scheme ensures 50 bits of security level, while the BN scheme (or the QX scheme) ensures at most 10 bits of security level in practice. Compared with the QX scheme, our scheme also saves the mult-exponentiation in verification of a multisignature. Detailed comparisons amongst our scheme, the BN scheme, the BGLS scheme and the QX scheme are depicted in Table 1.

[1] Since the signers' public keys are used as prefixes to the corresponding message in the BGLS scheme, verification of the BGLS multisignature needs $(\ell + 1)$ pairings necessarily.

Our technique for dealing with the rogue key attack in the plain public-key model is quite different from those of [2]. Instead of using a dynamic key with respective to messages [2] 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.

The structure of our multisignatures are similar to that of the WMS scheme [13] and its variant [14] which enables us to reduces the communication rounds of multisignatures to optimal (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 [13] 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 almost as secure as the Computational Diffie-Hellman (CDH) problem.

2 Preliminaries

We recall the basic definitions for multisignatures, then review the cryptographic complexity assumption in this section. We the notations of [22] in the following.

2.1 Definitions of Multisignatures

The definition of interactive multisignatures in the plain public-key model was first formalized in [2] with ℓ signers, each having as input its own public and private keys as well as the public keys of the other signers. The signers interact via a protocol to generate a multisignature.

In this paper, we consider the following 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.

A non-interactive multisignature scheme $\mathsf{MS} = (\mathsf{Setup}, \mathsf{Gen}, \mathsf{MSign}, \mathsf{MVf})$ consists of three algorithms and one protocol (adapted from [2]):

- $\mathsf{Setup}(1^\lambda)$: 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 of public parameters pp, outputs (an honest) signer i's private/public key pair (sk_i, pk_i).
- $\mathsf{MSign}(pp, \{sk_i\}, M, L)$: Given public parameters pp, a message M and a (multi)set $L = (pk_1, \dots, pk_\ell)$ of purported signers (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), signer i uses its private key sk_i to compute a partial signature σ_i and then broadcasts σ_i. Given σ_j for $j = 1, \dots, \ell$, any one can obtain a multisignature σ with respect to the public keys on L.

– $\mathsf{MVf}(pp, M, \sigma, L)$: Given $L = (pk_1, \ldots, pk_\ell)$, 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.

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

1. $pp \leftarrow \mathsf{Setup}(1^\lambda)$; $(pk^\star, sk^\star) \leftarrow \mathsf{Gen}(pp)$; $\mathcal{M} \leftarrow \phi$ where $\mathcal{M}$ is the set of pairs (M, L) previously queried for signatures.
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 $L = (pk_1, \ldots, pk_n)$ for any n, as long as $pk^\star$ appears at least once in L.
   - 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 $L^\star = (pk_1, \ldots, pk_\ell)$. If
$$\mathsf{MVf}(pp, M^\star, \sigma^\star, L^\star) = 1$$
and
$$(pk^\star \in L^\star) \wedge (M^\star, L^\star) \notin \mathcal{M} = 1$$
then return 1, otherwise 0.

Fig. 1. Experiment for security definition

Intuitively, security of multisignature scheme requires that it is infeasible for adversary $\mathcal{A}$ to forge a multisignature with respect to a new message which extends the classical security notion of digital signature scheme known as existential unforgeability under adaptively chosen-message attacks [23]. Following [2,5], we can assume there is a single honest signer. Then, unforgeability must hold despite that in the plain public-key model, the adversary can corrupt all other signers and choose their public keys arbitrarily (even registering the public key of the honest user as their own public keys), and can interact with the honest signer in any number of concurrent signing instances before outputting its forgery.

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 Figure 1 outputs 1.

We say the adversary $(t, q_s, q_h, \ell, \varepsilon)$-breaks multisignature scheme MS, if it in time t, after q_s signature queries or q_s invocations of $\mathsf{MSign}(\cdot,\cdot,\cdot,\cdot)$, 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

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

If there is no such adversary, 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 $\mathbb{G}$, $e : \mathbb{G} \times \mathbb{G} \to \mathbb{G}_T$ 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., $e : \mathbb{G}_1 \times \mathbb{G}_2 \to \mathbb{G}_T$), that allows for short signatures without side-effect. Our scheme is also adaptable for such a setting. For more details, refer to [24,25].

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

Definition 1 (CDH). *Given $(g, g^a, h) \in \mathbb{G} \times \mathbb{G} \times \mathbb{G}$ for some $a \overset{R}{\leftarrow} \mathbb{Z}_p$ and $h \overset{R}{\leftarrow} \mathbb{G}$, find $h^a \in \mathbb{G}$.*

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

$$\mathrm{Adv}^{\mathrm{cdh}}_{\mathcal{A}} \overset{def}{=} \Pr\left[h^a \leftarrow \mathcal{A}(g, g^a, h) : g \overset{R}{\leftarrow} \mathbb{G}, a \overset{R}{\leftarrow} \mathbb{Z}_p, h \overset{R}{\leftarrow} \mathbb{G}\right].$$

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 $\mathrm{Adv}^{\mathrm{cdh}}_{\mathcal{A}} \geq \varepsilon$. 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 the Waters-like signature (e.g., $\sigma = (sk \cdot H(m)^r, g^r)$) to construct multisignatures, however this scheme is different from the WMS multisignature scheme in [13] 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 additional protocols to get public key certificated).

Our scheme does not require the signers to share a common random or commit [2,17] during multisignature generation as well, otherwise it seems impossible to reduce the round of signing to minimum. Although our construction is similar to the WMS scheme [13], security of multisignatures in the plain public key model does not rely on the trust of the CAs because of the plain public key model. While security of the WMS scheme are proved in the KOSK model that means security also relies on the trust of the CAs. 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 $H_m : \{0,1\}^* \to \mathbb{G}$ and $H : \mathbb{G} \to \mathbb{G}$ 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 $x_i \overset{R}{\leftarrow} \mathbb{Z}_p$ and its private/public key pair is (sk_i, pk_i) where

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

$\mathsf{MSign}(pp, \{sk_i\}, M, L)$**:** On input pp, message M and $L = (pk_1, \ldots, pk_\ell)$, user i $(1 \le i \le \ell)$ executes the following:

1. Pick a random $r_i \overset{R}{\leftarrow} \mathbb{Z}_p$ and compute

$$s_i \leftarrow sk_i \cdot H_m(M||L)^{r_i} \quad \text{and} \quad t_i \leftarrow g^{r_i}.$$

2. Broadcast $\sigma_i = (s_i, t_i)$ 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 as follows:

$$\sigma = \bigotimes_{1 \le j \le \ell} \sigma_j = (\prod_{1 \le j \le \ell} s_i, \prod_{1 \le j \le \ell} t_i).$$

$\mathsf{MVf}(pp, M, \sigma, L)$**:** Given pp, $L = (pk_1, \ldots, pk_\ell)$, message M, and an alleged multisignature $\sigma = (s, t)$, a verifier accepts the multisignature if

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

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

Theorem 1. *Our multisignature scheme is correct. Namely any multisignature generated by legal signers can be verified.*

Proof. The scheme is *correct* because

$$\begin{aligned} e(s,g) &= e\left(\left(\prod_{i=1}^{\ell} H(pk_i)^{x_i}\right) \cdot H_m(M||L)^r, g\right) \\ &= e(H_m(M||L)^r, g) \cdot \prod_{i=1}^{\ell} e(H(pk_i), g^{x_i}) \\ &= e(H_m(M||L), t) \cdot \prod_{i=1}^{\ell} e(H(pk_i), pk_i) \\ &= e(H_m(M||L), t) \cdot \prod_{pk_i \in L} A_i, \end{aligned}$$

where

$$r = \sum_{i=1}^{\ell} r_i \quad \text{and} \quad t = g^r.$$

Remark 2. In our scheme, each signer generates its own randomness. Without any shared common random generator, we reduce the round of communication to minimum. On the other hand, our multisignature generation is also quite simple and straightforward: each signer produces its Waters-like signature $\sigma = (sk \cdot H(m)^r, g^r)$ on the message, then the multisignature is just the component-wise product of these signatures.

Remark 3. To compute $A_i = e(H(pk_i), pk_i)$ 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 multisignature verification since A_i is independent from messages. Comparing with those in the plain public key model, our multisignature scheme is the most efficient one in verification since our scheme achieves $\mathcal{O}(1)$-verification (respective to pairing computations).

Remark 4. 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. Indeed, we only need replace "$M||L$" by "M" in our signing protocol to yield such a scheme. However, it security only prevents forgery on a new message [13,14], not a pair of message/signers.

4 Security Analysis

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

Theorem 2. *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*

$$t' = t + \mathcal{O}(q_h + 3q_s + \ell + 2)T_e \quad \textit{and} \quad \varepsilon' = \frac{\varepsilon}{\mathrm{e}(q_s + 1)},$$

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, and the mathematical constant e *is the base of the natural logarithm.*

Proof. The strategy 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 $a \overset{R}{\leftarrow} \mathbb{Z}_p$, $h \overset{R}{\leftarrow} \mathbb{G}$, 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 $pk^\star \overset{def}{=} pk_1 = g^{x_1} \overset{def}{=} g^a$ with $x_1 \overset{def}{=} a$ unknown to $\mathcal{B}$.

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

$H_m(M||L)$**-Queries.** $\mathcal{B}$ responds to queries to random oracle $H_m(\cdot)$ as follows:

1. If there is a tuple $(M||L, b, r, H_m(M||L))$ in the H_m-list which is initially empty, return $H_m(M||L)$; otherwise, execute the following.
2. Choose $r \overset{R}{\longleftarrow} \mathbb{Z}_p$, a bit $b \overset{\delta}{\longleftarrow} \{0, 1\}$ such that b takes 1 with probability δ that will be specified later, return $H_m(M||L) = h^b \cdot g^r$ and add $(M||L, b, r, H_m(M||L))$ to the H_m-list.

$H(X)$**-Queries.** $\mathcal{B}$ initializes an H-list which only has $(pk^\star, h \cdot g^k, k)$ for $k \overset{R}{\longleftarrow} \mathbb{Z}_p$, by setting $H(pk^\star) = h \cdot g^k$ and then executes as follows:

1. If $(X, H(X), k)$ has been defined, return $H(X)$; otherwise do the following.
2. Choose $k \overset{R}{\longleftarrow} \mathbb{Z}_p$, 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.

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

1. If $pk^\star \notin L$, return $\perp$ and abort; otherwise find $(M||L, b, r, H_m(M||L))$ (where $H_m(M||L) = h^b \cdot g^r$) in the H_m-list. (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. If $b = 0$, abort; otherwise randomly choose α from $\mathbb{Z}_p$ and compute

$$\begin{aligned} s_1 &= (g^a)^{k-r}(hg^r)^\alpha \\ &= (hg^k)^{x_1}(hg^r)^{\alpha - x_1} \\ &= H(pk_1)^{x_1} H_m(M||L)^{\alpha - x_1}, \\ t_1 &= g^{\alpha - x_1} = g^\alpha (g^{x_1})^{-1} = g^\alpha (g^a)^{-1}. \end{aligned}$$

3. Output $\sigma_1 = (s_1, t_1)$.

Note that $\sigma_1 = (s_1, t_1)$ is valid when we set randomness $r' = \alpha - a$.

Output h^a. Eventually $\mathcal{A}$ outputs a multisignature forgery $\sigma = (s, t)$ on message $M^\star$ with respect to $L = (pk^\star = pk_1, pk_2, \ldots, pk_\ell)$, where $(M^\star, L) \notin \mathcal{M}$ (the set of previously queried messages and the signing group for partial signatures from user 1). Since σ is valid, we know that for some $d \in \mathbb{Z}_p$,

$$s = \left(\prod_{i=1}^{\ell} H(pk_i)^{x_i} \right) \cdot H_m(M^\star||L)^d, \quad t = g^d.$$

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

1. Perform additional queries $H(pk_i)$ for $1 \le i \le \ell$, making sure that $H(pk_i)$ for $2 \le i \le \ell$ is defined.
2. Let J be the index such that $pk_i = pk_1$ and Δ be the number of such keys for $1 \le i \le \ell$.
3. Find $(M^\star||L, b^\star, r^\star, H_m(M^\star||L))$ in the H_m-list.
4. If $b^\star = 1$, abort; otherwise $b^\star = 0$ compute and output

$$h^a = \left(s \cdot t^{-r^\star} \cdot \prod_{pk_i \neq pk_1} pk_i^{-k_i} \cdot (g^a)^{-k_1 \Delta} \right)^{\Delta^{-1}} \tag{4.1}$$

because

$$\begin{aligned} & s \cdot t^{-r^\star} \cdot \left(\prod_{pk_i \neq pk_1} pk_i^{k_i} \right)^{-1} \cdot (g^a)^{-k_1 \Delta} \\ &= s \cdot \left(g^{-r^\star} \right)^d \cdot \left(\prod_{pk_i \neq pk_1} (g^{k_i})^{x_i} \right)^{-1} \cdot (g^a)^{-k_1 \Delta} \\ &= s \cdot H_m(M^\star||L)^{-d} \left(\prod_{pk_i \neq pk_1} H(pk_i)^{x_i} \right)^{-1} \cdot (g^a)^{-k_1 \Delta} \\ &= \prod_{pk_i = pk_1} H(pk_i)^{x_i} \cdot (g^a)^{-k_1 \Delta} \\ &= \left((hg^{k_1})^{x_1} \right)^{\Delta} \cdot (g^{x_1})^{-k_1 \Delta} \\ &= h^{a\Delta}. \end{aligned}$$

where $H(pk_i) = g^{k_i}$ for $pk_i \neq pk_1$ and $x_1 = a$.

Although $\mathcal{B}$ perfectly simulates the random oracle $H_m(\cdot)$, in the simulation of $\mathcal{O}_{\mathsf{MSign}}(\cdot,\cdot,\cdot,\cdot)$ $\mathcal{B}$ succeeds with probability δ, namely the probability that $b_i = 1$ for each $i = 1, \ldots, q_s$. Thus $\mathcal{B}$ doesn't abort in the simulation of the signature oracle with probability

$$\Pr\left[\bigwedge_{i=1}^{q_s}(b_i = 1)\right] = \prod_{i=1}^{q_s}\Pr\left[b_i = 1\right] = \delta^{q_s}.$$

When $\mathcal{A}$ outputs a valid forgery, $\mathcal{B}$ will abort if $b^\star = 1$. Therefore, $\mathcal{B}$ succeeds with probability that $b^\star = 0$ at that time (i.e., $1 - \delta$). Let $\mathcal{E}$ denote the event that $\mathcal{A}$ outputs a valid forgery, and $\mathcal{E}_{\mathsf{suc}}|\mathcal{E}$ denote the event $\mathcal{B}$ succeeds in solving the CDH problem (i.e., outputting h^a) under the condition that $\mathcal{E}$ happens. We have $\Pr[\mathcal{E}_{\mathsf{suc}}|\mathcal{E}] = (1-\delta)\delta^{q_s}$. Then,

$$\begin{aligned}\varepsilon_{\mathcal{B}} &= \Pr[\mathcal{E} \bigwedge \mathcal{E}_{\mathsf{suc}}] \\ &= \Pr[\mathcal{E}] \cdot \Pr[\mathcal{E}_{\mathsf{suc}}|\mathcal{E}] \\ &= \varepsilon \cdot (1-\delta)\delta^{q_s}\end{aligned}$$

Let $f(\delta) = (1-\delta)\delta^{q_s}$. Since f is maximal at $\delta_0 = \frac{q_s}{q_s+1}$, we know $f(\delta_0) \geq \frac{1}{\mathrm{e}\cdot(q_s+1)}$, where the mathematical constant e is the base of the natural logarithm. Therefore,

$$\varepsilon_{\mathcal{B}} \geq \frac{\varepsilon'}{\mathrm{e}\cdot(q_s+1)}.$$

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 $\ell + 2$ exponentiation on extracting h^a. This takes time at most $t + \mathcal{O}(q_h + q_s + \ell + 2) \cdot T_e$, 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 $\mathcal{O}_{\mathsf{MSign}}(\cdot,\cdot,\cdot,\cdot)$, respectively.

5 Additional Related Work

The BGLS scheme was originally proposed for the purpose of aggregate signatures [18], and was later shown (through a new analysis [20]) to be a secure multisignature scheme as well. However, the BGLS scheme is extremely inefficient in multisignature verification. While in principle sequential aggregate signatures (e.g., [26,27]) can be used to construct multisignatures, 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* [28,29] which also will impose additional operational assumption for multisignatures.

6 Conclusion

We presented an efficient non-interactive multisignature scheme in the plain public key model. Our scheme not only reduces the trust assumption on the third party and but also achieves optimal rounds of communication. This scheme, to our best knowledge enjoys a tighter security proof, comparing with non-interactive construction in the plain public key model. Furthermore, our scheme only needs $\mathcal{O}(1)$ (pairings) in verification through pre-computation. We believe it is amongst the most practical schemes currently available in many realistic application scenarios.

Acknowledgements. This work has been supported by the National Natural Science Foundation of China, Grant No. 60873217, 61172085, 11061130539 and 61021004.

References

1. Micali, S., Ohta, K., Reyzin, L.: Accountable-Subgroup Multisignatures: Extended Abstract. In: Eighth ACM Conference on Computer and Communications Security, pp. 245–254. ACM Press, New York (2001)
2. Bellare, M., Neven, G.: Multisignatures in the Plain Public-Key Model and a General Forking Lemma. In: 13th ACM Conference on Computer and Communications Security, pp. 390–399. ACM Press, New York (2006)
3. Boldyreva, A.: Threshold Signatures, Multisignatures and Blind Signatures Based on the Gap-Diffie-Hellman-Group Signature Scheme. In: Desmedt, Y.G. (ed.) PKC 2003. LNCS, vol. 2567, pp. 31–46. Springer, Heidelberg (2002)
4. Kim, J., Tsudik, G.: Srdp: Securing Route Discovery in DSR. In: Second Annual International Conference on Mobile and Ubiquitous Systems: Networking and Services, pp. 247–260. IEEE Press, Los Alamitos (2005)
5. Bagherzandi, A., Cheon, J., Jarecki, S.: Multisignatures Secure under the Discrete Logarithm Assumption and a Generalized Forking Lemma. In: The 15th ACM Conference on Computer and Communications Security, pp. 449–458. ACM Press, New York (2008)
6. Bagherzandi, A., Jarecki, S.: Multisignatures Using Proofs of Secret Key Possession, as Secure as the Diffie-Hellman Problem. In: Ostrovsky, R., De Prisco, R., Visconti, I. (eds.) SCN 2008. LNCS, vol. 5229, pp. 218–235. Springer, Heidelberg (2008)
7. Castelluccia, C., Jarecki, S., Kim, J., Tsudik, G.: Secure Acknowledgment Aggregation and Multisignatures with Limited Robustness. Comput. Netw. 50, 1639–1652 (2006)
8. Lin, X., Sun, X., Ho, P.H., Shen, X.: Gsis: A Secure and Privacy Preserving Protocol for Vehicular Communications. IEEE Trans. on Vehicular Tech. 56, 3442–3456 (2007)
9. Lu, R., Lin, X., Zhu, H., Ho, P.H., Shen, X.: Ecpp: Efficient Conditional Privacy Preservation Protocol for Secure Vehicular Communications. In: The 27th Conference on Computer Communications IEEE INFOCOM 2008, pp. 14–18. IEEE Press, Los Alamitos (2008)
10. Lu, R., Lin, X., Shen, X.: Spring: A social-based Privacy-Preserving Packet Forwarding Protocol for Vehicular Delay Tolerant Networks. In: The 29th Conference on Computer Communications, IEEE INFOCOM 2010, pp. 14–19. IEEE Press, Los Alamitos (2010)

11. Itakura, K., Nakamura, K.: A Public Key Cryptosystem Suitable for Digital Multisignatures. NEC Research & Development 71, 1–8 (1983)
12. Ohta, K., Okamoto, R.: Multisignature Schemes Secure Against Active Insider Attacks. IEICE Transactions on Fundamentals E82-A, 21–31 (1999)
13. Lu, S., Ostrovsky, R., Sahai, A., Shacham, H., Waters, B.: Sequential Aggregate Signatures and Multisignatures Without Random Oracles. In: Vaudenay, S. (ed.) EUROCRYPT 2006. LNCS, vol. 4004, pp. 465–485. Springer, Heidelberg (2006)
14. Ristenpart, T., Yilek, S.: The Power of Proofs-of-Possession: Securing Multiparty Signatures Against Rogue-Key Attacks. In: Naor, M. (ed.) EUROCRYPT 2007. LNCS, vol. 4515, pp. 228–245. Springer, Heidelberg (2007)
15. Adams, C., Farrell, S., Kause, T., Monen, T.: Internet X.509 Public Key Infrastructure Certificate Management Protocol, cmp (2005)
16. Schaad, J.: Internet X.509 Public Key Infrastructure Certificate Request Message Format (2005)
17. Ma, C., Weng, J., Li, Y., Deng, R.: Efficient Discrete Logarithm Based Multi-Signature Scheme in the Plain Public Key Model. Des. Codes Cryptography 54, 121–133 (2010)
18. Boneh, D., Gentry, C., Lynn, B., Shacham, H.: Aggregate and Verifiably Encrypted Signatures. In: Biham, E. (ed.) EUROCRYPT 2003. LNCS, vol. 2656, pp. 416–432. Springer, Heidelberg (2003)
19. Bellare, M., Rogaway, P.: Random Oracles Are Practical: a Paradigm for Designing Efficient Protocols. In: 10th ACM Conference on Computer and Communications Security, pp. 62–73. ACM Press, New York (1993)
20. Bellare, M., Namprempre, C., Neven, G.: Unrestricted Aggregate Signatures. In: Arge, L., Cachin, C., Jurdziński, T., Tarlecki, A. (eds.) ICALP 2007. LNCS, vol. 4596, pp. 411–422. Springer, Heidelberg (2007)
21. Barr, K., Asanović, K.: Energy-Aware Lossless Data Compression. ACM Trans. Comput. Syst. 24, 250–291 (2006)
22. Qian, H., Xu, S.: Non-Interactive Multisignatures in the Plain Public-Key Model with Efficient Verification. Inf. Process. Lett. 111, 82–89 (2010)
23. Goldwasser, S., Micali, S., Rivest, R.: A Digital Signature Scheme Secure Against Adaptive Chosen-Message Attacks. SIAM Journal of Computing 17, 281–308 (1988)
24. Boneh, D., Lynn, B., Shacham, H.: Short Signatures from the Weil Pairing. In: Boyd, C. (ed.) ASIACRYPT 2001. LNCS, vol. 2248, pp. 514–532. Springer, Heidelberg (2001)
25. Galbraith, S., Paterson, K., Smart, N.: Pairings for Cryptographers. Discrete Applied Mathematics 156, 3113–3121 (2008)
26. Lysyanskaya, A., Micali, S., Reyzin, L., Shacham, H.: Sequential Aggregate Signatures from Trapdoor Permutations. In: Cachin, C., Camenisch, J.L. (eds.) EUROCRYPT 2004. LNCS, vol. 3027, pp. 74–90. Springer, Heidelberg (2004)
27. Neven, G.: Efficient Sequential Aggregate Signed Data. In: Smart, N.P. (ed.) EUROCRYPT 2008. LNCS, vol. 4965, pp. 52–69. Springer, Heidelberg (2008)
28. Gentry, C., Ramzan, Z.: Identity-Based Aggregate Signatures. In: Yung, M., Dodis, Y., Kiayias, A., Malkin, T. (eds.) PKC 2006. LNCS, vol. 3958, pp. 257–273. Springer, Heidelberg (2006)
29. Ahn, J.H., Green, M., Hohenberger, S.: Synchronized Aggregate Signatures: New Definitions, Constructions and Applications. In: 17th ACM Conference on Computer and Communications Security, pp. 473–484. ACM Press, New York (2010)