Multisignatures Using Proofs of Secret Key Possession, as Secure as the Diffie-Hellman Problem*

Ali Bagherzandi and Stanisław Jarecki

Department of Computer Science,
University of California, Irvine
{zandi,stasio}@ics.uci.edu

Abstract. A multisignature scheme allows a group of n players to produce a short string which is equivalent to n separate signatures on the same message. Assuming the Random Oracle Model (ROM), the aggregate signature schemes of Boneh et al. [BGLS03] and Bellare and Neven [BN06] provide multisignatures secure in the standard public key setting, but their multisignature verification algorithms involve respectively $O(n)$ bilinear maps and $O(n)$ exponentiations. Ristenpart and Yilek [RY07] recently showed two multisignature schemes relying on groups with bilinear maps, with just $O(1)$ bilinear maps in multisignature verification, which are secure if each public key is accompanied by so-called "proof of (secret key) possession" (POP). We show how to achieve secure multisignatures in the POP model using any group where CDH or DDH problems are hard. Both schemes have multisignature verification with $O(1)$ exponentiations, and their POP messages take $O(1)$ group elements and require $O(1)$ exponentiations to verify. Moreover, the security of the proposed schemes is *tightly* related to the CDH and DDH problems, in ROM.

1 Introduction

A multisignature scheme allows a group of n players to sign a common message so that instead of n separate signatures the players produce a short string which can be verified against the set of the public keys of the participating players. Such scheme is interesting if the resulting string is shorter than n separate signatures and/or the verification time is faster than n separate signature verifications. Applications of multisignatures include scenarios where the number of signers is moderate, like co-signing, distribution of certificate authorities, or aggregation of PKI certificate chains. However, multisignatures can potentially be useful also in very large groups of signers, *e.g.* for aggregation of acknowledgements in response to a broadcast.

Rogue Key Attacks and the KOSK Assumption. Multisignature schemes are possible because of homomorphic properties of arithmetic operations involved in standard signatures. For example, a BLS signature [BLS04] on message m under public key $y_i = g^{x_i}$ is $\sigma_i = H(m)^{x_i}$, i.e. a value s.t. $(g, y_i, H(m), \sigma_i)$ is a DDH tuple. A corresponding multisignature can be created as $\sigma = \prod_{i=1}^{n} \sigma_i$, and it can be verified under the

* Research supported by NSF CyberTrust Grant #0430622.

R. Ostrovsky, R. De Prisco, and I. Visconti (Eds.): SCN 2008, LNCS 5229, pp. 218–235, 2008.

combined public key $y = \prod_{i=1}^{n} y_i$, because $(g, y, H(m), \sigma)$ is also a DDH tuple. Unfortunately, the same homomorphic properties which enable aggregation of signatures into multisignatures can enable a "rouge key attack" on such schemes. For example, the above scheme is insecure because an adversary who picks $y_2 = g^x/y_1$ for some existing key y_1 and any x can use $x = DL(g, y_1 * y_2)$ to issue multisignatures on behalf of key set $\{y_1, y_2\}$. Indeed, as Micali et al. [MOR01] point out, many proposed multisignature schemes are vulnerable to such rouge key attacks, *e.g.* [LHL94, Har94], or their security requires trusted generation of each key, *e.g.* [OO91, OO99]. Interestingly, rouge key attackers usually do not know the private key corresponding to the rogue public key. Indeed, under the discrete logarithm assumption it is provably hard to compute the private key x_2 *s.t.* $y_2 = g^{x_2}$ in the above attack. This observation led Micali, Ohta, and Reyzin [MOR01] to construct the first multisignature scheme secure without assuming trusted key generation. However, that scheme requires all potential signers to engage in an interactive initialization protocol in which every player proves knowledge of its secret key to all others, and such initialization procedure does not tolerate dynamic groups and does not scale well to large groups. One way to remove this initialization procedure is to assume the so-called *knowledge of secret key* (KOSK) assumption [Bol03] on key registration process: The KOSK assumption states that if an adversary registers a public key then the adversary's algorithm must also explicitly output a corresponding secret key. Two secure multisignature schemes were proposed under this assumption using bilinear maps, by Boldyreva [Bol03] in ROM and by Lu et al. [LOS$^+$06] in the standard model (*i.e.* without ROM).

Multisignatures in the Key Registration Model. One way to realize the KOSK assumption is to employ so-called *Key Registration Model* (KR) for Public Key Infrastructure (PKI), introduced in the context of multisignatures by Ristenpart and Yilek [RY07]. In the KR model for PKI, a CA issues a certificate on a key only if its owner passes a special key registration protocol. For example, the PKCS#10 [PKC00] standard for CA operation asks the user to sign a challenge message under its public key. This challenge-signature pair is called a *proof of possession* of the secret key (POP) in PKCS#10, but we'll use this term more broadly, for any user-generated string verified by either the CA or by multisignature verifiers (see the "Key Verification" model below). The intuitive goal of the POP mechanism in PKCS#10 was to assure that someone has access to the secret key corresponding to the public key being certified, but this mechanism does not implement the KOSK assumption in general. Indeed, Ristenpart and Yilek [RY07] showed that the schemes of [Bol03, LOS$^+$06] are insecure in the KR model if key registration is implemented with POPs of PKCS#10. Nevertheless, [RY07] also showed that using a slight variant of the same POP mechanism the schemes of [Bol03, LOS$^+$06] yield secure multisignature schemes in the KR model, relying on bilinear maps.

Alternatively, one can realize the KOSK model by implementing POPs with concurrently secure zero-knowledge proofs of knowledge (ZKPK) of a secret key. Such ZKPK's can be achieved in ROM by the results of Fischlin [Fis05] using $O(\log \kappa)$ group elements where κ is the security parameter. Combined with the multisignature protocol of [MOR01], this implies a multisignature scheme in the KR model secure under the DL assumption. However, non-constant-sized POPs are less practical if POP messages are verified by multisignature receivers instead of by the CA's (see the "Key

Registration vs. Key Verification" below). Moreover, due to the heavy use of the forking lemma in the reduction of [MOR01], the exact security of this scheme is not optimal.

Multisignatures in the Plain Public Key Model. One of the drawbacks of multisignatures in the KR model is that they require modifications to the current operation of the CA's. (In addition to imposing non-standard trust requirements on CA's, as we argue below.) It is therefore highly interesting to provide multisignature schemes which are secure in the *plain* setting where no special registration process is assumed for public keys. The first such scheme is implied in ROM by an aggregate signature scheme of Boneh et al. [BGLS03], with the analysis extended by Bellare et al. [BNN07]), but its multisignature verification algorithm requires $O(n)$ bilinear map operations. Bellare and Neven recently proposed a multisignature secure in the plain setting which does bilinear maps [BN06]. While this scheme has a significantly smaller cost of multisignature verification, it still requires $O(n)$ exponentiations: The way [BN06] avoid KOSK and KR models is by using independent challenges in the proofs of knowledge of discrete logarithm performed by each player in the multisignature generation protocol. However, the multisignature verification operation then becomes a multi-exponentiation on n public keys and n different exponents, and to the best of our knowledge the cost of such multiexponentiation is still $O(n)$ the cost of a single exponentiation.

Key Registration vs. Key Verification. The fact that current multisignatures in the plain setting have slower verification than current schemes secure in the KR model motivates looking closer at the KR model. For example, to the best of our knowledge it has not been previously observed that the Key Registration model requires non-standard trust assumptions among the PKI participants. Consider an application of a multisignature scheme, where a multisignature is formed by some users certified by CA's trusted by the multisignature verifier, and some certified by CA's who are unknown to this verifier. Assume that the verifier is interested in checking whether or not the message was signed by all the users of the first type but does not care if users of the second type have also contributed to the multisignature. An example is a petition signed by individuals whose public keys are certified by different CA's, some widely known and trusted, some less so. If a multisignature scheme secure in the KR model is used then the verifier cannot conclude that the message was signed by the users she cares about, certified by the CA's she recognizes and trusts as long as a single participant in the multisignature is certified by a CA which she does not recognize and/or trust. This is because the scheme provides no security guarantees if the prescribed key registration procedure, e.g. POP verification, is not followed with regards to even a single key participating in the multisignature. This imposes a limitation on the use of multisignatures compared to standard signatures or multisignatures secure in the plain setting, since in either of the last two cases the verifier can decide if the users she cares about signed the petition whether or not it was also signed by keys certified by unknown or suspect CA's.

We propose to remove this limitation in the usage of multisignatures secure in the KR model by considering an alternative mode of PKI operation which we call the *Key Verification* (KV) Model. In the KV model each private key owner also produces a POP string, but instead of handing it to the CA during the key registration process she attaches it to her key (or a PKI certificate on the key). This POP message is then verified by a multisignature receiver instead of by the CA, for example together with verification

of PKI certificates on that key. We note that in the multisignature schemes we propose POP verification costs are comparable to DSA certificate verification, and this cost can be further reduced by batching. The Key Verification model of operation should also make it easier to adopt multisignature schemes: Since the CA operation does not need to change, a multisignature scheme secure in the KV model can potentially use existing private-public keys and certificates. For example, our CDH-based multisignature scheme can utilize existing DSA or Schnorr signature public keys. We stress that while the KR and KV models differ in how a multisignature scheme operates, any multisignature scheme secure in the KV model trivially implies a scheme secure in the KR model, and any scheme secure in the KR model which has a non-interactive key registration process (e.g. the schemes given by [RY07]) implies a scheme secure in the KV model.

Our Contributions. We propose two multisignature schemes in the KV (or KR) model, denoted MDDH and MCDH. Both schemes take three rounds, have multisignature verification procedures with $O(1)$ exponentiations, do not require bilinear maps, and their security is *tightly* related in ROM to, respectively, the CDH and DDH problems. The POP messages in both schemes take $O(1)$ group elements and their verification takes $O(1)$ exponentiations. Figure 1 summarizes the comparison between ours and previous multisignature schemes. In this table, RY+BLS and RY+Waters refers to the first and the second schemes of [RY07] respectively, BGLS refers to the multisignature scheme implied by the aggregate signature proposed by Boneh et al [BGLS03], MOR+Fischlin refers to the scheme of Micali et al [MOR01] with its initialization phase replaced by key registration using Fischlin's ZKPK's [Fis05], and BN refers to the scheme proposed by Bellare and Neven [BN06].

Compared to the two schemes of [RY07] our schemes do not rely on groups with bilinear maps, but they do so at the cost of relying on the ROM model, which one of the schemes of [RY07] avoids, and by using an interactive multisignature generation. This drawback is significant in many applications but it can be mitigated in applications where the same set of players is expected to repeatedly engage in several instances of the multisignature protocol, as long as the multisignature procedure is fast on-line, *i.e.* if the signed message is an input to the players only in the last round of the interaction, which is the case with our DDH-based scheme. (It is an open problem whether the CDH-based scheme can be made fast on-line without sacrificing other parameters.)

In comparison with the DL-based scheme in the Key Verification model implied by the combined results of [MOR01, Fis05], our POP messages are shorter and faster to verify, which is especially important in the Key Verification model where POP's must be attached to public keys and verified by multisignature recipients. To achieve 2^{80} security the POP size and verification time in the scheme implied by [MOR01, Fis05] would be larger by roughly a factor of ten when compared to our CDH-based and DDH-based schemes. Moreover, the security reduction from the DL problem implied by these two results is inexact, while our schemes have exact reductions from CDH or DDH problems, and in many groups of interest the DL and CDH problems are almost equivalent [MW00].

Finally, compared to the scheme of [BN06] which works in a plain model, our schemes require a Key Verification model. This is a drawback, but as discussed in a subsection above, in many scenarios the Key Verification model of PKI operation

MS Scheme	Assumption on Security(1)	Degradation in Security(2)	Protocol Rounds	Key Setup	Sig.Ver. Time(3)	Signing Time(3)	Signature Length(4)
RY+BLS	GapDH	$1/q_s$	1	POP	O(1)	O(1)	$\|G_1\|$
RY+Waters	GapDH	$1/q_s$	1	POP	O(1)	O(1)	$\|G_1\| + \|G_2\|$
BGLS	GapDH	$1/q_s$	1	Plain	O(n)	O(1)	$\|G_1\|$
MOR+Fischlin	DL(5)	$1/q_s{q_h}^2$	2	POP	O(1)	O(1)	$2\|q\|$
BN	DL(5)	$1/q_h$	3	Plain	O(n)	O(1)	$\|G\| + \|q\|$
MDDH	DDH	exact	3	POP	O(1)	O(1)	$2\|q\|$
MCDH	CDH(6)	exact	3	POP	O(1)	O(n)	$\|G\| + 2\|q\| + 2\kappa$

Fig. 1. (1) All schemes except RY+Waters assume a ROM model; (2) Security degradation is given as a factor f *s.t.* if a multisignature adversary succeeds with probability ϵ then the reduction breaks its underlying security assumption with probability $\Omega(f*\epsilon)$ in comparable time. Here q_s and q_h denote, respectively, the number of adversary's signature and hash queries; (3) Computational costs are the number of modular exponentiations (or bilinear maps for the GapDH-based schemes); (4) Signature length is measured in bits, where κ is the security parameter, $|G|$ is the number of bits required to represent elements in group G, q is the group order, and G_1 and G_2 are two groups of points on an elliptic curve with asymmetrical bilinear maps. For example $\kappa = 80$, $|G| = |q| = |G_1| = 160$ and $|G_2| = 6 * 160$; (5) The reduction for the Fischlin+MOR scheme is our best estimate. The reduction given in [BN06] for the BN scheme has ϵ/q_h degradation, but it seems that one can modify it to provide only $1/q_h$ degradation using the version of the forking lemma originally given by Pointcheval and Stern [PS00]; (6) For the MCDH protocol we only give an exact security reduction from the *expected-time* hardness of the CDH problem.

creates a small overhead in the certificate verification process. On the other hand, our schemes have tight reductions from CDH/DDH problems while the security reduction of [BN06] encounters a security degradation due to the use of the forking lemma, and our schemes make $O(1)$ exponentiation operations during multisignature verification compared to $O(n)$ exponentiation cost in [BN06]. We stress that while it might seem that our schemes require $O(n)$ verification, because we require that each multisignature verifier checks all n POP messages attached to the certificates of the n players involved in the multisignature, the $O(1)$ multisignature verification in the KV model is better than $O(n)$ verification in the plain model for two reasons: (1) Since every entity in PKI normally would keep a hash table of previously verified keys, the initial cost of key verification amortizes over all multisignature verifications which involve this key. (2) If the CA's use DL-based certificates, then the cost of key verification imposed by our schemes is only a constant factor higher than the cost of certificate verification.

In terms of multisignature size, our DDH-based scheme is the same as that of [BN06], and the multisignature in our CDH-based scheme is about 2 times larger than the multisignature of [BN06], when implemented over elliptic curves. Note, however, that if one takes the exact security results into account, the two schemes achieve the same level of provable security when the scheme of [BN06] is implemented over twice larger groups, assuming the near equivalence of the DL and CDH problems. Unlike all other multisignature schemes discussed, our CDH based scheme requires $O(n)$ signing time per party due to verification of NIZKs generated by each player. This may or may not be a drawback in different applications, but it seems that the communication costs of multisignature generation would often trump this computational cost. We point out that

our CDH based scheme has a tight reduction only from expected-time hardness of the CDH problem. However, in generic groups the expected-time hardness of the CDH is essentially the same as the fixed-time hardness of this problem, i.e. for every algorithm which solves the CDH problem in a generic group of size q with probability ϵ and expected-time T, it holds that $T/\epsilon \leq \sqrt{q}$.

Organization. After preliminaries in Section 2 we define secure multisignatures in the Key Verification model in Section 3, and present our DDH-based and CDH-based multisignature schemes respectively in Sections 4 and 5.

2 Preliminaries: Notation and Assumptions

Let G be a multiplicative group of a prime order q, and let g be its generator. All arithmetic operations are either modulo q, when involving numbers chosen in $\mathbb{Z}_q$, or they are operations in G, when involving group elements. We use notation $x \xleftarrow{r} S$ to denote a probabilistic process which assigns to variable x a uniformly chosen value in set S. We write $a|b$ to denote the concatenation of bit strings a and b.

The computational Diffie Hellman (CDH) problem in group G is a problem of computing g^{xy}, given the tuple (g, g^x, g^y) for random x, y in $\mathbb{Z}_q$, while the decisional Diffie Hellman (DDH) problem in G is the problem of distinguishing between tuples of the form (g, g^x, g^y, g^{xy}) for random x, y in $\mathbb{Z}_q$, and (g, g^x, g^y, g^z) for random x, y, z in $\mathbb{Z}_q$.

Definition 1. *The CDH problem is (t, ϵ)-hard in G if for any algorithm B running in time t, we have $\mathbf{Adv}_G^{\mathsf{CDH}}(\mathcal{B}) \leq \epsilon$ where:*

$$\mathbf{Adv}_G^{\mathsf{CDH}}(\mathcal{B}) = \Pr_{x,y \xleftarrow{r} \mathbb{Z}_q}[\mathcal{B}(g, g^x, g^y) = g^{xy}]$$

Definition 2. *The DDH problem is (t, ϵ)-hard in G if for any algorithm B running in time t, we have $\mathbf{Adv}_G^{\mathsf{DDH}}(\mathcal{B}) \leq \epsilon$ where:*

$$\mathbf{Adv}_G^{\mathsf{DDH}}(\mathcal{B}) = |\Pr_{x,y \xleftarrow{r} \mathbb{Z}_q}[\mathcal{B}(g, g^x, g^y, g^{xy}) = 1] - \Pr_{x,y,z \xleftarrow{r} \mathbb{Z}_q}[\mathcal{B}(g, g^x, g^y, g^z) = 1]|$$

Hardness of DL problem in G is implied by the hardness of either the DDH problem or the CDH problem in G, but the converse is not known to be true. However, all these problems have the same hardness as the DL problem in *generic groups* [Sho00]. Moreover, by the results of Maurer and Wolf [MW99], the CDH problem is very closely related to the DL problem in a large class of groups that are commonly used in cryptography. Also, see [Bon98] for various groups where DDH assumption might hold.

3 Multisignature Schemes

We define a Multisignature Scheme (MS) in the Key Verification (KV) model as a tuple $(\mathsf{Setup}, \mathsf{KGen}, \mathsf{KVrfy}, \mathsf{MSign}, \mathsf{Vrfy})$ where Setup, KGen, KVrfy and Vrfy are efficient probabilistic algorithms and MSign is an interactive protocol *s.t.*

- $\mathsf{par} \leftarrow \mathsf{Setup}(1^\kappa)$, on input the security parameter κ generates the public parameters par.
- $(sk, pk, \pi) \leftarrow \mathsf{KGen}(\mathsf{par})$, executed by each user on input par, generates this user's secret key sk, the corresponding public key pk, and a proof of validity of this public key, denoted π.
- $\{0,1\} \leftarrow \mathsf{KVrfy}(\mathsf{par}, pk, \pi)$ verifies whether pk is a valid key, given the proof π.
- MSign is a multisignature protocol executed by a group of players who intend to sign the same message m. Each player P_i executes MSign on public inputs par, message m and private input sk_i, its secret key, and outputs a multisignature σ.
- $\{0,1\} \leftarrow \mathsf{Vrfy}(\mathsf{par}, m, \mathsf{PKSet}, \sigma)$ verifies whether σ is a valid multisignature on message m on behalf of the set of players whose public keys are in set PKSet.

The above set of procedures must satisfy the following *correctness* property. Let par be any output of $\mathsf{Setup}(1^\kappa)$. First, any (sk, pk, π) output by $\mathsf{KGen}(\mathsf{par})$ satisfies $\mathsf{KVrfy}(\mathsf{par}, pk, \pi) = 1$. Second, for any $n \leq n_{max}$, any message m, and any (sk_i, pk_i, π_i) tuples, $i \in \{1, ..., n\}$, generated by $\mathsf{KGen}(\mathsf{par})$, if one executes n instances of protocol MSign, where the i-th instance executes on inputs (par, m, sk_i), and if all the messages between these instances are correctly delivered, then each instance outputs the same string σ *s.t.* $\mathsf{Vrfy}(\mathsf{par}, m, \{pk_1, pk_2, ...pk_n\}, \sigma) = 1$.

Multisignature security in Key Verification model. As in the previous works on multisignatures, *e.g.* [MOR01, BN06, RY07], we define multisignature security as universal unforgeability under a chosen message attack against a single honest player. Namely, we define the *adversarial advantage* of an adversary $\mathcal{A}$ against the multisignature scheme $\mathsf{MS} = (\mathsf{Setup}, \mathsf{KGen}, \mathsf{KVrfy}, \mathsf{MSign}, \mathsf{Vrfy})$, *i.e.* $\mathbf{Adv}^{\mathsf{uu-cma}}_{\mathsf{MS}}(\mathcal{A})$, as a probability that experiment $\mathbf{Exp}^{\mathsf{uu-cma}}_{\mathsf{MS}}(\mathcal{A})$ described in Figure 2 outputs 1,where the probability goes over the random coins of the adversary $\mathcal{A}$ and all the randomness used in the experiment. We call a multisignature scheme $(t, \epsilon, n_{max}, q_s)$-secure if $\mathbf{Adv}^{\mathsf{uu-cma}}_{\mathsf{MS}}(\mathcal{A}) \leq \epsilon$ for every adversary $\mathcal{A}$ that runs in time at most t, makes at most q_s signature queries, and where the size of the group of players S on behalf of which the adversary forges is bounded as $|S| \leq n_{max}$. In the random oracle model we consider also a notion of $(t, \epsilon, n_{max}, q_s, q_h)$-secure multisignature scheme, where adversary $\mathcal{A}$ is additionally restricted to at most q_h hash queries and the probability in the experiment $\mathbf{Exp}^{\mathsf{uu-cma}}_{\mathsf{MS}}(\mathcal{A})$ is taken also over the random choice of all hash functions.

Experiment $\mathbf{Exp}^{\mathsf{uu-cma}}_{\mathsf{MS}}(\mathcal{A})$
$\mathsf{par} \leftarrow \mathsf{Setup}(1^\kappa)$; $(sk^*, pk^*, \pi^*) \leftarrow \mathsf{KGen}(\mathsf{par})$; $\mathsf{List} \leftarrow \emptyset$;
Run $\mathcal{A}(\mathsf{par}, pk^*, \pi^*)$, and for every signature query m made by $\mathcal{A}$ do the following:
$\mathsf{List} \leftarrow \mathsf{List} \cup \{m\}$; Execute protocol MSign on behalf of an honest player on inputs (par, m, sk^*), forwarding messages to and from $\mathcal{A}$.
When $\mathcal{A}$ halts, parse its output as $(m, \sigma, \{(pk_i, \pi_i)\}_{i \in S})$ where S is some set of indexes.
If $(m \notin \mathsf{List})$ and $(pk_1 = pk^*)$ and $(\mathsf{KVrfy}(\mathsf{par}, pk_i, \pi_i) = 1$ for all $i \in \mathcal{S})$ and finally $(\mathsf{Vrfy}(\mathsf{par}, m, \{pk_i\}_{i \in S}, \sigma)) = 1$ then return 1; Otherwise return 0.

Fig. 2. Chosen Message Attack against Multisignature Scheme

Remarks on MS syntax and definition of security: (1) In the security experiment $\mathbf{Exp}^{\mathsf{uu-cma}}_{\mathsf{MS}}$ above we take the simplifying assumption that the Setup procedure is executed by an honest party. However, the public parameters in the two multisignature schemes in this paper are needed to define groups of prime order where the CDH and DDH assumptions hold, and such parameters can be chosen by a potentially dishonest party and then verified by every player. **(2)** The syntax of a multisignature scheme in the KV model is a simplification of the syntax used by [RY07], which modeled potentially interactive key registration processes. Here we allow only *non-interactive* proofs, but such proofs make multisignature schemes more flexible because they can be verified either by the CA's during the key registration process, as in [RY07], or by multisignature verifiers, e.g. together with verification of PKI certificates for a given key. **(3)** Note that a multisignature in the KV model generalizes multisignatures in the plain-model if one sets the proofs of public key validity to empty strings and sets the output of KVrfy on any inputs to 1. **(4)** However, in contrast to the definition of multisignatures in the plain model proposed by [MOR01] and [BN06], we do not include the set of participants' identities and/or their public keys as input in the multisignature protocol. The participating players must be aware of one another in the protocol execution, but this information is needed only to ensure proper communication, and does not need to be part of the inputs to the multisignature protocol. Removing this input from the multisignature protocol gives more flexibility to applications of multisignatures, because in some applications signers might care only about the message they are is signing and not about the identities of other signers. This is the case for example in aggregation of acknowledgments of broadcast reception. In such applications multisignature schemes analyzed in the model of [MOR01, BN06] would have to be preceded by an additional communication round for participants to broadcast their identities and/or public keys. On the other hand, multisignature schemes which conform to our simplified model imply schemes in the model of [MOR01, BN06] if the MSign protocol is executed on the message appended with the list of identities and/or public keys of the participating players. **(5)** The notion of multisignature security in [MOR01, BN06] treats a multisignature effectively as a signature on a *pair* (m, PKSet), and their notion of forgery is consequently broader than ours since it includes a case where an attacker forges a multisignature on a message that was previously signed by the honest player, but it was signed together with a different set of public keys. In our model such adversary would not be considered a successful forger since in our model an honest player is not required to be aware of the other participants in a multisignature protocol. However, a scheme secure according to our notion implies a scheme secure in the model of [MOR01, BN06] if players execute the MSign protocol on the concatenation of message m and the set of public keys PKSet of the participating players, as in item (4) above.

4 Three-Round DDH-Based Multisignature Scheme

We describe a multisignature scheme denoted MDDH, presented in Figure 3, with a tight security reduction from the DDH problem. The MDDH scheme is a multisignature version of the signature scheme of Katz and Wang [KW03], which also has an exact security reduction from the DDH problem. The MDDH scheme takes three rounds and

1. $\mathsf{Setup}(1^\kappa)$: Let G be a multiplicative group of prime order q and let g be a generator of G. Consider the following hash functions: $\mathcal{G}_1 : G^6 \rightarrow \mathbb{Z}_q$, $\mathcal{H}_1 : G \rightarrow \{0,1\}^{2\kappa}$ and $\mathcal{H}_2 : G^6 \times \{0,1\}^* \rightarrow \mathbb{Z}_q$. Pick $g, h \xleftarrow{r} G$. Set $\mathsf{par} \leftarrow (g, h, q)$.
2. $\mathsf{KGen}(\mathsf{par})$: Player P_i picks his (sk_i, pk_i, π_i) tuple as follows:
   Pick $x_i \xleftarrow{r} \mathbb{Z}_q$, compute $y_i \leftarrow g^{x_i}$, $z_i \leftarrow h^{x_i}$ and set $pk_i \leftarrow (y_i, z_i)$, $sk_i \leftarrow x_i$;
   Construct a "proof of possession" of x_i, *i.e.* a NIZK proof of $\mathsf{DL}_g(y_i) = \mathsf{DL}_h(z_i)$:
   Pick $k \xleftarrow{r} \mathbb{Z}_q$ and compute $u \leftarrow g^k$, $v \leftarrow h^k$;
   Set $e \leftarrow \mathcal{G}_1(g, h, y_i, z_i, u, v)$, compute $s \leftarrow k + ex_i$ and set $\pi_i \leftarrow (s, e)$.
3. $\mathsf{KVrfy}(\mathsf{par}, pk, \pi)$: Let pk=(y, z), π=(s, e). Accept if $e = \mathcal{G}_1(g, h, y, z, g^s y^{-e}, h^s z^{-e})$
4. Protocol MSign: Let $\mathcal{S}$ where $\mathcal{S} \leq n_{max}$ be the set of players that participate in the protocol. Each player can determine the set $\mathcal{S}$ after the first protocol step. Player P_i on inputs (par, m, sk_i), performs the following steps:
   4.1 Pick $k_i \xleftarrow{r} \mathbb{Z}_q$, compute $A_i \leftarrow g^{k_i}$, $B_i \leftarrow h^{k_i}$, $C_{A_i} \leftarrow \mathcal{H}_1(A_i)$, $C_{B_i} \leftarrow \mathcal{H}_1(B_i)$;
   Broadcast $(y_i, z_i, C_{A_i}, C_{B_i})$.
   4.2 Receive (C_{A_j}, C_{B_j}) for all $P_j \in \mathcal{S}/\{P_i\}$ and broadcast (A_i, B_i).
   4.3 Receive (A_j, B_j, y_j, z_j) for all $P_j \in \mathcal{S}/\{P_i\}$;
   Abort if $C_{A_j} \neq \mathcal{H}_1(A_j)$ or $C_{B_j} \neq \mathcal{H}_1(B_j)$ for any $P_j \in \mathcal{S}/\{P_i\}$;
   Compute $y \leftarrow \prod_{P_j \in \mathcal{S}} y_j$, $z \leftarrow \prod_{P_j \in \mathcal{S}} z_j$, $A \leftarrow \prod_{P_j \in \mathcal{S}} A_j$, $B \leftarrow \prod_{P_j \in \mathcal{S}} B_j$;
   Set $e \leftarrow \mathcal{H}_2(g, h, y, z, A, B, m)$, compute $s_i \leftarrow ex_i + k_i$ and broadcast s_i.
   4.4 Output multisignature $\sigma = (e, s)$, where $s = \sum_{P_j \in \mathcal{S}} s_j$.
5. $\mathsf{Vrfy}(\mathsf{par}, m, \{pk_1, pk_2, ..., pk_n\}, \sigma)$:
   Parse σ as (e, s) and each pk_i as (y_i, z_i). Compute $y \leftarrow \prod_{i=1}^n y_i$ and $z \leftarrow \prod_{i=1}^n z_i$;
   If $e = \mathcal{H}_2(g, h, y, z, g^s y^{-e}, h^s z^{-e}, m)$ then accept otherwise reject.

Fig. 3. MDDH multisignature scheme

has fast signing and verification procedures. It requires only two group exponentiations per party for signing and two double-exponentiations for verification. The length of the MDDH signature is $2|q|$, which can be 320 bits, only twice the size of shortest multisignature schemes [RY07, BGLS03], which, however, are based on a potentially stronger assumption of GapDH on a group with a bilinear map.

Theorem 1. *If DDH problem is (t', ϵ')-hard in group G, then multisignature* MDDH *in Figure 3 is $(t, \epsilon, n_{max}, q_s, q_h)$-secure in ROM where*

$$\epsilon \leq \epsilon' + \frac{{q_h}^2 + 2q_s(q_h + n_{max})}{2^{2\kappa}} + \frac{q_s q_h}{q - q_h} + \frac{q_h}{q}$$
$$t \geq t' - 2.4(q_s + n_{max})t_e - 4q_s n_{max} t_m$$

and t_m and t_e are the times of one multiplication and one q-bit exponentiation in G.

Proof sketch. Due to space constraints we relegate the formal proof of this theorem to the full version of this paper [BJ08], but we give a rough sketch of the proof here. Given an adversary $\mathcal{A}$ against the MDDH scheme we construct an algorithm $\mathcal{B}$ that solves the DDH problem in group G as follows: The reduction embeds its DDH challenge (g, h, y_1, z_1) into the public key of the sole honest player P_1 by setting $\mathsf{par} = (g, h)$ and $pk_1 = (y_1, z_1)$. Note that the multisignature protocol performed by each player

is a version of a standard HVZK proof system for DL-equality, with the first message A_i, B_i in this proof system preceded by a round of ROM-based commitments C_{A_i}, C_{B_i}. As observed in [BN06], this round of commitment enables straight-line simulation of this HVZK proof system: Namely, $\mathcal{B}$ picks both the challenge e and P_1's response s_1 uniformly at random in $\mathbb{Z}_q$, computes $A_1 = g^{s_1} y_1^{-e}$ and $B_1 = h^{s_1} z_1^{-e}$, *pretends* to commit to these values by sending random C_{A_1}, C_{B_1} values, and thanks to the fact that the adversary commits to his contributions A_i and B_i for $P_i \in \mathcal{S}/\{P_1\}$, the reduction $\mathcal{B}$ can compute the values A and B before she publishes his own contribution A_1, B_1 in step 4.2. In this way $\mathcal{B}$ can embed the challenge e in the output to the appropriate query (g, h, y, z, A, B, m) made by $\mathcal{A}$ to $\mathcal{H}_2$. This reduction fails only if (1) $\mathcal{A}$ manages to change one of his committed values; (2) $\mathcal{A}$ manages to decommit any of his commitments C to some X s.t. $C = \mathcal{H}_1(X)$ without querying $\mathcal{H}_1$ on X; (3) $\mathcal{A}$ makes query (g, h, y, z, A, B, m) to $\mathcal{H}_2$ before the reduction embeds challenge e in the answer. However, all these cases happen with at most negligible probability in ROM. Finally, the special soundness property of the above proof system ensures that if both the multisignature verification and all the key verification procedures pass then both (g, h, y, z) where $y = \prod_{i \in \mathcal{S}}(y_i)$ and $z = \prod_{i \in \mathcal{S}}(x_i)$ and (g, h, y_i, z_i) for each P_i are DH tuples, and hence so must be the input tuple (g, h, y_1, z_1). We leave the details of this proof to the full version [BJ08].

5 Three-Round CDH-Based Multisignature Scheme

We describe a multisignature scheme, denoted MCDH and presented in Figure 4, whose security is tightly related to the *expected-time* hardness of the CDH problem in the Key Verification model. The MCDH scheme is a multisignature version of the CDH-based signature of [KW03] and [GJ03]. It takes three rounds, the multisignature is $(|G| + 2|q| + 2\kappa)$-bit long, and its verification procedure requires only two exponentiations. We note, however, that the low round complexity and a tight reduction from the (expected time) CDH problem comes at the following non-standard cost: Each player in the multisignature generation procedure must verify ZK proofs issued by the other players, and thus the computational cost of the signing algorithm is $O(n)$ exponentiations where n is the size of the signing group. This, however, will not be an important drawback as long as the number of signers is modest or if the communication costs in n-sized group of signers dominate the $O(n)$-exponentiations cost for each signer. As we discuss at the end of this section, this cost can be avoided if one tolerates an increase in the number of protocol rounds.

Theorem 2. *If there is no adversary that can solve the CDH problem in group G in expected time t' with probability ϵ', then the multisignature scheme* MCDH*, described in figure 4 is $(t, \epsilon, n_{max}, q_s, q_h)$-secure in random oracle model where*

$$\epsilon \leq 2\epsilon' + \frac{2q_h^2 + 4q_s q_h}{2^{2\kappa}} + \frac{(2q_s + 3)q_h}{q} + \frac{q_s q_h}{q - q_h}$$

$$t \geq \frac{1}{2}\left(t' - (q_h - 6(n_{max} + 1)(q_s + 1))t_e - 4q_s n_{max} t_m\right)$$

where t_m and t_e are the times of one multiplication and one q-bit exponentiation in G.

1. $\mathsf{Setup}(1^\kappa)$: Let G be a multiplicative group of prime order q and let g be a generator of G. Consider the following hash functions: $\mathcal{G}_1 : G \to G$, $\mathcal{G}_2 : G^6 \to \mathbb{Z}_q$, $\mathcal{H}_1 : G \to \{0,1\}^{2\kappa}$, $\mathcal{H}_2 : \{0,1\}^* \to \{0,1\}^{2\kappa}$, $\mathcal{H}_3 : \{0,1\}^* \times \{0,1\}^{2\kappa} \to G$, $\mathcal{H}_4 : G^8 \to \mathbb{Z}_q^2$ and $\mathcal{H}_5 : G^6 \to \mathbb{Z}_q$. Set $\mathsf{par} \leftarrow (g, q)$;
2. $\mathsf{KGen}(\mathsf{par})$: Player P_i picks an (sk_i, pk_i, π_i) tuple as follows:
   Pick $x_i \xleftarrow{r} \mathbb{Z}_q$, compute $y_i \leftarrow g^{x_i}$ and set $pk_i \leftarrow y_i$ and $sk_i \leftarrow x_i$;
   Construct π_i as "proof of possession" of x_i:
   Set $h \leftarrow \mathcal{G}_1(y_i)$, $z \leftarrow h^{x_i}$ and construct a NIZK proof of $\mathsf{DL}_g(y_i) = \mathsf{DL}_h(z)$:
   Pick $k \xleftarrow{r} \mathbb{Z}_q$ and compute $u \leftarrow g^k$, $v \leftarrow h^k$;
   Set $e \leftarrow \mathcal{G}_2(g, h, y_i, z, u, v)$, compute $s \leftarrow k + ex_i$ and set $\pi_i \leftarrow (z, s, e)$;
3. $\mathsf{KVrfy}(\mathsf{par}, pk, \pi)$: Let $pk = y$, $\pi = (z, s, e)$ and $h \leftarrow \mathcal{G}_1(y)$.
   Accept if $e = \mathcal{G}_2(g, h, y, z, g^s y^{-e}, h^s z^{-e})$.
4. Protocol MSign: Let $\mathcal{S}$ where $\mathcal{S} \leq n_{max}$ be the set of players that participate in the protocol. Each player can determine the set $\mathcal{S}$ after the first protocol step. Player P_i on inputs (par, m, sk_i), performs the following steps:
   4.1 Pick $k_i \xleftarrow{r} \mathbb{Z}_q$, compute $A_i \leftarrow g^{k_i}$ and set $C_{A_i} \leftarrow \mathcal{H}_1(A_i)$. If bit $b_i^{(m)}$ is not set, pick $b_i^{(m)} \xleftarrow{r} \{0,1\}$. Broadcast $(b_i^{(m)}, y_i, C_{A_i})$.
   4.2 Receive $(b_j^{(m)}, y_j, C_{A_j})$ for all $P_j \in \mathcal{S}/\{P_i\}$; Abort if for any of them, $C_{A_j} = C_{A_i}$.
   Set $p \leftarrow \mathcal{H}_2(b_1^{(m)} | b_2^{(m)} | ... | b_{|\mathcal{S}|}^{(m)})$, where the global order among players can be determined *e.g.* by hashing the message broadcasted by players in the previous round. (If two players broadcast the same message, the order between them can be arbitrary.)
   Set $h \leftarrow \mathcal{H}_3(m, p)$ and Compute $z_i \leftarrow h^{x_i}$ and $B_i \leftarrow h^{k_i}$;
   Construct NIZK proof π_i that $\mathsf{DL}_g(y_i) = \mathsf{DL}_h(z_i)$ and $\mathsf{DL}_g(A_i) = \mathsf{DL}_h(B_i)$:
   Pick $r \xleftarrow{r} \mathbb{Z}_q$, set $u \leftarrow g^r$, $v \leftarrow h^r$ and $(e_i, f_i) \leftarrow \mathcal{H}_4(g, h, y_i, z_i, A_i, B_i, u, v)$;
   Compute $t_i \leftarrow r + e_i x_i + f_i k_i$ and set $\pi_i \leftarrow (e_i, f_i, t_i)$;
   Broadcast (A_i, B_i, z_i, π_i).
   4.3 Receive (A_j, B_j, z_j, π_j) for all $P_j \in \mathcal{S}/\{P_i\}$; Let $\pi_j = (e_j, f_j, t_j)$;
   Abort if $(e_j, f_j) \neq \mathcal{H}_4(g, h, y_j, z_j, A_j, B_j, g^{t_j} y_j^{-e_j} A_j^{-f_j}, h^{t_j} z_j^{-e_j} B_j^{-f_j})$ or $C_{A_j} \neq \mathcal{H}_1(A_j)$ for any $P_j \in \mathcal{S}/\{P_i\}$.
   Compute $y \leftarrow \prod_{P_j \in \mathcal{S}} y_j$, $z \leftarrow \prod_{P_j \in \mathcal{S}} z_j$, $A \leftarrow \prod_{P_j \in \mathcal{S}} A_j$ and $B \leftarrow \prod_{P_j \in \mathcal{S}} B_j$;
   Set $e \leftarrow \mathcal{H}_5(g, h, y, z, A, B)$, $s_i \leftarrow k_i + ex_i$ and broadcast s_i.
   4.4 Output $\sigma = (z, e, s, p)$, where $s = \sum_{P_j \in \mathcal{S}} s_j$.
5. $\mathsf{Vrfy}(\mathsf{par}, m, \{pk_1, pk_2, ..., pk_n\}, \sigma)$:
   Parse σ as (z, e, s, p) and each pk_i as y_i. Set $y \leftarrow \prod_{i=1}^n y_i$ and $h \leftarrow \mathcal{H}_3(m, p)$;
   If $e = \mathcal{H}_5(g, h, y, z, g^s y^{-e}, h^s z^{-e})$ then accept otherwise reject.

Fig. 4. MCDH multisignature scheme

Proof. Let $\mathcal{A}$ be an adversary that attacks the multisignature scheme MCDH, depicted in figure 4, in time t and with success probability ϵ and makes q_s signing queries and at most q_h hash queries and produces a forgery on behalf of $n \leq n_{max}$ players. We construct an algorithm $\mathcal{B}$, that given oracle access to $\mathcal{A}$, solves CDH problem in group G, *i.e.* given $(g, y_1, \hat{h}) \in G^3$ where $y_1 = g^{x_1}$ it outputs $\hat{h}^{x_1}$ in expected time t' and with success probability ϵ'. Assume $\mathcal{A}$ makes $q_{\mathcal{G}_1}$, $q_{\mathcal{G}_2}$, $q_{\mathcal{H}_1}$, $q_{\mathcal{H}_2}$, $q_{\mathcal{H}_3}$, $q_{\mathcal{H}_4}$ and $q_{\mathcal{H}_5}$ queries to $\mathcal{G}_1$, $\mathcal{G}_2$, $\mathcal{H}_1$, $\mathcal{H}_2$, $\mathcal{H}_3$, $\mathcal{H}_4$ and $\mathcal{H}_5$ respectively and $q_{\mathcal{G}_1} + q_{\mathcal{G}_2} + q_{\mathcal{H}_1} + q_{\mathcal{H}_2} + q_{\mathcal{H}_3} + q_{\mathcal{H}_4} + q_{\mathcal{H}_5} \leq q_h$, the total number of hash queries. In what follows we show how the CDH-attacker $\mathcal{B}$ proceeds given a CDH challenge $(g, y_1, \hat{h})$.

Initialization: The algorithm $\mathcal{B}$ sets up tables G_1, G_2, H_1, H_2, H_3, H_4 and H_5 to simulate the hash functions $\mathcal{G}_1$, $\mathcal{G}_2$, $\mathcal{H}_1$, $\mathcal{H}_2$, $\mathcal{H}_3$, $\mathcal{H}_4$ and $\mathcal{H}_5$ respectively which are filled out throughout the simulation. It also uses table B to store the bits $b^{(m)}$ assigned to each message m. The algorithm $\mathcal{B}$ then sets the public parameters and the honest player's public key as $\mathsf{par} = g$ and $pk_1 = y_1$ respectively. Algorithm $\mathcal{B}$ picks $\beta_{y_1} \xleftarrow{r} \mathbb{Z}_q$ and assigns $\mathsf{G}_1[y_1] = g^{\beta_{y_1}}$. It then computes $z_1 \leftarrow (y_1)^{\beta_{y_1}}$ and produces a simulated NIZK proof of DL-equality between $\mathsf{DL}_g(y_1)$ and $\mathsf{DL}_h(z_1)$ where $h = \mathsf{G}_1[y_1]$. To simulate this proof, $\mathcal{B}$ picks $e, s \xleftarrow{r} \mathbb{Z}_q$ and assigns $\mathsf{G}_2[(g, h, y_1, z_1, g^s y_1^{-e}, h^s z_1^{-e})] = e$. Finally, $\mathcal{B}$ executes $\mathcal{A}$ on input $(\mathsf{par}, pk_1, \pi_1)$ where $\pi_1 = (s, e)$. Note that $\mathsf{G}_1[y_1]$ and $\mathsf{G}_2[(g, h, y_1, z_1, g^s y_1^{-e}, h^s z_1^{-e})]$ are set before the execution of $\mathcal{A}$ starts. Note also that indeed $\mathsf{DL}_g(y_1) = \mathsf{DL}_h(z_1)$ since $z_1 = (y_1)^{\beta_{y_1}} = g^{x_1 \beta_{y_1}} = h^{x_1}$.

Answering hash queries: If a query has been made before, then $\mathcal{B}$ returns the appropriate entry from the appropriate table *e.g.* if $\mathcal{A}$ queries $\mathcal{G}_1$ on y, $\mathcal{B}$ responds with $\mathsf{G}_1[y]$. If $\mathcal{A}$ makes a new query to $\mathcal{G}_2$, $\mathcal{H}_1$, $\mathcal{H}_4$ and $\mathcal{H}_5$, $\mathcal{B}$ answers with an element chosen uniformly at random from the appropriate domain. If $\mathcal{A}$ makes a new query y to $\mathcal{G}_1$, $\mathcal{B}$ answers with $\hat{h}^{\beta_y}$ where $\beta_y \xleftarrow{r} \mathbb{Z}_q$. If $\mathcal{A}$ makes a new query $\boldsymbol{b} = b_1|...|b_{|\mathcal{S}|}$ to $\mathcal{H}_2$, $\mathcal{B}$ answers with an element chosen uniformly at random form $\{0,1\}^{2\kappa}$ except when the following failure cases happen: (a) If a collision happens in $\mathcal{H}_2$ then the simulator sets a flag $\mathsf{bad}_1 \leftarrow \mathsf{true}$, stops and returns "fail". (b) If $\mathcal{A}$ queries $\mathcal{H}_2$ for the first time on $\boldsymbol{b} = b_1|...|b_{|\mathcal{S}|}$ and $\mathcal{H}_2(\boldsymbol{b}) = p$ for some previous query (m, p) to $\mathcal{H}_3$ then the simulator sets a flag $\mathsf{bad}_2 \leftarrow \mathsf{true}$, stops and returns "fail". If $\mathcal{A}$ makes a new query (m, p) to $\mathcal{H}_3$, the simulator looks up the table H_2: (a) If there exists no entry in H_2 corresponding to p then the simulator picks $\alpha_{(m,p)} \xleftarrow{r} \mathbb{Z}_q$ and answers the query as $\hat{h}^{\alpha_{(m,p)}}$; (b) If the simulator finds an entry $\boldsymbol{b} = b_1|...|b_{|\mathcal{S}|}$ corresponding to p in H_2 then it assigns a bit ${b_1}^{(m)}$ to message m if it has not been yet assigned, picks $\alpha_{(m,p)} \xleftarrow{r} \mathbb{Z}_q$ and checks whether the first element of the vector $\boldsymbol{b}$ is equal to ${b_1}^{(m)}$: If so, the simulator responds to the query as $g^{\alpha_{(m,p)}}$ and otherwise it responds to the query as $\hat{h}^{\alpha_{(m,p)}}$.

Answering signature queries: To answer each signature query, simulator runs the adversary twice after step 4.1. In the first execution, $\mathcal{B}$ runs the adversary to the point that it can learn A_j, B_j and z_j for all $P_j \in \mathcal{S}$, but it does not complete this execution since it will not know how to correctly create the response s_1 in the zero-knowledge proof on behalf of player P_1. The simulator then rewinds the adversary and uses the values it has learned in the first execution to simulate the proof-of-knowledge protocol on behalf of player P_1. If the adversary uses the same values $\{A_j, B_j, z_j\}_{P_j \in \mathcal{S}}$ in both executions then the simulator knows the query $\mathcal{H}_5\,(g, h, y, z, A, B)$ into which it should embed its chosen challenge e. The only way this simulation can fail is if the adversary changes his values z_j, A_j and B_j for $P_j \in \mathcal{S}/\{P_1\}$ in the second execution. However due to the soundness property of NIZK proof of DL-equality and the collision resistance property of $\mathcal{H}_1$, this can happen with only a negligible probability. This is because the adversary has revealed y_j's and committed to A_j's in the first round. Moreover, the adversary gives a NIZK proof that $\mathsf{DL}_g(y_j) = \mathsf{DL}_h(z_j)$ and $\mathsf{DL}_g(A_j) = \mathsf{DL}_h(B_j)$ for all $P_j \in \mathcal{S}/\{P_1\}$. The details of the signature query simulation on input m are given bellow. We point out that if the adversary aborts and/or sends values which do not pass

verification procedures in any point in the simulation below then the simulator stops this instance of the multisignature generation protocol, just like an honest player P_1.

Step 1. If bit $b_1^{(m)}$ has not been chosen for message m, pick $b_1^{(m)} \stackrel{r}{\leftarrow} \{0,1\}$; Otherwise use the previously chosen value. Pick $C_{A_1} \stackrel{r}{\leftarrow} \{0,1\}^{2\kappa}$. Send $(b_1^{(m)}, y_1, C_{A_1})$ to $\mathcal{A}$.

Step 2,3. Upon receiving $(b_j^{(m)}, y_j, C_{A_j})$ for all $P_j \in \mathcal{S}/\{P_1\}$, verify whether for all $P_j \in \mathcal{S}/\{P_1\}$, $C_{A_j} \neq C_{A_1}$; Abort if verification fails. Set $p \leftarrow \mathcal{H}_2(b_1^{(m)}|b_2^{(m)}|...|b_{|\mathcal{S}|}^{(m)})$ and $h \leftarrow \mathcal{H}_3(m,p)$. Retrieve $\alpha_{(m,p)}$ assigned to (m,p) in the simulation of this query to $\mathcal{H}_3$ and compute $z_1 \leftarrow y_1^{\alpha_{(m,p)}}$. Note that our simulation procedure for $\mathcal{H}_3$ ensures that $h = \mathcal{H}_3(m,p) = g^{\alpha_{(m,p)}}$ and thus $z_1 = y_1^{\alpha_{(m,p)}} = g^{x_1\alpha_{(m,p)}} = h^{x_1}$.

Run SIM_R as a subroutine and let $\{(A_j^{(1)}, B_j^{(1)}, z_j^{(1)})\}_{P_j \in \mathcal{S}/\{P_1\}}$ be the values it returns. If SIM_R did not stop, rewind the adversary to the point where SIM_R is called, and run SIM_L as a subroutine on input $\{(A_j^{(1)}, B_j^{(1)}, z_j^{(1)})\}_{P_j \in \mathcal{S}/\{P_1\}}$.

Step 4. Compute the multisignature from appropriate values gained in SIM_R simulation.

Procedure SIM_R:

Step 2'. Pick $k_1 \stackrel{r}{\leftarrow} \mathbb{Z}_q$ and compute $A_1^{(1)} \leftarrow g^{k_1}$, $B_1^{(1)} \leftarrow h^{k_1}$. If $\mathsf{H}_1[A_1^{(1)}]$ is not set, assign $\mathsf{H}_1[A_1^{(1)}] \leftarrow C_{A_1}$; Otherwise set $\mathsf{bad}_3 \leftarrow \mathsf{true}$, stop and return "fail".

Simulate the NIZK proof that $\mathsf{DL}_g(y_1) = \mathsf{DL}_h(z_1)$ and $\mathsf{DL}_g(A_1^{(1)}) = \mathsf{DL}_h(B_1^{(1)})$:

Pick $e_1, f_1, t_1 \stackrel{r}{\leftarrow} \mathbb{Z}_q^3$, set $u_1 \leftarrow g^{t_1} y_1^{-e_1} (A_1^{(1)})^{-f_1}$, $v_1 \leftarrow h^{t_1} z_1^{-e_1} (B_1^{(1)})^{-f_1}$.

If $\mathsf{H}_4[(g,h,y_1,z_1,A_1^{(1)},B_1^{(1)},u_1,v_1)]$ is not set, set it to (e_1,f_1); Otherwise set $\mathsf{bad}_4 \leftarrow \mathsf{true}$, stop and return "fail".

Send $(A_1^{(1)}, B_1^{(1)}, z_1, (e_1,f_1,t_1))$ to $\mathcal{A}$.

Step 3'. Upon receiving $(A_j, B_j, z_j, (e_j, f_j, t_j))$ for all $P_j \in \mathcal{S}/\{P_1\}$, verify whether for all $P_j \in \mathcal{S}/\{P_1\}$, $(e_j, f_j) = \mathcal{H}_4(g,h,y_j,z_j,A_j,B_j, g^{t_j} y_j^{-e_j} A_j^{-f_j}, h^{t_j} z_j^{-e_j} B_j^{-f_j})$ and $C_{A_j} = \mathcal{H}_1(A_j)$. If the verification does not pass, stop the simulation of this multisignature instance. Otherwise return $\{(A_j, B_j, z_j)\}_{P_j \in \mathcal{S}/\{P_1\}}$.

Procedure $SIM_L(\{(z_j^{(1)}, A_j^{(1)}, B_j^{(1)})\}_{P_j \in \mathcal{S}/\{P_1\}})$:

Step 2. Pick $(s_1, e) \stackrel{r}{\leftarrow} \mathbb{Z}_q^2$ and compute $A_1^{(2)} \leftarrow g^{s_1} y_1^{-e}$, $B_1^{(2)} \leftarrow h^{s_1} z_1^{-e}$. If $\mathsf{H}_1[A_1^{(2)}]$ is not set, assign $\mathsf{H}_1[A_1^{(2)}] \leftarrow C_{A_1}$; Otherwise set $\mathsf{bad}_5 \leftarrow \mathsf{true}$, stop and return "fail".

Simulate the NIZK proof that $\mathsf{DL}_g(y_1) = \mathsf{DL}_h(z_1)$ and $\mathsf{DL}_g(A_1^{(2)}) = \mathsf{DL}_h(B_1^{(2)})$

Pick $e_1, f_1, t_1 \stackrel{r}{\leftarrow} \mathbb{Z}_q^3$, set $u_1 \leftarrow g^{t_1} y_1^{-e_1} (A_1^{(2)})^{-f_1}$, $v_1 \leftarrow h^{t_1} z_1^{-e_1} (B_1^{(2)})^{-f_1}$.

If $\mathsf{H}_4[(g, h, y_1, z_1, A_1^{(2)}, B_1^{(2)}, u_1, v_1)]$ is not set, set it to (e_1, f_1); Otherwise set $\mathsf{bad}_6 \leftarrow \mathsf{true}$, stop and return "fail".

Compute $y \leftarrow \prod_{P_j \in \mathcal{S}} y_j$, $z \leftarrow z_1 \prod_{P_j \in \mathcal{S}/\{P_1\}} z_j^{(1)}$, $A \leftarrow A_1^{(2)} \prod_{P_j \in \mathcal{S}/\{P_1\}} A_j^{(1)}$, $B \leftarrow B_1^{(2)} \prod_{P_j \in \mathcal{S}/\{P_1\}} B_j^{(1)}$. If $\mathsf{H}_5[(g, h, y, z, A, B)]$ is not set, set it to e; Otherwise set $\mathsf{bad}_7 \leftarrow \mathsf{true}$, stop and return "fail".

Send $(A_1^{(2)}, B_1^{(2)}, z_1, (e_1, f_1, t_1))$ to $\mathcal{A}$.

Step 3. Upon receiving $(A_j, B_j, z_j, (e_j, f_j, t_j))$ for all $P_j \in \mathcal{S}/\{P_1\}$, verify whether for all $P_j \in \mathcal{S}/\{P_1\}$,

(a) $A_j = A_j^{(1)}$; If not, set $\mathsf{bad}_8 \leftarrow \mathsf{true}$, stop and return "fail".

(b) $B_j = B_j^{(1)}$ and $z_j = z_j^{(1)}$; If not, set $\mathsf{bad}_9 \leftarrow \mathsf{true}$, stop and return "fail".

(c) $(e_j, f_j) = \mathcal{H}_4(g, h, y_j, z_j, A_j, B_j, g^{t_j} y_j^{-e_j} A_j^{-f_j}, h^{t_j} z_j^{-e_j} B_j^{-f_j})$ and $C_{A_j} = \mathcal{H}_1(A_j)$; If not stop the simulation of this multisignature instance.

If all the verifications pass, send s_1 to $\mathcal{A}$.

Finalization: After receiving a valid forgery $(m, \sigma, \{(pk_i, \pi_i)\}_{P_i \in \mathcal{S}})$ from $\mathcal{A}$, the algorithm $\mathcal{B}$ attempts to output $\hat{z} = \hat{h}^{x_1}$. Let $\sigma = (z, e, s, p)$ and $pk_i = y_i$ and $\pi_i = (z_i, e_i, s_i)$ for all $P_i \in \mathcal{S}$. If in the simulation of $\mathcal{H}_3$ for query (m, p), the simulator finds an entry $\boldsymbol{b} = b_1|...|b_{|\mathcal{S}|}$ corresponding to p in H_2, and moreover the first element of the vector $\boldsymbol{b}$ is equal to $b_1^{(m)}$, then it stops and returns "fail"; otherwise $\mathcal{B}$ retrieves $\alpha_{(m,p)}$ assigned to (m, p) in the simulation of $\mathcal{H}_3$ and β_{y_i} assigned to y_i in the simulation to $\mathcal{G}_1$ for all $P_i \in \mathcal{S}$ and returns $\hat{z}$ where

$$\hat{z} = \begin{cases} \frac{z^{1/\alpha_{(m,p)}}}{\prod_{P_i \in \mathcal{S}/\{1\}} (z_i)^{1/\beta_{y_i}}} & \text{when } \mathcal{S}/\{P_1\} \neq \emptyset \\ z^{1/\alpha_{(m,p)}} & \text{otherwise} \end{cases}$$

Note that if $(g, y, \hat{h}^{\alpha_{(m,p)}}, z)$ where $y = \prod_{P_i \in \mathcal{S}} (y_i)$ and $(g, y_i, \hat{h}^{\beta_{y_i}}, z_i)$ where $P_i \in \mathcal{S}/\{P_1\}$ are all DH tuples, then $(g, y_1, \hat{h}, \hat{z})$ is also a DH tuple. We will argue that if the multisignature verification passes then with a high probability $(g, y, \hat{h}^{\alpha_{(m,p)}}, z)$ is a DH tuple and if the key verification passes for all of the adversary's public keys then with a high probability $(g, y_i, \hat{h}^{\beta_{y_i}}, z_i)$'s are also all DH tuples, in which case $\hat{z}$ is the answer to the reduction's CDH challenge. But first let's look at the probability of failure events. Let E_i for $i = 1..9$, denote the failure event that $\mathsf{bad}_i = \mathsf{true}$. The algorithm $\mathcal{B}$ provides a perfect simulation for adversary $\mathcal{A}$ conditioned on events E_i where $i = 1..9$ not happening. More precisely, if events E_i for $i = 1..9$ do not happen then: Firstly, the view of the adversary interacting with the simulator in SIM_R branch is identical to the view of the adversary in the real execution conditioned on the event that the simulation stops in step (3'), *i.e.* if $\mathcal{A}$'s responses in that step do not pass the verification procedure. Secondly, the view of the adversary interacting with the simulator in SIM_L branch is identical to the view of the adversary in the real execution conditioned on the event that $\mathcal{A}$'s responses in step (3') are correct, *i.e.* that values $\{(z_j^{(1)}, A_j^{(1)}, B_j^{(1)})\}_{P_j \in \mathcal{S}/\{P_1\}}$

which are input to SIM_L satisfy the verification condition. (Note that the SIM_L branch executes only under this condition.) We start by upper-bounding the probabilities of all the "fail" events:

The event E_1 corresponds to a collision in $\mathcal{H}_2$. Thus $\Pr[E_1] \leq (q_{\mathcal{H}_2})^2/2^{2\kappa}$. To upper bound E_2, it is enough to upper bound the event that $\mathcal{H}_2$ is queried on some $\boldsymbol{b} = b_1|...|b_{|\mathcal{S}|}$ where $\mathcal{H}_2(\boldsymbol{b}) = p$ for some previous query (m, p) to $\mathcal{H}_3$. The probability that any query to $\mathcal{H}_2$ hits some p *s.t.* (m, p) is also queried to $\mathcal{H}_3$ is at most $q_{\mathcal{H}_3}/2^{2\kappa}$ and there are at most $q_{\mathcal{H}_2}$ queries to $\mathcal{H}_2$, thus $\Pr[E_2] \leq q_{\mathcal{H}_2} q_{\mathcal{H}_3}/2^{2\kappa}$. The events E_3 and E_5 reflect the possibility that $\mathcal{H}_1$ has been queried on A_1 before it is set by $\mathcal{B}$ in a particular signing query. Since no information about A_1 is revealed before it is set by $\mathcal{B}$, thus both of these events can be upper bounded by $q_s q_{\mathcal{H}_1}/2^{2\kappa}$. Similarly both E_4 and E_6 can be upper bounded by $q_s q_{\mathcal{H}_4}/q$. The event E_7 reflects the possibility that $\mathcal{H}_5$ has been queried on (g, h, y, z, A, B) before it is set by $\mathcal{B}$ in a particular signing query. Here we have two cases: either adversary has queried $\mathcal{H}_1$ on A_1 or B_1 in a particular signing query or he has not. In the first case which happens with probability at most $2q_{\mathcal{H}_1}/2^{2\kappa}$, the adversary knows both A and B and can easily query $\mathcal{H}_5$ on (g, h, y, z, A, B) before it is set by $\mathcal{B}$. In the second case $\mathcal{A}$ still has some information about A and B and can happen to query $\mathcal{H}_5$ on (g, h, y, z, A, B) with probability at most $q_{\mathcal{H}_5}/(q - q_{\mathcal{H}_1})$. Thus, $\Pr[E_7] \leq 2q_s q_{\mathcal{H}_1}/2^{2\kappa} + q_s q_{\mathcal{H}_5}/(q - q_{\mathcal{H}_1})$. The event E_8 corresponds to a collision in $\mathcal{H}_1$. Now since $C_{A_j} \neq C_{A_1}$ for all $P_j \in \mathcal{S}$, and C_{A_1} is the only output of $\mathcal{H}_1$ that is being manipulated by $\mathcal{B}$ in SIM_R and SIM_L, therefore with regards to value C_{A_j}, for any $P_j \in \mathcal{S}/\{P_1\}$, the hash function $\mathcal{H}_1$ remains collision resistant across these SIM_L and SIM_R executions. Thus the value $A_j^{(1)}$ revealed for C_{A_j} in SIM_R and value A_j revealed for C_{A_j} in SIM_L can be different with probability at most $(q_{\mathcal{H}_1})^2/2^{2\kappa}$. Hence $\Pr[E_8] \leq (q_{\mathcal{H}_1})^2/2^{2\kappa}$. The event E_9 reflects the possibility that $\mathcal{A}$ has cheated on at least one of the NIZK proofs in SIM_R or SIM_L branches. However due to special soundness property of this double-DL-equality proof system, for given tuple $(g, h, y_i, z_i, A_i, B_i)$ not satisfying double-DL-equality, for any $u_i, v_i \in G$, there's at most q different $(e, f) \in \mathbb{Z}_q$ pairs that satisfy the verification equation for some t. Therefore the probability of hitting such pair in $q_{\mathcal{H}_4}$ queries is bounded by $q_{\mathcal{H}_4}/q$. Thus $\Pr[E_9] \leq 2q_{\mathcal{H}_4}/q$.

There is also a possibility of failure in reduction after $\mathcal{A}$ outputs a valid forgery. Namely if in the simulation of $\mathcal{H}_3$ for query (m, p) where m is in the forgery, the simulator finds an entry $\boldsymbol{b} = b_1|...|b_{|\mathcal{S}|}$ corresponding to p in H_2, and moreover the first element of the vector $\boldsymbol{b}$ is equal to $b_1^{(m)}$, the query is answered by $g^{\alpha_{(m,p)}}$ and therefore is useless for the reduction. However since with probability $1/2$, the first element of the vector $\boldsymbol{b}$ is not equal to $b_1^{(m)}$, therefore with probability at least $1/2$ the reduction proceeds to output $\hat{z}$ after obtaining a valid forgery from $\mathcal{A}$.

Now since $\mathsf{Vrfy}(g, m, \{pk_i\}_{P_i \in \mathcal{S}}, \sigma) = 1$, thus due to special soundness property of DL-equality proof system, except for a probability of $q_{\mathcal{H}_5}/q$, $(g, y, \hat{h}^{\alpha_{(m,p)}}, z)$ is a DH tuple. Similarly Since $\mathsf{KVrfy}(\mathsf{par}, pk_i, \pi) = 1$ for all $P_j \in \mathcal{S}/\{P_1\}$, except for a probability of $q_{\mathcal{G}_2}/q$, the tuples $(g, y_j, \hat{h}^{\beta_{y_j}}, z_j)$ where $P_j \in \mathcal{S}/\{P_1\}$ are all DH tuples. Therefore, $(g, y_1, \hat{h}, \hat{z})$ is also a DH tuple except for these error probabilities, and thus $\mathcal{B}$ solves the DH problem with advantage $\epsilon' = 1/2(\epsilon - \mathsf{err})$ where

$$\mathsf{err} \leq \frac{q_{\mathcal{H}_2}(q_{\mathcal{H}_2}+q_{\mathcal{H}_3})+4q_s q_{\mathcal{H}_1}+(q_{\mathcal{H}_1})^2}{2^{2\kappa}}+\frac{2(q_s+1)q_{\mathcal{H}_4}+q_{\mathcal{H}_5}+q_{\mathcal{G}_2}}{q}+\frac{q_s q_{\mathcal{H}_5}}{q-q_{\mathcal{H}_1}}$$

The fact that $q_{\mathcal{G}_1}+q_{\mathcal{G}_2}+q_{\mathcal{H}_1}+q_{\mathcal{H}_2}+q_{\mathcal{H}_3}+q_{\mathcal{H}_4}+q_{\mathcal{H}_5} \leq q_h$ yields the desired result.

Calculating the running time of the simulator is little bit complicated due to the tree-like nature of its execution structure. Let's focus on "answering to the signature queries" part since it is the most time consuming part of the execution. Let $r_1|r_2|...|r_i$ be the randomness of the real execution of the protocol in which the randomness of the j^{th} signature query instance, $1 \leq j \leq i$, is r_j. Therefore $r_1|r_2|...|r_i$ determines the path of real execution up to i^{th} signature query. Accordingly let $t_i(r_1|r_2|...|r_i)$ be the running time of the real execution of the protocol in i^{th} round of answering signature queries. By assumption we have $\forall r_1, r_2, ..., r_{q_s} \sum_{i=1}^{q_s} t_i(r_1|r_2|...|r_i) = T$ where T is the total running time of the forger in "answering to the signature queries" part.

Let s_i^L and s_i^R be the running times of the simulator in SIM_L and SIM_R branches of the execution respectively that interact with the adversary in the i^{th} signature query. As we mentioned before the view of the adversary interacting with the simulator in SIM_R branch is similar to the view of the adversary in real execution of the protocol conditioned on the halting of the adversary except possibly with some negligible factor of failure. Similarly except for a negligible probability, the view of the adversary interacting with the simulator in SIM_L branch is similar to the view of the adversary in real execution. This can be stated more formally as follows: for all random coins of the adversary, $(A, SIM_R)_{\$\ SIM_R} = (A, P)_{\$\ P|\mathcal{A}\ \text{aborts}}$ and $(A, SIM_L)_{\$\ SIM_L} = (A, P)_{\$\ P}$. This means that there is a one to one correspondence between the randomness of each signature query in the real execution and the execution of the simulator both in SIM_L and in SIM_R. Therefore we can calculate the running time of the simulator by calculating the following:

$$T' = t_1(r_1^L) + t_1(r_1^R) + ... + t_{q_s}(r_1^L|r_2^L|...|r_{q_s-1}^L|r_{q_s}^L) + t_{q_s}(r_1^L|r_2^L|...|r_{q_s-1}^L|r_{q_s}^R)$$

Note that there exist random coins that lead to non-constant reduction: Consider an extreme case in which $\forall_{1\leq i\leq q_s-1}, t_i(r_1^L|r_2^L|...|r_i^L) = 0$ and $t_i(r_1^L|r_2^L|...|r_i^R) = T$ and $t_{q_s}(r_1^L|r_2^L|...|r_{q_s}^L) = t_{q_s}(r_1^L|r_2^L|...|r_{q_s}^L) = 0$. In this case $T' = q_s T$. However we can still get a tight bound on the *expected* running time of the simulator.

$$\begin{aligned} E[T'] &= \sum_{r_1^L, r_1^R, ..., r_{q_s}^L, r_{q_s}^R \leftarrow \{0,1\}^{2^{2q_s\kappa}}} \left(\frac{1}{2^{2q_s\kappa}} \sum_{i=1}^{q_s} \left(t_i(r_1^L|r_2^L|...|r_i^L) + t_i(r_1^L|r_2^L|...|r_i^R) \right) \right) \\ &= \left(t_{q_s}(r_1^L|r_2^L|...|r_{q_s}^R) + \sum_{i=1}^{q_s-1} t_i(r_1^L|r_2^L|...|r_i^L) \right) + \sum_{i=1}^{q_s} t_i(r_1^L|r_2^L|...|r_i^L) \\ &+ \sum_{i=1}^{q_s-1} \left(t_i(r_1^L|r_2^L|...|r_i^R) - t_i(r_1^L|r_2^L|...|r_i^L) \right) = 2T \end{aligned}$$

The last equation is because according to the definition the first two terms add up to $2T$ and for any function f defined on any domain D, $\sum_{x,y\in D^2} (f(x) - f(y)) = 0$.

There are two multi-exponentiations and one single exponentiation in initialization phase, one single exponentiation per each query to $\mathcal{H}_3$ and one single exponentiation per each query to $\mathcal{G}_1$. The reduction also makes three single exponentiations, at most

$4n_{max} + 2$ multi-exponentiations and at most $4n_{max}$ multiplications per each signing query in the signing phase, and $2(n-1)$ multi-exponentiations in KVrfy and two multi-exponentiations in Vrfy algorithms and n_{max} single exponentiations in finalization phase. Therefore, ignoring the cost of hash-table lookups, assuming that a two or three exponent multi-exponentiation take at most 25% more time than an exponentiation, the total running time of the algorithm $\mathcal{B}$ can be upper-bounded as

$$t' \leq 2t + (q_h + 6(n_{max}+1)(q_s+1))t_e + 4q_s n_{max} t_m$$

Note on the Security Reduction. There are two crucial tricks we use which allow us to compress the protocol to three rounds and maintain exact security reduction. First, we use the signature randomization technique introduced by Katz and Wang [KW03], which in the multisignature setting extends the signed message with n bits instead of just one. This idea allows to replace the $1/q_s$ factor encountered in the security reduction for the full-domain hash signature with a constant factor of $1/2$. However, to make the exact security reduction to go through, each player must have its own bit for the reduction to play with, and hence the signature size grows by exactly n bits. However we use an intermediate hash function to avoid this linear blowup and maintain constant signature size. Secondly, we use the ZK proofs to ensure that the adversary cannot manipulate values z_j and B_j provided by potentially corrupt players in the second round. Inclusion of such ZK proofs in the protocol fixes these values across instances of the same adversarial algorithm executing on the same inputs. This enables a very simple rewinding schedule in the simulation: The simulator attempts the execution once, learns the adversarial contributions z_j, B_j if the adversary reveals them accompanied with a correct ZK proof, rewinds the adversary, and equipped with the knowledge of the values the adversary is bound to use again (we are aided here by the fact that the soundness of the ZK proofs we use is unconditional), the simulator then successfully straight-line simulates the protocol on behalf of an honest player. If the adversary fails to reveal correct z_j, B_j values, the simulator has an even easier job because the first execution already forms a correct simulation, since the honest player would abandon the protocol if any protocol participant failed in this way. Thus the simulator repeats an execution of every instance of the signature scheme at most twice. At surface, the time of such simulation seems to be at most twice the total time of the adversary. However, upon closer inspection, it is clear that there are adversaries for which the running time of the simulator is q_s times the running time of the adversary. Namely in an extreme scenario in which the adversary takes its maximum time on the random coins that run it in the first execution in the simulation and takes zero time on the remaining random coins. However as we argue in the proof of the theorem 2 the expected running time of the simulation is at most twice as the expected running time of the adversary.

References

[BGLS03] Boneh, D., Gentry, C., Lynn, B., Shacham, H.: Aggregate and verifiably encrypted signatures from bilinear maps. In: Biham, E. (ed.) EUROCRYPT 2003. LNCS, vol. 2656, pp. 416–432. Springer, Heidelberg (2003)

[BJ08] Bagherzandi, A., Jarecki, S.: Multisignatures using proofs of secret key possession, as secure as the diffie-hellman problem. ePrint Archive (2008)

[BLS04] Boneh, D., Lynn, B., Shacham, H.: Short signatures from the weil pairing. J. Cryptology 17(4), 297–319 (2004)

[BN06] Bellare, M., Neven, G.: Multi-signatures in the plain public-key model and a general forking lemma. In: ACM Conference on Computer and Communications Security, pp. 390–399 (2006)

[BNN07] 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)

[Bol03] 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)

[Bon98] Boneh, D.: The decision diffie-hellman problem. In: Buhler, J.P. (ed.) ANTS 1998. LNCS, vol. 1423, pp. 48–63. Springer, Heidelberg (1998)

[Fis05] Fischlin, M.: Communication-efficient non-interactive proofs of knowledge with online extractors. In: Shoup, V. (ed.) CRYPTO 2005. LNCS, vol. 3621, pp. 152–168. Springer, Heidelberg (2005)

[GJ03] Goh, E.-J., Jarecki, S.: A signature scheme as secure as the diffie-hellman problem. In: Biham, E. (ed.) EUROCRYPT 2003. LNCS, vol. 2656, pp. 401–415. Springer, Heidelberg (2003)

[Har94] Harn, L.: Group-oriented (t,n) threshold digital signature scheme and digital multisignature. In: IEEE Proceedings on Computers and Digital Techniques, vol. 141(5), pp. 307–313 (1994)

[KW03] Katz, J., Wang, N.: Efficiency improvements for signature schemes with tight security reductions. In: ACM Conference on Computer and Communications Security, pp. 155–164 (2003)

[LHL94] Li, C.-M., Hwang, T., Lee, N.-Y.: Threshold-multisignature schemes where suspected forgery implies traceability of adversarial shareholders. In: De Santis, A. (ed.) EUROCRYPT 1994. LNCS, vol. 950, pp. 194–204. Springer, Heidelberg (1995)

[LOS+06] 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)

[MOR01] Micali, S., Ohta, K., Reyzin, L.: Accountable-subgroup multisignatures: extended abstract. In: ACM Conference on Computer and Communications Security, pp. 245–254 (2001)

[MW99] Maurer, U.M., Wolf, S.: The relationship between breaking the diffie-hellman protocol and computing discrete logarithms. SIAM J. Comput. 28(5), 1689–1721 (1999)

[MW00] Maurer, U.M., Wolf, S.: The diffie-hellman protocol. Des. Codes Cryptography 19(2/3), 147–171 (2000)

[OO91] Ohta, K., Okamoto, T.: A digital multisignature scheme based on the fiat-shamir scheme. In: Matsumoto, T., Imai, H., Rivest, R.L. (eds.) ASIACRYPT 1991. LNCS, vol. 739, pp. 139–148. Springer, Heidelberg (1993)

[OO99] Ohta, K., Okamoto, T.: Multisignature schemes secure against active insider attacks. IEICE Transactions on Fundamentals of Electronics Communications and Computer Sciences E82-A(1), 21–31 (1999)

[PKC00] PKCS#10. Certification request syntax standard. In: RSA Data Security, Inc. (2000)

[PS00] Pointcheval, D., Stern, J.: Security arguments for digital signatures and blind signatures. J. Cryptology 13(3), 361–396 (2000)

[RY07] 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)

[Sho00] Shoup, V.: Practical threshold signatures. In: Preneel, B. (ed.) EUROCRYPT 2000. LNCS, vol. 1807, pp. 207–220. Springer, Heidelberg (2000)