Multisignatures Secure under the Discrete Logarithm Assumption and a Generalized Forking Lemma

Ali Bagherzandi
Dept. of Computer Science
University of California, Irvine
zandi@ics.uci.edu

Jung Hee Cheon
Dept. of Math. Sciences
Seoul National University
jhcheon@snu.ac.kr

Stanisław Jarecki
Dept. of Computer Science
University of California, Irvine
stasio@ics.uci.edu

ABSTRACT

Multisignatures allow n signers to produce a short joint signature on a single message. Multisignatures were achieved in the plain model with a non-interactive protocol in groups with bilinear maps, by Boneh et al [4], and by a three-round protocol under the Discrete Logarithm (DL) assumption, by Bellare and Neven [3], with multisignature verification cost of, respectively, $O(n)$ pairings or exponentiations. In addition, multisignatures with $O(1)$ verification were shown in so-called Key Verification (KV) model, where each public key is accompanied by a short proof of well-formedness, again either with a non-interactive protocol using bilinear maps, by Ristenpart and Yilek [15], or with a three-round protocol under the Diffie-Hellman assumption, by Bagherzandi and Jarecki [1].

We improve on these results in two ways: First, we show a two-round $O(n)$-verification multisignature secure under the DL assumption in the plain model, improving on the three-round protocol of [3]. Second, we show a two-round $O(1)$-verification multisignature secure under the DL assumption in the KV model, improving on assumptions in [15, 1] and communication rounds in [1]. Exact security of both schemes matches (in ROM) that of Schnorr signatures. The reduced round complexity is due to a new multiplicatively homomorphic equivocable commitment scheme which can be of independent interest. Moreover, our KV model scheme is enabled by a generalized forking lemma, which shows that standard non-interactive zero-knowledge (NIZK) proofs of knowledge in ROM admit efficient *simultaneous* post-execution extraction of witnesses of all proof instances. As a consequence of this lemma, any DL-based multisignature secure in so-called Knowledge-of-Secret-Key model can be implemented in the KV model using standard ROM-based NIZK's of DL as proofs of key well-formedness.

Categories and Subject Descriptors

E.1 [**Data Structures**]; F.2.1 [**Analysis of Algorithms and Problem Complexity**]: Numerical Algorithms and Problems

General Terms

Algorithms, Reliability, Security

1. INTRODUCTION

A multisignature protocol allows a group of n players to sign a common message in such a way that instead of n separate signatures the players produce a short string, called a multisignature, which can be then verified against the set of the public keys of these n players. Such scheme provides advantages over standard signatures if the size of the multisignature is that of a single standard signature rather than n signatures, and even more so if the verification efficiency is comparable to single signature verification instead of n signature verifications. Applications of multisignatures include cases where the set of signers is small, e.g. distribution of certificate authorities, or authentication of routes in mobile networks, but potential applications can also include large sets of signers, e.g. in aggregation of broadcast acknowledgements, where it is especially beneficial to reduce both multisignature size *and* verification time.

Rogue Key Attacks, KOSK Assumption. Multisignature protocols based on various signature schemes are possible because of homomorphic properties of arithmetic operations involved in signature algorithms. For example, a BLS signature [5] on message m under public key $y_i = g^{x_i}$ is $\sigma_i = H(m)^{x_i}$. Therefore a multisignature σ can be created as $\sigma = \sigma_1 * ... * \sigma_n$, and it can be verified under the combined public key $y = y_1 * ... * y_n$ because $DL(g, y) = DL(H(m), \sigma)$. However, the same homomorphic properties often enable so-called "rouge key attacks" on such schemes. For example, if an adversary picks public key $y_2 = g^x/y_1$ for some y_1 and any chosen x, he can then issue valid multisignatures under keys $\{y_1, y_2\}$. Micali et al [12] showed how to avoid such rogue key attacks under so-called "Knowledge of Secret Key" (KOSK) assumption, which requires the adversary to essentially provide a secret key for every public key it chooses.

Key Verification or Registration Models. Micali et al implemented the KOSK assumption via an interactive pre-processing protocol involving all potential signers [12]. However, it can also be implemented in a *Key Verification* (KV) model [1], where each key y_i admitted in a multisignature verification procedure must be accompanied by a valid proof of well-formedness π_i, e.g. if π_i's are non-interactive concurrently extractable proofs of secret key knowledge. A version of this model was introduced by Ristenpart and Yilek [15] as a *Key Registration* (KR) model for PKI, which stipulates that a Certification Authority (CA) can certify a public key only if its owner passes certain registration procedure. The KR model thus shifts the proof verification overhead from multisignature verifiers to the CA's. (However, as we explain below, the KR model requires non-standard trust assumptions on CA's.)

Prior Work Related to DL-based Multisignatures. Multisignature schemes proven secure in the KR model in [15] use non-interactive proofs of key well-formedness and hence they are secure also in the KV model. Technically, the non-interactive proofs used by the schemes of [15] are *not* concurrently extractable proofs of knowledge. Instead, they are NIZK proofs of DL equality, called "proofs of secret key possession" in [15], which do not guarantee efficient concurrent witness extraction, yet they turn out to suffice for security of multisignatures based on the Gap Diffie-Hellman (DH) assumption [15] or on Computational or Decisional DH assumptions in the Random Oracle Model (ROM) [1], following a general paradigm of replacing proofs of knowledge (e.g. of discrete logarithm) with proofs of computational ability (e.g. of correct exponentiation of a challenge), used e.g. in [17]. However, this paradigm seems to yield only schemes secure under assumptions related to the DH assumption, and not the Discrete Logarithm (DL) assumption. Thus, to implement DL-based multisignatures in the KV model, one seemingly needs to resort to concurrently extractable zero-knowledge proofs of knowledge (ZKPK) of discrete logarithm. Such proofs remain impractical in the standard model (e.g. [8]), but in ROM, due to the results of Fischlin [9], concurrent ZKPK's of DL can be achieved in a way that is arguably efficient enough *if* these ZKPK's are verified by CA's, but less so if they are attached to certificates and verified by multisignature receivers as part of certificate verification. (In practice they seem to require about 10 times more bandwidth and computation than standard ROM-based NIZK's.) However, note that trusting the CA's to perform proof verifications places non-standard trust assumptions on CA's, because *all* CA's must be trusted to perform those checks. In particular, a multisignature in the KR model becomes insecure if a key of a single participant in multisignature generation is certified by an untrustworthy CA. It has been an open problem whether *standard* ROM-based NIZK of DL, e.g. (r, s) s.t. $g^s = ry^{H(y,r)}$ can be used instead of Fischlin's NIZK's to implement the KOSK assumption, thus leading to efficient multisignatures in the KV model.

Multisignatures have also been proposed in the standard PKI setting using groups with bilinear maps by Boneh et al [4, 2], and under the DL assumption by Bellare and Neven [3], but the multisignature verification in these schemes requires $O(n)$ pairings or exponentiations, respectively. Moreover, the DL-based scheme of [3] requires 3 rounds of interaction, which makes the scheme less convenient for applications where multisignature generation could be piggybacked on a 2-round application protocol, e.g. aggregation of authentication in route discovery (see e.g. [11]) or aggregation of acknowledgments to a broadcast (see e.g. [6]).

Our Results. We provide two new multisignature schemes based on the DL assumption, both with two-round protocols. The first scheme (Section 4.2) is secure in the KV model, formally defined in Section 2. It improves on a scheme implied by Micali et al [12] and Fischlin's NIZKs [9] by using standard ROM-based NIZK of DL as a proof of key well-formedness, thus reducing its size and verification time to a minimum, and settling the open question mentioned above. Moreover, the exact security of our scheme matches (in ROM) that of standard DL-based signatures by Schnorr [16], as given by the forking lemma analysis of Pointcheval and Stern [14]. Such exact security seems unlikely to hold for the scheme implied by the results of [12, 9]. Our second scheme (Section 6) is secure in the plain model and uses $O(n)$ exponentiations in verification, but improves on the scheme of [3] by reducing the protocol rounds to two, which seems minimal for DL-based schemes, while also preserving the same exact security as that of Schnorr signatures. Our schemes have several other convenient features: (1) A signer can concurrently engage in any number of multisignature instances; (2) A signer doesn't need to know anything about other participating signers; (3) The message to be signed can be provided in the second (last) protocol round; (4) Both schemes use standard DL-based keys and can safely reuse e.g. the keys used for Schnorr signatures.

These results are enabled by two contributions of general interest. The low round complexity of both schemes is due to a new multiplicatively homomorphic equivocable commitment scheme (Section 4). As shown by Damgard [7], equivocable commitments due to Pedersen [13] imply a practical 3-round straight-line simulatable ZKPK of DL in the CRS model. Our commitment scheme can play the same role but it in addition it allows aggregation of n instances of such proofs, thus compacting them to allow a short multisignature, with an efficient reduction enabled by straight-line simulatability of the proof system. (We note that our commitment scheme provides only restricted equivocability, but enough for straight-line simulation of ZKPK of DL.) Secondly, short proofs of key well-formedness in our KV model scheme are enabled by a generalized forking lemma (Section 3), which shows that witnesses to polynomially many instances of standard ROM-based NIZK's can be efficiently *simultaneously* extracted *after* adversary ends its execution (as opposed to on-line extraction in Fischlin's NIZK's). This implies that any DL-based multisignature secure under the KOSK assumption is secure in the KV model in ROM when standard ROM-based NIZK's are used as proofs of key well-formedness. An *expected* polynomial-time post-execution extraction of all witnesses in such proofs was previously shown by Jens Groth in [10], so our contribution is a strict polynomial-time extraction procedure which matches up to an $O(n^2)$ factor, where n is the number of proof instances, the time/probability bounds given by the Bellare-Neven version [3] of the Pointcheval-Stern forking lemma [14].

Notation and Setting. We use G to denote a multiplicative group of prime order q. All arithmetic operations are either done modulo q, when involving elements in $\mathbb{Z}_q$, or they are operations in G.

2. MULTISIGNATURE SCHEMES

A Multisignature Syntax. We define a multisignature scheme in the key verification model as a tuple $\mathsf{MS} = (\mathsf{Setup}, \mathsf{KGen}, \mathsf{MSign}, \mathsf{Vrfy}, \mathsf{KVrfy})$ where Setup, KGen, Vrfy and KVrfy are efficient probabilistic algorithms, and MSign is a distributed protocol *s.t.*

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

This set of procedures must satisfy the following *completeness* properties: Let $\mathsf{par} \leftarrow \mathsf{Setup}(1^\kappa)$. First, for any tuple (sk, pk, π)

```
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 procedure MSign on inputs $(\mathsf{par}, m, sk^*)$, forwarding messages to and from $\mathcal{A}$.
      (We allow $\mathcal{A}$ to make any number of such queries concurrently.)
  When $\mathcal{A}$ halts, parse its output as $(m, \sigma, \{(pk_2, \pi_2), (pk_3, \pi_3), ..., (pk_n, \pi_n)\})$. Set $\mathsf{PKSet} = \{pk^*\} \cup \{pk_2, pk_3, ..., pk_n\}$.
  If $(\mathsf{KVrfy}(\mathsf{par}, pk_i, \pi_i) = 1$ for all $i = 2$ to $n) \wedge (m \notin \mathsf{List}) \wedge (\mathsf{Vrfy}(\mathsf{par}, m, \mathsf{PKSet}, \sigma) = 1)$ then return 1, otherwise return 0.
```

Figure 1: Chosen Message Attack against a Multisignature Scheme

outputted by $\mathsf{KGen}(\mathsf{par})$, $\mathsf{KVrfy}(\mathsf{par}, pk, \pi) = 1$. Second, for any number n and any message m, if for $i = 1..n$ one generates (sk_i, pk_i, π_i) by running $\mathsf{KGen}(\mathsf{par})$ and executes MSign on input par, m, and sk_i, then assuming that all messages between these players are delivered correctly, each player outputs the same string σ that moreover satisfies

$$\mathsf{Vrfy}(\mathsf{par}, m, \{pk_1, pk_2, ...pk_n\}, \sigma) = 1$$

Remarks on the assumptions behind the syntax:
(1) In the security game in figure 1 we take a simplifying assumption that the Setup procedure is executed by an honest party. However, the public parameters in our two schemes are only needed to define a multiplicative group of prime order where the DL assumption holds, and such parameters can be chosen by any party.
(2) The syntax of a multisignature scheme in the KV model is a simplification of the syntax used by [15], which models potentially interactive key registration processes. Here we only allow non-interactive proofs π_i of well-formedness of key y_i. Such proofs can be verified *either* by CA's as part of the key registration process (as in the KR model of [15]) *or* by multisignature verifiers, e.g. together with validation of a PKI certificate on y_i.
(3) If the proof of validity of the public key is set to empty string and the algorithm KVrfy just returns true, then the above definition is equivalent to the definition of the multisignature schemes in the plain model as proposed in [3].
(4) Unlike in the definition of multisignatures used by [12] and [3], we do not require the set of participant identities and/or the set of their public keys as inputs to 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 cryptographic protocol. The schemes secure in this setting provide additional flexibility to applications of multisignatures; because in many applications a signer might care only about the message it is signing and not about the identities of the other signers. (Otherwise they can always include the list of participating players in the signed message.) In such applications protocols of [12, 3] would have to be preceded by an additional communication round where participants broadcast their identities and/or keys.
Multisignature security in Key Verification model. As in previous works on multisignatures, e.g. [12, 3, 15], we define security of a multisignature scheme as universal unforgeability under a chosen message attack against a single honest player. Namely we define the *adversarial advantage* of $\mathcal{A}$ against the multisignature scheme $\mathsf{MS} = (\mathsf{Setup}, \mathsf{KGen}, \mathsf{MSign}, \mathsf{Vrfy}, \mathsf{KVrfy})$ as a probability that experiment $\mathbf{Exp}^{\mathsf{uu-cma}}_{\mathsf{MS}}(\mathcal{A})$ described in figure 1 outputs 1, *i.e.*

$$\mathbf{Adv}^{\mathsf{uu-cma}}_{\mathsf{MS}}(\mathcal{A}) = Pr[\mathbf{Exp}^{\mathsf{uu-cma}}_{\mathsf{MS}}(\mathcal{A}) = 1] \qquad (1)$$

where the probability goes over the random coins of the adversary and all the randomness used in the experiment. We call a multisignature scheme (t, ϵ, n, q_s)-secure if it holds that $\mathbf{Adv}^{\mathsf{uu-cma}}_{\mathsf{MS}}(\mathcal{A}) \leq \epsilon$ for every adversary $\mathcal{A}$, that runs in time at most t, makes at most q_s signature queries and produces forgeries on behalf of n parties. In random oracle model we also consider a notion of a $(t, \epsilon, n, q_s, q_h)$-secure multisignature scheme where $\mathcal{A}$ is an adversary 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 over random hash functions.

We note that in [12] and [3] the notion of CMA forgery is broader than the one we consider above: As we pointed out in remark (4) above, their signers take as input the set of public keys of all participating players PKSet along with the message m as input. Moreover, their notion of multisignature security treats the multisignature effectively as a signature on a *pair* (m, PKSet): Their notion of forgery is extended to include 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. However, a scheme secure according to our notion implies a scheme secure in this stronger model if every message m input into our multisignature protocol is simply amended by the set of public keys PKSet.

3. GENERALIZED FORKING LEMMA

Consider an experiment in which an adversary $\mathcal{A}$, on input par, interacts with a random oracle $H : \{0,1\}^* \rightarrow \mathbb{Z}_q$. Denote the randomness involved in this execution as $f = (\rho, h_1, h_2, ..., h_{q_h})$, where ρ is $\mathcal{A}$'s random input, h_j is the j-th response of H, and q_h is the maximum number of hash queries $\mathcal{A}$ makes. Let Ω denote the set of all vectors f. The probability in all experiments we consider goes over $f \in \Omega$, unless noted otherwise. We consider as adversary's *success* an event that $\mathcal{A}$ outputs a pair $(J, \{\phi_j\}_{j \in J})$ where J is a (non-empty) set of up to n indexes $J \subseteq \{1, ..., q_h\}$. By convention we assume that if $\mathcal{A}$ fails then it outputs $(J, \{\phi_j\}_{j \in J})$ s.t. $J = \emptyset$. Otherwise we assume that $|J| = n$, by treating J as a *multiset* and allowing repetitions, and we assume that if $J = \{j_1, ..., j_n\}$ then $j_1 \leq ... \leq j_n$. For fixed par, let $S \subseteq \Omega$ be the set of vectors f s.t. $\mathcal{A}(\mathsf{par}, f)$ succeeds, and let $\epsilon = \Pr[S]$. Denote the index set J output by $\mathcal{A}(\mathsf{par}, f)$ on $f \in S$ as $Ind(f)$. Let $f_j = (\rho, h_1, ..., h_{j-1})$, e.g. $f_1 = \rho$, $f_2 = (\rho, h_1)$, etc. The *forking lemma* of Pointcheval and Stern [14] (see also Bellare and Neven [3]), considers a restricted class of algorithms $\mathcal{A}$ where $n = 1$. The lemma involves an execution of a *forking algorithm* $F_{\mathcal{A}}$ which runs $\mathcal{A}$ on random $f \in \Omega$, and if $f \in S$ then it runs $\mathcal{A}$ again, on random f' chosen subject to the constraint that $f'_j = f_j$ where $\{j\} = Ind(f)$ (here $n = 1$, so $Ind(f)$ is a singleton if $f \in S$). Let $f' = (\rho, h_1, ..., h_{j-1}, h'_j, ..., h'_{q_h})$. We say that algorithm $F_{\mathcal{A}}$

has a *forking success* if both f and f' are in S, $Ind(f') = Ind(f)$, and $h'_j \neq h_j$. The forking lemma lower-bounds the probability of $F_{\mathcal{A}}$'s forking success in the above experiment by $(\epsilon - 1/q)^2/q_h$ (see lemma 1 in [3]). A common application of this lemma is to ROM-based NIZK's created via the Fiat-Shamir heuristic, where value ϕ_j in $\mathcal{A}$'s output is a pair (x_j, π_j) where x_j is an instance of a language membership problem L and $\pi_j = (a_j, h_j, z_j)$ is a non-interactive zero-knowledge proof for $x_j \in L$, with $h_j = H(x_j, a_j)$ playing the role of a verifier's challenge. A successful forking algorithm outputs two such proofs involving the same instance x_j, the same prover's first message a_j, and different challenges $h_j \neq h'_j$, which for many proof systems allows for efficient extraction of a witness for x_j in L.

We describe this generalized forking process as algorithm $GF_{\mathcal{A}}$, where $\hat{\epsilon}$ denotes the expected value of ϵ for adversary $\mathcal{A}(\mathsf{par})$ when par is uniformly chosen.

```
Algorithm $GF_{\mathcal{A}}$ on inputs par:
1. Pick $f = (\rho, h_1, ..., h_{q_h}) \leftarrow \Omega$.
2. Compute $(J, \{\phi_j\}_{j \in J}) \leftarrow \mathcal{A}(\mathsf{par}, f)$.
3. If $J = \emptyset$ then stop (and *fails*).
4. Let $J = \{j_1, ..., j_n\}$, $X = \{(h_j, \phi_j)\}_{j \in J}$ and $X' = \{\}$.
5. For $i = 1$ to $n$, repeat lines 5.1-5.2
5.1.  Set $\mathsf{succ}_i = 0$, $k_i = 0$ and $k_{max} = 8nq_h/\hat{\epsilon} * \ln(8n/\hat{\epsilon})$.
5.2.  Repeat lines 5.2.1-5.2.4 until $\mathsf{succ}_i = 1$ or $k_i > k_{max}$
5.2.1     Increment $k_i$. Pick random $f'$ in $\Omega$ s.t. $f'_{j_i} = f_{j_i}$.
5.2.2     Let $f' = (\rho, h_1, ..., h_{j_i\text{-}1}, h'_{j_i}, ..., h'_{q_s})$.
5.2.3     Compute $(J', \{\phi'_j\}_{j \in J'}) \leftarrow \mathcal{A}(\mathsf{par}, f')$.
5.2.4     If $h'_{j_i} \neq h_{j_i}$, $J' \neq \emptyset$, and $j_i \in J'$, then
              add $(h'_{j_i}, \phi'_{j_i})$ to $X'$ and set $\mathsf{succ}_i = 1$.
6. If for all $i = 1$ to $n$ $\mathsf{succ}_i = 1$, then Output $(X, X')$ .
6.1   Otherwise stop (and *fail*).
```

LEMMA 1. **[Generalized Forking Lemma]** *Let* IG *be a randomized algorithm that generates* par *and* $\mathcal{A}$ *be a randomized algorithm making at most* q_h *hash queries s.t.* $\mathcal{A}(\mathsf{par})$ succeeds *(i.e. outputs* $(J, \{\phi_j\}_{j \in J})$ *s.t.* $|J| = n$*) with probability* $\hat{\epsilon}$*, where the probability goes over* $\mathsf{par} \xleftarrow{r} \mathsf{IG}$ *and* $f \xleftarrow{r} \Omega$*. Let* $q > 8nq_h/\hat{\epsilon}$*. Then algorithm* $GF_{\mathcal{A}}(\mathsf{par})$ *has a* forking success *(i.e. outputs two* n*-element lists* (X, X')*) with probability* $\mathsf{frk} \geq \hat{\epsilon}/8$*, where probability goes over coins of* IG *and* $GF_{\mathcal{A}}$*.*

Note that if the running time of $\mathcal{A}(\mathsf{par})$ is bounded above by $t(\mathsf{par})$ then the running time of $GF_{\mathcal{A}}(\mathsf{par})$ is at most $t(\mathsf{par}) * 8n^2q_h/\hat{\epsilon} * \ln(8n/\hat{\epsilon})$. Hence the expected running time of $GF_{\mathcal{A}}(\mathsf{par})$ over all par is bounded above by the expected running time of $\mathcal{A}(\mathsf{par})$ multiplied by $8n^2q_h/\hat{\epsilon} * \ln(8n/\hat{\epsilon})$.

In the proof we will rely on the following version of the "splitting lemma" of [14]. Let $A, B, S \subseteq X \times Y$ be any sets s.t. $B = \{(x,y) \mid \Pr_{y' \in Y}[(x,y') \in A] \geq \delta\}$ and $A \subseteq S$.

LEMMA 2. **[Splitting Lemma]** *For all* A, B, S *as above:*
(1) $\Pr[B|A] \geq 1 - \delta/\Pr[A]$
(2) $\Pr[A \cap B] \geq \Pr[A] - \delta$
(3) $\Pr[A \cap B \mid S] \geq \Pr[A|S] - \delta/\Pr[S]$

PROOF. Inequality (1) follows because $\Pr[B|A] < 1 - \delta/\Pr[A]$ implies

$$\begin{aligned}\Pr[A] &= \Pr[A \cap B] + \Pr[A \cap \bar{B}] \\ &= \Pr[B|A] * \Pr[A] + \Pr[A|\bar{B}] * \Pr[\bar{B}] \\ &< (1 - \delta/\Pr[A]) * \Pr[A] + \delta * 1 = \Pr[A]\end{aligned}$$

Inequality (2) follows from (1) because $\Pr[A \cap B] = \Pr[B|A] * \Pr[A]$. As for inequality (3), since $A \cap B \subseteq S$, we have $\Pr[A \cap B \mid S] = \Pr[A \cap B]/Pr[S]$, and thus by (2) we have $\Pr[A \cap B \mid S] \geq (\Pr[A] - \delta)/\Pr[S] = \Pr[A \mid S] - \delta/\Pr[S]$. □

PROOF OF LEMMA 1. Let $\epsilon(\mathsf{par})$ and $\mathsf{frk}(\mathsf{par})$ be the success probability of $\mathcal{A}$ and $GF_{\mathcal{A}}$ for fixed input par.

Let $\mathcal{P}$ be the set of all possible par and $\mathcal{P}'$ be the set of par satisfying $\epsilon(\mathsf{par}) > \hat{\epsilon}/2$. We will argue that for an input $\mathsf{par} \in \mathcal{P}'$, $\mathsf{frk}(\mathsf{par}) \geq \epsilon(\mathsf{par})/4$. Then we have

$$\begin{aligned}\mathsf{frk} &\geq \frac{1}{|\mathcal{P}|} \sum_{\mathsf{par} \in \mathcal{P}'} \mathsf{frk}(\mathsf{par}) \\ &\geq \frac{1}{|\mathcal{P}|} \left(\sum_{\mathsf{par} \in \mathcal{P}} \epsilon(\mathsf{par})/4 - \sum_{\mathsf{par} \notin \mathcal{P}'} \epsilon(\mathsf{par})/4 \right) \\ &\geq \hat{\epsilon}/4 - \hat{\epsilon}/8 = \hat{\epsilon}/8,\end{aligned}$$

where the last inequality follows because $\mathsf{par} \notin \mathcal{P}'$ implies that $\epsilon(\mathsf{par}) \leq \hat{\epsilon}/2$

Fix an input instance $\mathsf{par} \in \mathcal{P}'$ and let $\epsilon = \epsilon(\mathsf{par})$ and $\mathsf{frk} = \mathsf{frk}(\mathsf{par})$. For $f \in S$ define $i\text{-}Ind(f)$ as the i-th element of $Ind(f)$. Let $E = E_1 \cap E_2 \cap ... \cap E_n$ where E_i's are defined as follows:

$$\begin{aligned}A_j &= \{f \in S \mid j \in Ind(f)\} \\ B_j &= \{f \in \Omega \mid \Pr_{f' \in \Omega}[f' \in A_j \mid f'_j = f_j] \geq \epsilon/(2nq_h)\} \\ E_i &= \{f \in S \mid f \in B_j \text{ where } j = i\text{-}Ind(f)\}\end{aligned}$$

We will argue that $\mathsf{frk} > \epsilon/4$ follows from the following inequality:

$$\Pr[E|S] \geq 1/2 \tag{2}$$

This is because if $\Pr[E|S] \geq 1/2$ then f chosen in step 1 of $GF_{\mathcal{A}}$ is in E with probability $\Pr[E|S] * \Pr[S] \geq \epsilon/2$. Moreover, if $f \in E \subseteq S$ then by definition of E we have that $f \in B_{j_i}$ for each $j_i \in Ind(f) = \{j_1, ..., j_n\}$. Therefore, for each i from 1 to n, by definition of B_{j_i}, the probability that f' chosen in line 5.2.2 satisfies $f' \in A_{j_i}$ (and hence $j_i \in Ind(f')$) is at least $\epsilon/2nq_h$. Since the probability that $h'_{j_i} = h_{j_i}$ is at most $1/q < \hat{\epsilon}/8nq_h < \epsilon/4nq_h$, the probability that condition in line 5.2.4 is satisfied is at least $\epsilon/4nq_h$. Since $\epsilon > \hat{\epsilon}/2$, the probability that all $8nq_h/\hat{\epsilon} * \ln(8n/\hat{\epsilon})$ executions of loop 5.2 fail (for any given i) is at most $\epsilon/4n$. By the union bound, the probability that the rewinding procedure in step 5 fails for some i is thus at most $\epsilon/4$. Hence, the probability frk that $GF_{\mathcal{A}}$ succeeds is at least the probability that $f \in E \subseteq S$ and that the procedure in step 5 does not fail, which is at least $\epsilon/2 - \epsilon/4 = \epsilon/4$.

It remains to show that (2) holds. We will first argue:

$$\forall_{i=1,..,n} \quad \Pr[E_i|S] \geq 1 - 1/(2n) \tag{3}$$

It's easy to see that (3) implies (2) (see explanations below):

$$\begin{aligned}\Pr[E|S] &= \Pr[\cap_{i=1}^{n} E_i \mid S] && (4)\\ &= 1 - \Pr[\cup_{i=1}^{n} \bar{E}_i \mid S] && (5)\\ &\geq 1 - \sum\nolimits_{i=1}^{n} Pr[\bar{E}_i \mid S] && (6)\\ &= 1 - \sum\nolimits_{i=1}^{n} (1 - Pr[E_i \mid S]) && (7)\\ &\geq (1 - \sum\nolimits_{i=1}^{n} (1 - (1\text{-}1/(2n)))) = 1/2 && (8)\end{aligned}$$

Step (6) follows by the union bound while step (8) follows from inequality (3).

Thus it remains to argue that (3) holds. Let us define the following:

$$\begin{aligned}A_j^i &= \{f \in S \mid j = i\text{-}Ind(f)\} \\ B_j^i &= \{f \in \Omega \mid \Pr_{f' \in \Omega}[f' \in A_j^i \mid f'_j = f_j] \geq \epsilon/(2nq_h)\}\end{aligned}$$

Note that if $f \in E_i \subseteq S$ then $f \in A^i_j \cap B_j$ for $j = i\text{-}Ind(f)$, and vice versa, which implies:

$$\forall_{i=1,..,n} \;\; E_i = \bigcup_{j=1}^{q_h} (A^i_j \cap B_j) \qquad (9)$$

The following sequence of inequalities implies (3) and hence concludes the proof (see the explanations below):

$$\begin{aligned} \Pr[E_i|S] &= \sum_{j=1}^{q_h} \Pr[A^i_j \cap B_j \mid S] && (10)\\ &\geq \sum_{j=1}^{q_h} \Pr[A^i_j \cap B^i_j \mid S] && (11)\\ &\geq \sum_{j=1}^{q_h} (\Pr[A^i_j \mid S] - \frac{\epsilon}{2nq_h \Pr[S]}) && (12)\\ &= 1 - 1/(2n) && (13) \end{aligned}$$

Equality (10) follows from (9) and the fact that sets $A^i_j \cap B_j$ partition set E_i into q_h non-intersecting subsets. Inequality (11) holds because $B^i_j \subseteq B_j$ for all i. Inequality (12) follows from part (3) of the splitting lemma (lemma 2) and the definition of B^i_j. Equality (13) follows because for any i sets A^i_j partition set S into q_h non-intersecting subsets, so $\sum_{j=1}^{q_h} \Pr[A^i_j|S] = \Pr[S|S] = 1$, while $q_h * (\epsilon/(2nq_h))/Pr[S] = 1/(2n)$. $\square$

4. MULTIPLICATIVELY HOMOMORPHIC EQUIVOCABLE COMMITMENTS

Commitments: We model a commitment scheme $\mathcal{C}$ in common reference string (CRS) model as a tuple of probabilistic poly-time algorithms CSetup, CGen, Com and Open, *s.t.*

- $\mathsf{cpar} \leftarrow \mathsf{CSetup}(1^\kappa)$ on input the security parameter κ, generates public parameters cpar, which also determine the commitment message space $\mathcal{M}$.
- $K \leftarrow \mathsf{CGen}(\mathsf{cpar})$, on input the parameters cpar, generates a commitment key K.
- $(c, d) \leftarrow \mathsf{Com}_K(m)$ generates the commitment c and decommitment d on message $m \in \mathcal{M}$.
- $\{0, 1\} \leftarrow \mathsf{Open}_K(c, d, m)$ determines if d is a valid decommitment of commitment c to message m.

These algorithms must satisfy a correctness requirement, namely if $\mathsf{cpar} \leftarrow \mathsf{CSetup}(1^\kappa)$, $K \leftarrow \mathsf{CGen}(\mathsf{cpar})$, $(c, d) \leftarrow \mathsf{Com}_K(m)$, then $\mathsf{Open}_K(c, d, m) = 1$. A commitment scheme must also satisfy requirements of hiding and binding. Below we define a statistical notion of hiding and a computational notion of binding since these are the variants of these notions which our scheme satisfies.
ϵ-Hiding: For every $\mathsf{cpar} \leftarrow \mathsf{CSetup}(1^\kappa)$, $m_0, m_1 \in \mathcal{M}$, and $K \leftarrow \mathsf{CGen}(\mathsf{cpar})$, there is less than ϵ statistical difference between distributions of c's output by $\mathsf{Com}_K(m_0)$ and $\mathsf{Com}_K(m_1)$. A commitment scheme is *perfectly* hiding if $\epsilon = 0$.
(t, ϵ)-Binding: For any $\mathcal{A}$ running in time t and any $\mathsf{cpar} \leftarrow \mathsf{CSetup}(1^\kappa)$ the probability of the following event is less than ϵ:

$$\mathsf{Open}_K(c, d_0, m_0) = \mathsf{Open}_K(c, d_1, m_1) = 1 \;\wedge\; m_0 \neq m_1$$

where $(c, d_0, d_1, m_0, m_1) \leftarrow \mathcal{A}(K)$, $K \leftarrow \mathsf{CGen}(\mathsf{cpar})$ and probability goes over coins of CGen and $\mathcal{A}$.

Homomorphic Commitments: To enable aggregation of outputs produced by n players into a single short multisignature, our commitment scheme must itself support aggregation. This is possible if the commitment scheme is *homomorphic*. We call a commitment scheme homomorphic for operation $\otimes_m : \mathcal{M} \times \mathcal{M} \rightarrow \mathcal{M}$ if there are efficiently computable operations $\otimes_c$ and $\otimes_d$ s.t. if $\mathsf{Open}_K(c_1, d_1, m_1) = 1$ and $\mathsf{Open}_K(c_2, d_2, m_2) = 1$, then $\mathsf{Open}_K(c, d, m) = 1$ for $c = c_1 \otimes_c c_2$, $d = d_1 \otimes_d d_2$, and $m = m_1 \otimes_m m_2$. For example, Pedersen commitment [13] is *additively* homomorphic, and therefore one can aggregate Pedersen commitments on separate messages into a single Pedersen commitment on the *sum* of these messages. Similarly, ElGamal encryption can be used to implement a *multiplicatively* homomorphic commitment scheme. Our construction is multiplicatively homomorphic, i.e. $\otimes_m$ is a group multiplication. For convenience we denote $\otimes_c$ as $\otimes$ and $\otimes_d$ as $\oplus$ in the sequel.

Restricted Equivocability: A commitment scheme is *equivocable* if there exists an efficient simulator that generates the commitment key K, indistinguishable from the real key, together with a *trapdoor* td. The trapdoor allows the simulator to create fake commitments which are indistinguishable from the real ones, but the simulator can later decommit them to *any* message. As far as we know, no commitment scheme has been proposed that is equivocable and multiplicatively homomorphic at the same time. Pedersen commitment is equivocable but only additively homomorphic, and while the commitment scheme based on ElGamal encryption is multiplicatively homomorphic, it is perfectly binding, and hence not equivocable (like every commitment scheme implemented with standard public-key encryption). Here we do not create such commitment scheme either. Instead we show a simple scheme which is multiplicatively homomorphic and has *restricted equivocability* in the sense that the simulator can open its fake commitments only to messages of certain special form. Namely, we'll show a multiplicatively homomorphic commitment scheme on $\mathcal{M} = G$, s.t. for any elements g, y^* in $G/\{1\}$, the simulator can open its fake commitment to a message of the form $m = g^\alpha (y^*)^\beta$, given any β in $\mathbb{Z}_q$ which the simulator receives *after* creating the fake commitment. Moreover, while value α can be chosen by the simulator after it receives β, the distribution of α's outputted by the simulator must be indistinguishable from the uniform distribution in $\mathbb{Z}_q$. Looking ahead, this type of restricted equivocability is enough to enable straight-line simulation of a Zero-Knowledge Proof of Knowledge (ZKPK) of discrete logarithm $\mathsf{DL}_g(y)$ (see section 4.2 below). This ZKPK is a basic building block of any multisignature based on DL, and the straight-line simulatable *and* aggregatable version of this proof system, enabled by our restricted-equivocable and multiplicatively homomorphic commitment scheme, leads to a multisignature scheme with fewer rounds and exact security matching that of standard discrete-log based signatures.

Formally, we model this type of restricted equivocability as follows. Let f be a family of efficiently computable functions indexed by the commitment parameter cpar, $f_{\mathsf{cpar}} : \mathcal{D} \times \mathcal{D} \times \mathcal{S} \rightarrow \mathcal{M}$ where $\mathcal{D}$ and $\mathcal{S}$, like $\mathcal{M}$, are defined by cpar. We call a commitment scheme *ϵ-equivocable for function (family) f* if there exist efficient algorithms tdCGen, tdCom and RstEqv, where $\mathsf{tdCGen}(\mathsf{cpar}, y^*) \rightarrow (K, td)$, $\mathsf{tdCom}_K(td) \rightarrow (\tilde{c}, st)$, $\mathsf{RstEqv}_K(td, st, \beta) \rightarrow (\tilde{d}, \alpha)$, *s.t.* for any cpar outputted by CSetup and any $y^* \in \mathcal{S}$ the following two properties hold: First, there is at most ϵ statistical difference between the distribution of K values output by $\mathsf{CGen}(\mathsf{cpar})$ and by $\mathsf{tdCGen}(\mathsf{cpar}, y^*)$. Second, for all $(K, td) \leftarrow \mathsf{tdCGen}(\mathsf{cpar}, y^*)$ and $\beta \in \mathcal{D}$, the following two distributions are identical:

$$\{\, (c, d, \alpha) \;\;|\;\; \alpha \xleftarrow{r} \mathcal{D};\; m = f_{\mathsf{cpar}}(\alpha, \beta, y^*);\; (c, d) \leftarrow \mathsf{Com}_K(m) \,\}, \qquad (14)$$

$$\{\, (\tilde{c}, \tilde{d}, \alpha) \;\;|\;\; (\tilde{c}, st) \leftarrow \mathsf{tdCom}_K(td);\; (\tilde{d}, \alpha) \leftarrow \mathsf{RstEqv}_K(td, st, \beta) \,\} \qquad (15)$$

4.1 DL-Based Commitment Scheme

We describe a commitment scheme denoted $\mathcal{CS}$, which is multiplicatively homomorphic on message space a multiplicative group G of prime order q, perfectly hiding, computationally binding under the DL assumption, and equivocable for function $f_g : \mathbb{Z}_q \times \mathbb{Z}_q \times G/\{1\} \to G$, $f_g(\alpha, \beta, y^*) = g^\alpha (y^*)^\beta$. The scheme has features of both Pedersen Commitment and ElGamal encryption:

- $\mathsf{CSetup}(1^\kappa)$: Set $\mathsf{cpar} \leftarrow g$, where g generates group G of prime order q large enough so that the DL assumption in group G holds with security parameter κ. To simplify notation we will assume that group G and its order q are implicitly defined by g.

- $\mathsf{CGen}(g)$: Pick $y \xleftarrow{r} G/\{1\}$ and $\alpha_1, \alpha_2 \xleftarrow{r} \mathbb{Z}_q$ s.t. $\alpha_1 \neq \alpha_2$. Set $h \leftarrow g^{\alpha_1}$, $z \leftarrow y^{\alpha_2}$, and $K \leftarrow (g, h, y, z)$.

- $\mathsf{Com}_K(m)$: Pick $r_1, r_2 \xleftarrow{r} \mathbb{Z}_q$ and return (c, d) where $c = (g^{r_1} h^{r_2}, y^{r_1} z^{r_2} m)$ and $d = (r_1, r_2)$.

- $\mathsf{Open}_K(c, d, m)$: Let $c = (c_1, c_2)$ and $d = (r_1, r_2)$. Return 1 iff $(c_1 = g^{r_1} h^{r_2}) \wedge (c_2 = y^{r_1} z^{r_2} m)$.

- $\mathsf{tdCGen}(\mathsf{cpar}, y^*)$: Pick $\gamma \xleftarrow{r} \mathbb{Z}_q$ and $\gamma_1, \gamma_2 \xleftarrow{r} \mathbb{Z}_q^*$. Let $h = g^\gamma$, $y = g^{\gamma_1}$, and $z = (y^*)^{\gamma_2}$. Set $K \leftarrow (g, h, y, z)$ and $td \leftarrow (y^*, \gamma, \gamma_1, \gamma_2)$, and return (K, td).

- $\mathsf{tdCom}_K(td)$: Pick $r, a, b \xleftarrow{r} \mathbb{Z}_q$. Set $st \leftarrow (r, a, b)$ and return $(\tilde{c}, st)$ where $\tilde{c} = (g^r, g^a (y^*)^b)$.

- $\mathsf{RstEqv}_K(td, st, \beta)$: Compute $r_2 = {\gamma_2}^{-1}(b - \beta)$, $r_1 = r - \gamma r_2$, and $\alpha = a - r_1 \gamma_1$ (all modulo q), and return $(\tilde{d}, \alpha)$ where $\tilde{d} = (r_1, r_2)$. Note that r_1, r_2, α satisfy the following set of equations mod q:

$$r_1 + \gamma r_2 = r \ , \quad \gamma_1 r_1 + \alpha = a \ , \quad \gamma_2 r_2 + \beta = b \qquad (16)$$

  Therefore for $m = g^\alpha (y^*)^\beta$ we have $(g^{r_1} h^{r_2}, y^{r_1} z^{r_2} m)$ $= (g^r, g^a (y^*)^b) = \tilde{c}$, and hence $\mathsf{Open}(\tilde{c}, \tilde{d}, m) = 1$.

We argue the claimed security properties:

Perfect Hiding: Note that the commitment produced by Com_K on $m = y^\tau$ is a pair $(g^{r_1 + \alpha_1 r_2}, y^{r_1 + \alpha_2 r_2 + \tau})$, and note that this is a pair of random elements in $G \times G$ for every τ if $\alpha_1 \neq \alpha_2$ and $(r_1, r_2) \xleftarrow{r} \mathbb{Z}_q \times \mathbb{Z}_q$.

Computational Binding: The commitment scheme $\mathcal{CS}$ is (t, ϵ)-binding if the DL problem in G is (t, ϵ)-hard. Indeed, an attacker $\mathcal{A}$ on binding can be used to solve the DL problem as follows: Given the DL challenge (g, h), the reduction picks $y, z \xleftarrow{r} G$ s.t. $\mathsf{DL}_g(y) \neq \mathsf{DL}_h(z)$, and runs $\mathcal{A}(g, h, y, z)$. By assumption, with probability ϵ, $\mathcal{A}$ outputs (c, d, m, d', m') *s.t.* $\mathsf{Open}_K(c, d, m) = \mathsf{Open}_K(c, d', m') = 1$ and $m \neq m'$. Denote $d = (r_1, r_2)$, $d' = (r_1', r_2')$, $\Delta r_1 = r_1 - r_1'$ and $\Delta r_2 = r_2 - r_2'$. Since $c = (g^{r_1} h^{r_2}, y^{r_1} z^{r_2} m) = (g^{r_1'} h^{r_2'}, y^{r_1'} z^{r_2'} m')$ it follows that $y^{\Delta r_1} z^{\Delta r_2} = m'/m$ and $g^{\Delta r_1} h^{\Delta r_2} = 1$. Therefore, either $\Delta r_1 = \Delta r_2 = 0$ or $\mathsf{DL}_g(h) = -\Delta r_1 / \Delta r_2$. But $\Delta r_1 = \Delta r_2 = 0$ implies that $y^{\Delta r_1} z^{\Delta r_2} = 1$ and hence $m = m'$. Thus, if $m \neq m'$, $\mathsf{DL}_g(h)$ can be computed as $-\Delta r_1 / \Delta r_2$.

Multiplicative Homomorphism: The commitment scheme $\mathcal{CS}$ is multiplicatively homomorphic on $\mathcal{M} = G$. Operators $\otimes$ and $\oplus$ are defined as follows: If $c = (c_1, c_2)$ and $c' = (c_1', c_2')$ then $c \otimes c' = (c_1 c_1', c_2 c_2')$, and if $d = (r_1, r_2)$ and $d' = (r_1', r_2')$ then $d \oplus d' = (r_1 + r_1', r_2 + r_2')$.

Restricted Equivocability: The commitment scheme $\mathcal{CS}$ is $(2/q)$-equivocable for function $f_g : \mathbb{Z}_q \times \mathbb{Z}_q \times G/\{1\} \to G$, where $f_g(\alpha, \beta, y^*) = g^\alpha (y^*)^\beta$. First note that the statistical difference between the distribution of keys $K = (g, h, y, z)$ produced by $\mathsf{CGen}(g)$ and $\mathsf{tdCGen}(g, y^*)$ is at most $2/q$, because elements h and y are distributed identically in both cases, while z in CGen is random in G subject to the constraint $\mathsf{DL}_g(h) \neq \mathsf{DL}_y(z)$, and in tdCGen it is a random generator of G. It remains to argue that for every $g \in G$, $y^* \in G/\{1\}$, every K output by $\mathsf{tdCGen}(g, y^*)$, and every $\beta \in \mathbb{Z}_q$, triple (c, d, α) in distribution (14) is distributed identically to triple $(\tilde{c}, \tilde{d}, \alpha)$ in distribution (15). First note that commitment c is a deterministic function of d and m, and hence, for every g, y^*, β, it's a deterministic function of (d, α) because $m = g^\alpha (y^*)^\beta$. The fake decommitment $\tilde{c}$ is determined by the same function of $(\tilde{d}, \alpha)$. Therefore we only need to argue that (d, α) and $(\tilde{d}, \alpha)$ part of these two distributions are identically distributed. First note that $(d, \alpha) = (r_1, r_2, \alpha)$ in the first distribution is uniform in $(\mathbb{Z}_q)^3$. We therefore need to argue that the same is true about $(\tilde{d}, \alpha) = (r_1, r_2, \alpha)$ in the second distribution. The reason this holds is that for every $g \in G$, $y^* \in G/\{1\}$, every $\gamma, \gamma_1, \gamma_2$ chosen by $\mathsf{tdCGen}(g, y^*)$, and every $\beta \in \mathbb{Z}_q$, algorithm RstEqv assigns a unique triple (r_1, r_2, α) to every triple (a, b, r), i.e. $(r_1, r_2, \alpha) = F(a, b, r)$ where F is a permutation on $(\mathbb{Z}_q)^3$. Since (a, b, r) is chosen uniformly in $(\mathbb{Z}_q)^3$ by tdCom, (r_1, r_2, α) outputted by RstEqv is uniform in $(\mathbb{Z}_q)^3$ as well.

4.2 Aggregatable ZKPK of DL with Straight-Line Simulation

Three Round HVZK PK of DL: Let G be a prime order group of order q and let g be a generator of G. An honest verifier zero knowledge (HVZK) proof of knowledge (PK) of DL of a group element $y \in G/\{1\}$, denoted by (a, e, s) can be performed by the following protocol: The prover picks $k \xleftarrow{r} \mathbb{Z}_q$ and computes the first message $a = g^k$; The verifier picks the challenge $e \xleftarrow{r} \mathbb{Z}_q$ and sends it to the prover; The prover computes the response $s = ex + k$ where $x = \mathsf{DL}_g(y)$; Finally the verifier accepts iff $g^s = a y^e$. For the purpose of subsequent discussion we briefly recall that this proof system is HVZK because for any challenge e a simulator can pick the response s uniformly in $\mathbb{Z}_q$ and compute the first message a as $g^s y^{-e}$, and it is a proof of knowledge because $x = \mathsf{DL}_g(y)$ can be computed from two accepting transcripts (a, e, s) and (a, e', s') where $e' \neq e$ (such two related transcripts can be achieved by rewinding the prover) because if $a = g^s y^{-e} = g^{s'} y^{-e'}$ then $x = \mathsf{DL}_g(y) = (s - s')(e - e')^{-1} \bmod q$.

Three Round Straight-Line Simulatable and Computationally Sound ZKPK of DL in the CRS Model: Using the restricted equivocable commitment scheme like $\mathcal{CS}$, one can compile the HVZK PK of DL described above into a three round straight-line simulatable and computationally sound ZKPK of DL in the CRS model following the technique of Damgard [7]. (Even though the commitment scheme is not fully equivocable, it has enough equivocability to allow straight-line simulation of this particular proof.) Let $\mathcal{C} = (\mathsf{CGen}, \mathsf{Com}, \mathsf{Open}, \mathsf{tdCGen}, \mathsf{tdCom}, \mathsf{RstEqv})$ be a commitment scheme for public parameter $\mathsf{cpar} = g$ over the group G generated by g. Assume $\mathcal{C}$ is (t_B, ϵ_B)-binding and ϵ_E-equivocable for function $f_{\mathsf{cpar}} : \mathbb{Z}_q \times \mathbb{Z}_q \times G/\{1\} \to G$ where $f_{\mathsf{cpar}}(\alpha, \beta, y) = g^\alpha y^\beta$. Let (a, e, s) be a three round HVZK of PK of DL of a group element $y \in G/\{1\}$. The compilation is as follows: The CRS is the instance of the restricted equivocable commitment scheme. The prover computes $(c, d) \leftarrow \mathsf{Com}_K(a)$ and sends commitment c to the verifier. The verifier picks the challenge $e \in \mathbb{Z}_q$ and

1. $\mathsf{Setup}(1^\kappa)$: Let G be a multiplicative group of prime order q, where the DL assumption holds with security parameter κ and let g be a generator of G. Run CGen on input g to obtain the commitment key K and set hash functions $\mathcal{G} : G^2 \rightarrow \mathbb{Z}_q$ and $\mathcal{H} : G \times \{0,1\}^* \times \{0,1\}^* \rightarrow \mathbb{Z}_q$. The public parameter is $\mathsf{par} = (g, G, q, K)$.
2. $\mathsf{KGen}(\mathsf{par})$: Player P_i picks his (sk_i, pk_i, π_i) tuple as follows:

   Pick $x_i \overset{r}{\leftarrow} \mathbb{Z}_q$, compute $y_i \leftarrow g^{x_i}$ and set $pk_i \leftarrow y_i$, $sk_i \leftarrow x_i$;
   Construct a "proof of possession" of x_i which is a NIZK proof of knowledge of $x_i = \mathsf{DL}_g(y_i)$:
   Pick $k \overset{r}{\leftarrow} \mathbb{Z}_q$, set $e \leftarrow \mathcal{G}(y_i, g^k)$, $s \leftarrow k + ex_i \pmod q$ and $\pi_i \leftarrow (s, e)$;
3. $\mathsf{KVrfy}(\mathsf{par}, pk, \pi)$: Let $pk = y$ and $\pi = (s, e)$; If $e = \mathcal{G}(y, g^s y^{-e})$ then accept otherwise reject.
4. Protocol MSign: Let $\mathcal{P}$ be the set of players that participate in the protocol. (Each player can determine the set $\mathcal{P}$ after the first step of MSign.) Player P_i on inputs (par, m, sk_i), performs the following steps:

   4.1 Pick $k_i \overset{r}{\leftarrow} \mathbb{Z}_q$ and compute $A_i \leftarrow g^{k_i}$ and $(c_i, d_i) \leftarrow \mathsf{Com}_K(A_i)$ and broadcast (y_i, c_i);
   4.2 Upon receiving (y_j, c_j) for all $P_j \in \mathcal{P}$, Set $y \leftarrow \prod_{P_j \in \mathcal{P}} y_j$, $c \leftarrow \bigotimes_{P_j \in \mathcal{P}} c_j$ and $e \leftarrow \mathcal{H}(y, c, m)$;
   Compute $s_i \leftarrow ex_i + k_i \pmod q$ and broadcast (s_i, d_i);
   4.3 Output multisignature $\sigma = (s, e, c, d)$, where $s = \sum_{P_j \in \mathcal{P}} s_j$, $d = \bigoplus_{P_j \in \mathcal{P}} d_j$.
5. $\mathsf{Vrfy}(\mathsf{par}, m, \{pk_1, pk_2, ..., pk_n\}, \sigma)$:

   Parse σ as (e, s, c, d) and each pk_i as y_i and compute $y \leftarrow \prod_{i=1}^n y_i$.
   If $(e = \mathcal{H}(y, c, m) \wedge \mathsf{Open}_K(c, d, g^s y^{-e}) = 1)$ then accept otherwise reject.

Figure 2: $\mathcal{MS}1$, a multisignature scheme in key verification model

sends it back to the prover. The prover responds with s accompanied by a and the decommitment d. The verifier accepts iff $g^s = ay^e$ *and* $\mathsf{Open}_K(c, d, a) = 1$. This proof system is straight-line simulatable: The simulator runs $\mathsf{tdCGen}(\mathsf{cpar})$ to obtain K and the trapdoor td. It then computes $(c, st) \leftarrow \mathsf{tdCom}_K(td)$ and sends c to the verifier. Upon receiving the challenge, e, the simulator uses the *restricted* equivocability property of the commitment scheme to open the fake commitment c to a first message a such that the corresponding (a, e, s) be an accepting conversation of three round HVZK PK of $\mathsf{DL}_g(y)$ by computing $(d, s) \leftarrow \mathsf{RstEqv}_K(td, st, e)$. The simulator sends to the verifier $a = g^s y^{-e}$, d and s. Since the commitment scheme is ϵ_E-equivocable for function $f_{\mathsf{cpar}}(\alpha, \beta, y) = g^\alpha y^\beta$, thus the view of the verifier communicating with the simulator and the view of the verifier in the real protocol is at most ϵ_E apart. This proof system is also computationally binding based on DL assumption. Namely for any cheating prover P^*, there exist an extractor such that given oracle access to P^*, extracts $x = \mathsf{DL}(y)$ with a probability at least $1 - \epsilon_B$. The extractor runs the prover to receive an accepting conversation $(c, e, (a, s, d))$. It then rewinds P^* to the beginning of the second round and sends her a different challenge $e' \neq e$ to obtain another accepting conversation $(c, e', (a', s', d'))$. We have $\mathsf{Open}_K(c, d, a) = 1$ and $\mathsf{Open}_K(c, d', a') = 1$. If $a \neq a'$ then this is an attack against the binding property of the commitment scheme and if $a = a'$, since $g^s = ay^e$ and $g^{s'} = a'y^{e'}$, the extractor can extract the witness by setting $x = \mathsf{DL}_g(y) = (s - s')(e - e')^{-1}$.

5. DL-BASED MULTISIGNATURE IN THE KEY VERIFICATION MODEL

We show a two-round multisignature scheme $\mathcal{MS}1$ (figure 2) secure under the Discrete Logarithm assumption in the Key Verification model. The $\mathcal{MS}1$ scheme relies on a commitment scheme $\mathcal{C} = (\mathsf{CGen}, \mathsf{Com}, \mathsf{Open}, \mathsf{tdCGen}, \mathsf{tdCom}, \mathsf{RstEqv})$ which is exactly like scheme $\mathcal{CS}$ of section 4.1, i.e. multiplicatively homomorphic on message space G and equivocable for function $f_g(\alpha, \beta, y) = g^\alpha y^\beta$. When instantiated with $\mathcal{CS}$, the scheme $\mathcal{MS}1$ requires just three exponentiation per party for signing and two for verification, and it is secure under the DL assumption, with reduction efficiency matching those for standard DL-based signatures. The length of the resulting multisignature is only $4|q|$ bits, because in commitment scheme $\mathcal{CS}$ the commitment can be omitted since it can be computed from the message and the decommitment. The novelty of scheme $\mathcal{MS}1$ is that it achieves $O(1)$-cost verification in the KV model based on only the DL assumption while using short proofs of key well-formedness, $|\pi_i| = 2|q|$, each taking just 1 exponentiation to verify. In contrast, the combined results of [12] and [9] imply a DL-based KV-model multisignature with $O(1)$ multisignature verification but with significantly more expensive proofs, and the scheme is either three rounds (and hence is less practical) or has worse exact security. The key fact enabling the security proof is the generalized forking lemma of section 3.

THEOREM 3. *If* DL *problem is (t', ϵ')-hard in group G and $\mathcal{C}$ is a commitment scheme parameterized with g that is (t_B, ϵ_B)-binding, ϵ_E-equivocable for function $f_g : \mathbb{Z}_q \times \mathbb{Z}_q \times G/\{1\} \rightarrow G$ s.t. $f_g(\alpha, \beta, y) = g^\alpha y^\beta$, and multiplicatively homomorphic on group G, then multisignature scheme $\mathcal{MS}1$ instantiated with $\mathcal{C}$ is $(t, \epsilon, n, q_s, q_h)$-secure in the random oracle model where*

$$\begin{aligned} \epsilon &\leq 8(\epsilon' + \epsilon_B) + 9\epsilon_E \\ t &\geq \frac{\epsilon}{8n^2 q_h \ln(8n/\epsilon)} \min(t', t_B) - q_s t_{sign} \end{aligned}$$

and t_{sign} is the time required for signing by each party.

PROOF. Let $\mathcal{C}$ = $(\mathsf{CGen}, \mathsf{Com}, \mathsf{Open}, \mathsf{tdCGen}, \mathsf{tdCom}, \mathsf{RstEqv})$ be a commitment scheme for public parameters $\mathsf{cpar} = g$ and the message space $\mathcal{M}$ equal to G generated by g. Assume $\mathcal{C}$ is multiplicatively homomorphic, (t_B, ϵ_B)-binding and ϵ_E-equivocable for function $f_{\mathsf{cpar}} : \mathbb{Z}_q \times \mathbb{Z}_q \times G/\{1\} \rightarrow G$ where $f_{\mathsf{cpar}}(\alpha, \beta, y) = g^\alpha y^\beta$. Given a $(t, q_s, q_h, n, \epsilon)$-forger $\mathcal{F}$, consider two simulators $\mathcal{B}_0$ and $\mathcal{B}_1$ that simulate the role of the honest player as in the experiment $\mathbf{Exp}^{\mathsf{uu-cma}}_{\mathsf{MS}}$ interacting with the forger $\mathcal{F}$. $\mathcal{B}_0$ takes as an

```
**Init**:
$(td, K) \leftarrow \mathsf{tdCGen}(g, y_1)$;
$\mathsf{par} \leftarrow (g, \mathsf{G}, \mathsf{H}, K)$; $pk_1 \leftarrow y_1$;
Construct a "proof of possession" of $x_1$ which is a
*simulated* NIZK proof of knowledge of $\mathsf{DL}_g(y_1)$:
    $(e, s) \xleftarrow{r} \mathbb{Z}_q^2$; $\mathsf{G}[(y_1, g^s y_1^{-e})] \leftarrow e$; $\pi_1 \leftarrow (s, e)$;
Execute $\mathcal{F}$ on input $(\mathsf{par}, pk_1, \pi_1)$;

**SimMSign**$(m)$:
  1. $(\tilde{c}, st) \leftarrow \mathsf{tdCom}_K(td)$; $c_1 \leftarrow \tilde{c}$; Send $(y_1, c_1)$ to $\mathcal{F}$;
  2. Upon receiving $(y_j, c_j)$ for all $P_j \in \mathcal{P}$, do
     $c \leftarrow \bigotimes_{P_j \in \mathcal{P}} c_j$; $y \leftarrow \prod_{P_j \in \mathcal{P}} y_j$; $e \leftarrow \mathcal{H}(y, c, m)$;
     $(\tilde{d}, \alpha) \leftarrow \mathsf{RstEqv}_K(td, st, e)$; $s_1 \leftarrow \alpha$; $d_1 \leftarrow \tilde{d}$;
     Send $(s_1, d_1)$ to $\mathcal{F}$
  3. Upon receiving $(s_j, d_j)$ for all $P_j \in \mathcal{P}$, do
     $d \leftarrow \bigoplus_{P_j \in \mathcal{P}} d_j$; $s \leftarrow \sum_{P_j \in \mathcal{P}} s_j$;
     Return $\sigma = (e, s, c, d)$;
```

```
**SimHash**:

$\mathsf{HashQuery}_{\mathcal{G}}(y, u)$: If $(y, u)$ is an $i^{\text{th}}$ distinct query of $\mathcal{F}$ to $\mathcal{H}$
or $\mathcal{G}$, then set $\mathsf{G}[(y, u)] \leftarrow e_i$;
Return $\mathsf{G}[(y, u)]$;

$\mathsf{HashQuery}_{\mathcal{H}}(y, c, m)$: If $(y, c, m)$ is an $i^{\text{th}}$ distinct query of
$\mathcal{F}$ to $\mathcal{H}$ or $\mathcal{G}$, then set $\mathsf{H}[(y, c, m)] \leftarrow e_i$;
Return $\mathsf{H}[(y, c, m)]$;

**Finalize**:
Upon receiving a valid forgery $(m, \sigma, \{(pk_i, \pi_i)\}_{i=2..n})$ from
$\mathcal{F}$, parse $\sigma = (s, e, c, d)$ and $pk_i = y_i$ and $\pi_i = (s_i, e_i)$ for
$i = 2, ..., n$; Compute $y = \prod_{i=1}^{n} y_i$ and query $\mathcal{H}$ on $(y, c, m)$
and $\mathcal{G}$ on $(y_i, g^{s_i} y^{-e_i})$ for $i = 2..n$.
Return $(J, \{\phi_j\}_{j \in J})$ where $J = \{j_0, j_2, j_3 ..., j_n\}$ and $j_0$ is
the index of $e$ in the hash table $\mathsf{H}$ and $\phi_{j_0} = (m, \sigma)$ and for
$i = 2..n$, $j_i$'s are the indices of $e_i$'s in the hash table $\mathsf{G}$ and
$\phi_{j_i} = (y_i, \pi_i)$.
```

Figure 3: Procedures used in the simulation of multisignature scheme $\mathcal{MS}1$

input a set $\{e_1, ..., e_{q_h}\}$ where e_i's are in $\mathbb{Z}_q$ and runs Setup procedure to obtain par and follows the real protocol (*i.e.* procedures KGen and MSign) on behalf of the honest player. Additionally, $\mathcal{B}_0$ answers the forger's hash queries and performs an extra finalization process by following procedures $\mathsf{SimHash}$ and $\mathsf{Finalize}$ in figure 3. The simulator $\mathcal{B}_1$, on the other hand, takes as an input a DL challenge $y_1 \in G/\{1\}$ and a set $\{e_1, ..., e_{q_h}\}$ where e_i's are in $\mathbb{Z}_q$ and follows the Init, $\mathsf{SimMSign}$, $\mathsf{SimHash}$ and $\mathsf{Finalize}$ procedures detailed in figure 3 to perform the initialization, answering to signature queries, answering to hash queries and finalization processes, respectively. Intuitively, the simulator $\mathcal{B}_1$ embeds the DL challenge in the public key of the honest player and utilizes the (*restricted*) equivocability property of the commitment scheme $\mathcal{C}$ to simulate the signature protocol on behalf of the honest player. Both $\mathcal{B}_0$ and $\mathcal{B}_1$, after receiving a valid forgery from $\mathcal{F}$, perform a finalization phase in which the message-forged-multisignature pair and the public-key -forged-POP pairs are returned together with the set of indices of the hash responses upon which they are based. Namely both $\mathcal{B}_0$ and $\mathcal{B}_1$ return $(J, \{\phi_j\}_{j \in J})$ *s.t.* if we denote $J = \{j_0, j_2, j_3 ..., j_n\}$, $\phi_0 = (m, \sigma)$ and for all $i = 2..n$, $\phi_i = (y_i, \pi_i)$ then the following equations hold:

$$\mathsf{Vrfy}(\mathsf{par}, m, \{y_1, y_2, ..., y_n\}, \sigma) = 1 \tag{17}$$

$$\forall i = 2..n : \mathsf{KVrfy}(\mathsf{par}, y_i, \pi_i) = 1 \tag{18}$$

The simulators $\mathcal{B}_0$ and $\mathcal{B}_1$ set up empty tables G and H to simulate the hash functions $\mathcal{G}$ and $\mathcal{H}$, and use the set $\{e_1, ..., e_{q_h}\}$ to answer the hash queries. (This convention makes it easy to use the generalized forking lemma in relation to $\mathcal{B}_0$ and $\mathcal{B}_1$.)

Consider $GF_{\mathcal{B}_l}$, the forking algorithm associated with simulator $\mathcal{B}_l$ for either $l = 0$ or $l = 1$. The success event of $GF_{\mathcal{B}_l}$ denoted by $E^{\mathcal{B}_l}$ is that the algorithm $GF_{\mathcal{B}_l}$ outputs $X_l = \{e_j, \phi_j\}_{j \in J}^{(l)}$ and $\tilde{X}_l = \{\tilde{e}_j, \tilde{\phi}_j\}_{j \in J}^{(l)}$, and $e_j \neq \tilde{e}_j$ for all $j \in J$, where $J = \{j_0, j_2, j_3, ..., j_n\}$ is the set of indices of the hash responses participating in the forgery produced by the first execution of $\mathcal{B}_l$ as run by $GF_{\mathcal{B}_l}$. Thus according to the $\mathsf{Finalize}$ process in figure 3, $\phi_{j_0} = (m, (s, e_{j_0}, c, d))$, $\tilde{\phi}_{j_0} = (\tilde{m}, (\tilde{s}, \tilde{e}_{j_0}, \tilde{c}, \tilde{d}))$ and for $i = 2..n$, $\phi_{j_i} = (y_i, (s_i, e_{j_i}))$ and $\tilde{\phi}_{j_i} = (\tilde{y}_i, (\tilde{s}_i, \tilde{e}_{j_i}))$. Since for $i = 0$ and every $i = 2..n$, the random coins and the hash responses of the algorithm $\mathcal{B}_l$ previous to j_i^{th} query is the same in the first execution and the execution leading to the addition of $(\tilde{e}_{j_i}, \tilde{\phi}_{j_i})$ to $\tilde{X}_l$, all the computations and communications and in particular the queries submitted to the hash functions $\mathcal{H}$ and $\mathcal{G}$ before j_i^{th} query, must be the same, too. Thus the occurrence of $E^{\mathcal{B}_l}$ implies $y = \tilde{y}$, $c = \tilde{c}$, $m = \tilde{m}$ and for all $i = 2..n$, $y_i = \tilde{y}_i$ and $g^{s_i} y_i^{-e_{j_i}} = g^{\tilde{s}_i} \tilde{y}_i^{-\tilde{e}_{j_i}}$. The success event $E^{\mathcal{B}_l}$ can be partitioned into two cases (1) event $E_1^{\mathcal{B}_l}$ in which $E^{\mathcal{B}_l}$ happens and $g^s y^{-e_{j_0}} = g^{\tilde{s}} \tilde{y}^{-\tilde{e}_{j_0}}$ (2) event $E_2^{\mathcal{B}_l}$ in which $E^{\mathcal{B}_l}$ happens and $g^s y^{-e_{j_0}} \neq g^{\tilde{s}} \tilde{y}^{-\tilde{e}_{j_0}}$. Obviously $E^{\mathcal{B}_l} = E_1^{\mathcal{B}_l} \cup E_2^{\mathcal{B}_l}$ and hence $\Pr[E^{\mathcal{B}_l}] \leq \Pr[E_1^{\mathcal{B}_l}] + \Pr[E_2^{\mathcal{B}_l}]$. On the other hand according to the generalized forking lemma, $E^{\mathcal{B}_l}$ can be lower bounded by $\epsilon_{\mathcal{B}_l}$, the success probability of the simulator $\mathcal{B}_l$:

$$\frac{\epsilon_{\mathcal{B}_l}}{8} \leq \Pr[E^{\mathcal{B}_l}] \leq \Pr[E_1^{\mathcal{B}_l}] + \Pr[E_2^{\mathcal{B}_l}] \tag{19}$$

If e_j's are uniformly distributed in $\mathbb{Z}_q$ then $\mathcal{F}$'s view in interaction with $\mathcal{B}_0$ is identical to the real execution of the protocol. As for $\mathcal{B}_1$, P_1's public key and proof of possession of secret key, (y_1, π_1), is distributed as in the real execution of the protocol. This is because y_1 is uniform in G and π_1 is uniform in $\mathbb{Z}_q^2$. Since $\mathcal{C}$ is ϵ_E-equivocable, by definition, the distributions of the commitment keys in the simulation and the real execution have at most ϵ_E statistical difference and additionally, the distribution of the tuples (c_1, d_1, s_1) generated in each signature instance in the interaction between $\mathcal{F}$ and $\mathcal{B}_1$ is identical to the distribution of the same variables in the real execution. Thus, since our simulation is straight-line, total statistical distance between $\mathcal{F}$'s view in interaction with $\mathcal{B}_1$ and in real execution is at most ϵ_E. This implies in particular that $\epsilon_{\mathcal{B}_0} = \epsilon$, $|\epsilon_{\mathcal{B}_1} - \epsilon| \leq \epsilon_E$ and $|\Pr[E_2^{\mathcal{B}_0}] - \Pr[E_2^{\mathcal{B}_1}]| \leq \epsilon_E$. Thus equation (19) becomes:

$$\frac{\epsilon - \epsilon_E}{8} \leq \Pr[E_1^{\mathcal{B}_1}] + \Pr[E_2^{\mathcal{B}_0}] + \epsilon_E \tag{20}$$

The actual reduction algorithm $\mathcal{R}$, runs both $GF_{\mathcal{B}_0}$ and $GF_{\mathcal{B}_1}$. If $E_1{}^{\mathcal{B}_1}$ happens, then $g^s y^{-e_{j_0}} = g^{\tilde{s}} \tilde{y}^{-\tilde{e}_{j_0}}$ and since $y = \tilde{y}$ and $e_{j_0} \neq \tilde{e}_{j_0}$ thus $\sum_{i=1}^{n} x_i = (s - \tilde{s})(e_{j_0} - \tilde{e}_{j_0})^{-1}$ where $x_i = \mathsf{DL}_g(y_i)$ for $i = 1..n$. On the other hand, since for all $i = 2..n$, $y_i = \tilde{y}_i$, $g^{s_i} y_i^{-e_{j_i}} = g^{\tilde{s}_i} \tilde{y}_i^{-\tilde{e}_{j_i}}$ and $e_{j_i} \neq \tilde{e}_{j_i}$, thus the DL's for all y_i's where $i \in \{2, ..., n\}$ can be computed as $\mathsf{DL}_g(y_i) = (s_i -$

1. $\mathsf{Setup}(1^\kappa)$: Let G be a multiplicative group of prime order q, where the DL assumption holds with security parameter κ and let g be a generator of G. Run CGen on input g to obtain the commitment key K and set the hash function $\mathcal{H} : G \times \{0,1\}^* \times \{0,1\}^* \times \{0,1\}^* \to \mathbb{Z}_q$. The public parameter is $\mathsf{par} = (g, G, q, K)$.
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 $y_i \leftarrow g^{x_i}$ and set $pk_i \leftarrow y_i$, $sk_i \leftarrow x_i$ and π_i to the empty string.
3. $\mathsf{KVrfy}(\mathsf{par}, pk, \pi)$: Since the scheme is in the plain model, KVrfy just returns true.
4. Protocol MSign: Let $\mathcal{P}$ be the set of players that participate in the protocol. (Each player can determine the set $\mathcal{P}$ after the first step of MSign.) Player P_i on inputs (par, m, sk_i), performs the following steps:

   4.1 Pick $k_i \xleftarrow{r} \mathbb{Z}_q$ and compute $A_i \leftarrow g^{k_i}$ and $(c_i, d_i) \leftarrow \mathsf{Com}_K(A_i)$ and broadcast (y_i, c_i);

   4.2 Upon receiving (y_j, c_j) for all $P_j \in \mathcal{P}$, Set $pkSet \leftarrow \{y_j\}_{P_j \in \mathcal{P}}$, $c \leftarrow \bigotimes_{P_j \in \mathcal{P}} c_j$ and $e_i \leftarrow \mathcal{H}(y_i, c, pkSet, m)$;
   Compute $s_i \leftarrow e_i x_i + k_i \pmod q$ and broadcast (s_i, A_i, d_i);

   4.3 Output multisignature $\sigma = (s, A, c, d)$, where $s = \sum_{P_j \in \mathcal{P}} s_j$, $d = \bigoplus_{P_j \in \mathcal{P}} d_j$ and $A = \prod_{P_j \in \mathcal{P}} A_j$.
5. $\mathsf{Vrfy}(\mathsf{par}, m, \{pk_1, pk_2, ..., pk_n\}, \sigma)$:

   Parse σ as (s, A, c, d) and each pk_i as y_i. For $i = 1, 2, ..., n$, set $e_i \leftarrow \mathcal{H}(y_i, c, \{y_1, y_2, ..., y_n\}, m)$.
   If $(g^s = A \prod_{i=1}^n y_i^{e_i} \wedge \mathsf{Open}_K(c, d, A) = 1)$ then accept otherwise reject.

Figure 4: $\mathcal{MS}2$, a multisignature scheme in the plain model

$\tilde{s_i})(e_{j_i} - \tilde{e}_{j_i})^{-1}$. Thus $\mathcal{R}$ can compute $\mathsf{DL}_g(y_1)$ by setting

$$\mathsf{DL}_g(y_1) = \frac{s - \tilde{s}}{e_{j_0} - \tilde{e}_{j_0}} - \sum_{i=2}^{n} \frac{s_i - \tilde{s}_i}{e_{j_i} - \tilde{e}_{j_i}}$$

If ${E_2}^{\mathcal{B}_0}$ happens, then $\mathcal{R}$ immediately translates it into an attack against the binding property of the commitment scheme $\mathcal{C}$ by outputting $(c, d, \tilde{d}, g^s y^{-e_{j_0}}, g^{\tilde{s}} y^{-\tilde{e}_{j_0}})$. To see this note that as argued before, $y = \tilde{y}$, $c = \tilde{c}$ and since ${E_2}^{\mathcal{B}_0}$ occurred, thus $g^s y^{-e_{j_0}} \neq g^{\tilde{s}} y^{-\tilde{e}_{j_0}}$ and due to validity of the forgeries we have $\mathsf{Open}_K(c, d, g^s y^{-e_{j_0}}) = \mathsf{Open}_K(c, \tilde{d}, g^{\tilde{s}} y^{-\tilde{e}_{j_0}}) = 1$. Moreover the commitment key K is output by CGen in the execution of $\mathcal{B}_0$.

Thus $\Pr[E_1^{\mathcal{B}_1}] \leq \epsilon'$ and $\Pr[E_2^{\mathcal{B}_0}] \leq \epsilon_B$ and hence equation (20) becomes $(\epsilon - \epsilon_E)/8 \leq \epsilon' + \epsilon_B + \epsilon_E$, which implies:

$$\epsilon \leq 8(\epsilon' + \epsilon_B) + 9\epsilon_E$$

The running time t_R of the reduction algorithm $\mathcal{R}$ is equal to $(8n^2 q_h/\epsilon)\ln(8n/\epsilon)$ times the maximum of running time of the algorithms $\mathcal{B}_0$ and $\mathcal{B}_1$. But the running time of $\mathcal{B}_0$ and $\mathcal{B}_1$ is dominated by the running time of the forger $\mathcal{F}$ plus the time spent by the simulators to answer the hash and signing queries. Thus $t_R \leq (8n^2 q_h/\epsilon)\ln(8n/\epsilon)(t + q_s t_{sign})$ where t_{sign} is the maximum time spent by the simulators $\mathcal{B}_0$ and $\mathcal{B}_1$ to answer one signature query. On the other hand since $\mathcal{R}$ either answers the DL challenge or outputs an attack against the binding property of the commitment $\mathcal{C}$, it must be true that $\min(t', t_B) \leq t_R$. Thus:

$$t \geq \frac{\epsilon}{8n^2 q_h \ln(8n/\epsilon)} \min(t', t_B) - q_s t_{sign}$$

$\square$

The corollary below follows by setting $\epsilon_E = 2/q$, $t_B = t'$ and $\epsilon_B = \epsilon'$. Note also that if $\mathcal{MS}1$ is instantiated with the commitment scheme $\mathcal{CS}$ then the signing time consists of one single-exponentiation and two double-exponentiations and that the time of one double-exponentiation is 1.2 times the time of a group exponentiation.

COROLLARY 4. *If* DL *problem is (t', ϵ')-hard in group G with prime order q and the $\mathcal{MS}1$ protocol, described in figure 2 is instantiated with the commitment scheme $\mathcal{CS}$ described in subsection 4.1, then the resulting multisignature scheme is $(t, \epsilon, n, q_s, q_h)$-secure in random oracle model where*

$$\begin{aligned} \epsilon &\leq 16\epsilon' + 18/q \\ t &\geq \frac{\epsilon}{8n^2 q_h \ln(8n/\epsilon)} t' - 3.4 q_s t_{exp} \end{aligned}$$

and t_{exp} is the time of one exponentiation in G.

6. DL-BASED MULTISIGNATURE IN THE PLAIN MODEL

We show a two-round multisignature scheme $\mathcal{MS}2$ (figure 4) secure under the DL assumption in the *Plain Public Key* model. The $\mathcal{MS}2$ scheme relies on a commitment scheme $\mathcal{C} = (\mathsf{CGen}, \mathsf{Com}, \mathsf{Open}, \mathsf{tdCGen}, \mathsf{tdCom}, \mathsf{RstEqv})$ with the same properties as were required by the $\mathcal{MS}1$ scheme of section 4.2, i.e. a commitment scheme which is multiplicatively homomorphic and equivocable for function $f_g(\alpha, \beta, y) = g^\alpha y^\beta$. When instantiated with the commitment scheme $\mathcal{CS}$ of section 4, multisignature $\mathcal{MS}2$ is secure under the DL assumption in the plain model, has fast signing procedure requiring only three exponentiations per player, and the resulting multisignature takes $4|q|$ bits. The multisignature verification takes $O(n)$ exponentiations, matching the efficiency of the best previously known DL-based multisignature in the plain model of [3]. However, the advantage of this scheme over the scheme of [3] is in reduction of rounds from three to two. The exact security we show for this scheme matches that of [3], and indeed matches the exact security bounds shown for standard DL-based signatures.

THEOREM 5. *If* DL *problem is (t', ϵ')-hard in group G and there exists a commitment scheme parameterized with g that is (t_B, ϵ_B)-binding, ϵ_E-equivocable for function $f_g : \mathbb{Z}_q \times \mathbb{Z}_q \times G/\{1\} \to G$ where $f_g(\alpha, \beta, y) = g^\alpha y^\beta$, and multiplicatively homomorphic on group G, then multisignature scheme in $\mathcal{MS}2$, described in figure 4 is $(t, \epsilon, n, q_s, q_h)$-secure in random oracle model*

```
Init:

$(td, K) \leftarrow \mathsf{tdCGen}(g, y_1)$; $ctr \leftarrow 1$;
$\mathsf{par} \leftarrow (g, \mathsf{H}, K)$; $pk_1 \leftarrow y_1$;
Execute $\mathcal{F}$ on input $(\mathsf{par}, pk_1, )$;

SimMSign$(m)$:

  1. $(\tilde{c}, st) \leftarrow \mathsf{tdCom}_K(td)$; $c_1 \leftarrow \tilde{c}$;
     Send $(c_1, y_1)$ to $\mathcal{F}$;
  2. Upon receiving $(c_j, y_j)$ for all $P_j \in \mathcal{P}$, do
     $c \leftarrow \bigotimes_{P_j \in \mathcal{P}} c_j$; $pkSet \leftarrow \{y_i\}_{P_i \in \mathcal{P}}$;
     $e_1 \leftarrow \mathcal{H}(y_1, c, pkSet, m)$;
     $(\tilde{d}, \alpha) \leftarrow \mathsf{RstEqv}_K(td, st, e_1)$;
     $s_1 \leftarrow \alpha$; $d_1 \leftarrow \tilde{d}$; $\mathcal{A}_1 \leftarrow g^{s_1} y_1^{-e_1}$;
     Send $(s_1, d_1, A_1)$ to $\mathcal{F}$
  3. Upon receiving $(s_j, d_j, A_j)$ for all $P_j \in \mathcal{P}$, do
     $d \leftarrow \bigoplus_{P_j \in \mathcal{P}} d_j$; $s \leftarrow \sum_{P_j \in \mathcal{P}} s_j$; $A \leftarrow \prod_{P_j \in \mathcal{P}} A_j$;
     return $\sigma = (s, A, c, d)$;

SimHash:

$\mathsf{HashQuery}_{\mathcal{H}}(y, c, pkSet, m)$:
If $\mathsf{H}[y, (c, pkSet, m)]$ is undefined, then
    If $(y, y_1 \in pkSet)$ then
        $ctr \leftarrow ctr + 1$;
        For all $y_i \in pkset$ s.t. $y_i \neq y_1$ do
            $\mathsf{H}[y_i, (c, pkSet, m)] \stackrel{r}{\leftarrow} \mathbb{Z}_q$;
        $\mathsf{H}[y_1, (c, pkSet, m)] \leftarrow e_{ctr}$;
    else
        $\mathsf{H}[y, (c, pkSet, m)] \stackrel{r}{\leftarrow} \mathbb{Z}_q$;
return $\mathsf{H}[y, (c, pkSet, m)]$;

Finalize:

Upon receiving a valid forgery $(m, \sigma, \{(pk_i, \pi_i)\}_{i=2..n}$ from
$\mathcal{F}$, parse $\sigma = (s, A, c, d)$ and $pk_i = y_i$ for $i = 2..n$; Let
$pkSet = \{y_i\}_{i=1..n}$. Query $\mathcal{H}$ on $(y_1, c, pkSet, m)$;
Return $(\{j_0\}, \{\phi_{j_0}\})$ where $j_0$ is the index of the hash re-
sponse in $\mathsf{H}[y_1, (c, pkSet, m)]$ and $\phi_{j_0} = (m, \sigma)$.
```

Figure 5: Procedures used in the simulation of multisignature scheme $\mathcal{MS}2$

where

$$\begin{aligned} \epsilon &\leq 8(\epsilon' + \epsilon_B) + 9\epsilon_E \\ t &\geq \frac{\epsilon}{8q_h \ln(8/\epsilon)} \min(t', t_B) - q_s t_{sign} \end{aligned}$$

and t_{sign} is the time required for signing by each party.

The proof is similar to the proof of theorem 3 and relies on the simulator depicted in figure 5. However it uses an interesting technique to respond to the hash functions that enables the use of the generalized forking lemma to extract the DL of the challenge in plain model. More precisely, in order to use the generalized forking lemma to extract $\mathsf{DL}_g(y_1)$ from several executions of the forger within the general forking algorithm, we need firstly that the public key set output as part of the forgery should be the same in these executions and secondly, certain random oracle responses need to be the same in these executions of the forger even though the corresponding queries may not occur until after the fork. To address the first issue, the set of public keys, $pkSet$, is also included in the hash query to $\mathcal{H}$ and to address the second issue, we assign the responses to all queries of the form $(y, c, pkSet, m)$ where $y \in pkSet$ when the first query of that type comes. To have a better idea about how our algorithm for answering the hash queries works imagine simulating the hash function as a table whose rows are indexed by $(c, pkSet, m)$ and whose columns are indexed by y. The hash response to query $(y, c, pkSet, m)$ is $\mathsf{H}[(y, (c, pkSet, m))]$. To answer query $(y, c, pkSet, m)$, if $y, y_1 \in pkSet$ then for all $y_i \in pkSet$ we assign values to entries indexed by $(y_i, (c, pkSet, m))$. If $y \notin pkSet$ or $y_1 \notin pkSet$, then forgery cannot be built and we answer the query with a random value. All the entries indexed by $(y_1, c, pkSet, m)$ where $y_1 \in pkSet$ are assigned from the set $\{e_1, ..., e_{q_h}\}$ in answering to hash queries so that we can use the generalized forking lemma as formulated in section 3.

Acknowledgement Ali Bagherzandi and Stanislaw Jarecki were supported by NSF CyberTrust Grant 0430622. Jung Hee Cheon was supported in part by the SRC Program of KOSEF (R11-2007-035-01002-0).

7. REFERENCES

[1] A. Bagherzandi and S. Jarecki. Multisignatures using proofs of secret key possession, as secure as the Diffie-Hellman problem. In *SCN'08*.

[2] M. Bellare, C. Namprempre, and G. Neven. Unrestricted aggregate signatures. Cryptology ePrint Archive, 2006/285.

[3] M. Bellare and G. Neven. Mult-signatures in the plain public-key model and a general forking lemma. In *ACM CCS'06*.

[4] D. Boneh, C. Gentry, B. Lynn, and H. Shacham. Aggregate and verifiably encrypted signature from bilinear maps. In *Eurocrypt'03*.

[5] D. Boneh, B. Lynn, and H. Shacham. Short signatures from Weil pairing. *J. Cryptology*, 17(4):297–319, 2004.

[6] C. Castelluccia, S. Jarecki, J. Kim, and G. Tsudik. Secure acknowledgment aggregation and multisignatures with limited robustness. *Computer Networks*, 50(10):1639–1652, 2006.

[7] I. Damgård. Efficient concurrent zero-knowledge in the auxiliary string model. In *Eurocrypt'00*.

[8] A. DeSantis and G. Persiano. Zero knowledge proofs of knowledge without interaction. In *FOCS'92*.

[9] M. Fischlin. Communication-efficient non-interactive proofs of knowledge with online extractors. In *Crypto'05*.

[10] J. Groth. Evaluating security of voting schemes in the universal composability framework. Cryptology ePrint Archive, 2002/002.

[11] J. Kim and G. Tsudik. SRDP: Securing route discovery in DSR. In *MobiQuitous*, pages 247–260, 2005.

[12] S. Micali, K. Ohta, and L. Reyzin. Accountable subgroup multisignatures. In *ACM CCS'01*.

[13] T. P. Pedersen. Non-interactive and information theoretic secure verifiable secret sharing. In *Crypto'91*.

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

[15] T. Ristenpart and S. Yilek. The power of proofs of possession: Securing multiparty signatures against rogue-key

attacks. In *Eurocrypt'07*.

[16] C. Schnorr. Efficient identification and signatures for smart cards. In *Crypto'89*.

[17] V. Shoup and R. Gennaro. Securing threshold cryptosystems against chosen ciphertext attack. *J. Cryptology*, 15(2):75–96, 2002.