1-1-2015

An identity-based multi-proxy multi-signature scheme without bilinear pairings and its variants

Maryam Rajabzadeh Asaar
Sharif University of Technology

Mahmoud Salmasizadeh
Sharif University of Technology, salmasi@sharif.edu

Willy Susilo
University of Wollongong, wsusilo@uow.edu.au

An identity-based multi-proxy multi-signature scheme without bilinear pairings and its variants

Abstract

The notions of identity-based multi-proxy signature, proxy multi-signature and multi-proxy multisignature have been proposed to facilitate public key certificate management of these kinds of signatures by merely employing signer's identities in place of the public keys and their certificates. In the literature, most identity-based multi-proxy signature, proxy multi-signature and multi-proxy multi-signature schemes are based on bilinear pairings. Without incorporating bilinear pairings, Tiwari and Padhye proposed an identity-based proxy multi-signature scheme in 2011. Subsequently, an identity-based multi-proxy multi-signature scheme was proposed byTiwari et al. in 2012. First, we review identity-based (multi)-proxy multi-signature schemes without bilinear pairings and show that unfortunately, they are insecure in their security models. Secondly, we propose an identity-based multi-proxy multi-signature scheme without bilinear pairings, where identity-based multi-proxy signature and proxy multi-signature schemes are its special cases. Then, we prove that they are secure under Rivest, Shamir and Adleman (RSA) assumption in the random oracle model by presenting a new Forking Lemma. The proposal and its special cases are the first identity-based multi-proxy signature, proxy multi-signature and multi-proxy multi-signature from RSA assumption.

Keywords

variants, bilinear, its, multi, without, pairings, proxy, scheme, signature, identity

Disciplines

Engineering | Science and Technology Studies

Publication Details

An Identity-based Multi-Proxy Multi-Signature Scheme without Bilinear Pairings and its Variants [*]

Maryam Rajabzadeh Asaar[1], Mahmoud Salmasizadeh[2], and Willy Susilo[**][3]

[1] Department of Electrical Engineering,
[2] Electronics Research Institute (Center),
Sharif University of Technology, Tehran, Iran.
[3] Centre for Computer and Information Security Research,
University of Wollongong, Australia.
asaar@ee.sharif.ir, salmasi@sharif.edu, wsusilo@uow.edu.au

Abstract. The notions of identity-based multi-proxy signature, proxy multi-signature and multi-proxy multi-signature have been proposed to facilitate public key certificate management of these kinds of signatures by merely employing signer's identities in place of the public keys and their certificates. In the literature, most identity-based multi-proxy signature, proxy multi-signature and multi-proxy multi-signature schemes are based on bilinear pairings. Without incorporating bilinear pairings, Tiwari and Padhye proposed an identity-based proxy multi-signature scheme in 2011. Subsequently, an identity-based multi-proxy multi-signature scheme was proposed by Tiwari et al. in 2012. First, we review identity-based (multi)-proxy multi-signature schemes without bilinear pairings and show that unfortunately, they are insecure in their security models. Second, we propose an identity-based multi-proxy multi-signature scheme without bilinear pairings, where identity-based multi-proxy signature and proxy multi-signature schemes are its special cases. Then, we prove that they are secure under RSA assumption in the random oracle model by presenting a new Forking Lemma. The proposal and its special cases are *the first* identity-based multi-proxy signature, proxy multi-signature and multi-proxy multi-signature from RSA assumption.

Keywords: identity-based multi-proxy signature, identity-based proxy multi-signature, identity-based multi-proxy multi-signature, random oracle model, RSA assumption.

1 Introduction

The notion of proxy signatures for the first time was introduced by Mambo et al. [1] in 1996. In a proxy signature scheme, an original signer, Alice, can delegate her signing right for signing messages to another signer, Bob, called the proxy signer. Since the notion of proxy signatures has been introduced, several variants of proxy signatures have been proposed. These include proxy signatures from RSA and integer factorization problem [2–7], identity-based proxy signature schemes based on the bilinear pairings [8–15], designated verifier proxy signatures [16–18], short proxy signatures [19], proxy verifiably encrypted signatures [20], proxy signature schemes without random oracles [21], multi-proxy signatures [22–24], proxy multi-signatures [23], multi-proxy multi-signatures [25, 26], identity-based multi-proxy signatures [13, 27–29], identity-based proxy multi-signatures [28, 30–33] and identity-based multi-proxy multi-signature schemes [28, 34–37]. In this study, we focus on identity-based multi-proxy signature, proxy multi-signature and multi-proxy multi-signature schemes.

In a multi-proxy signature scheme, an original signer can delegate her signing right for signing messages to a group of n-proxy signers, called the proxy agent, such that only cooperation of all proxy signers in the proxy group generates the proxy signatures of roughly the same size as that of standard proxy signatures on behalf of the original signer instead of transmitting n individual proxy signatures. This primitive can be used in a company when the boss of the company is on a business trip and some important documents have to be signed. Hence, the boss delegates her signing capability to every department manager of the company such that only all managers jointly can sign important documents on behalf of the boss. Various multi-proxy signatures [22–24] have been proposed till now.

In a proxy multi-signature scheme [23], a proxy signer can generate the signature on behalf of a group of d-original signers, called original group, such that only the cooperation of all the original signers in the original group can authorize the proxy signer to generate a proxy signature of roughly the same size as that of a standard proxy signature on behalf of the original group instead of transmitting d individual proxy signatures. This primitive can be used where a company releases a document which needs to be signed by different managers of that company. These managers can authorize an entity, a proxy signer, to generate proxy multi-signatures on the document when these managers cannot participate in generating signatures.

Similarly, it is possible to extend the two previous signatures to multi-proxy multi-signature in which a group of original signers gives signing delegation to a group of proxy signers to generate signatures on messages on behalf of the original signers. However, a verifier still needs the certified public keys of $n + 1$ signers in a multi-proxy signature, $d + 1$ signers in a proxy multi-signature and $n + d$ signers in a multi-proxy multi-signature to verify the validity of these signatures. If these public keys and their certificates are transmitted with these signatures, it defeats the main purpose of a multi-proxy signature, proxy multi-signature or multi-proxy multi-signature, to save bandwidth. On the other hand, these kinds of schemes in their basic formats require extensive public-key infrastructure for practical use. In order to save bandwidth and provide more flexible management of public keys, lots of identity-based multi-proxy signature schemes [13, 27–29], identity-based proxy multi-signature schemes [28, 30–33] and identity-based multi-proxy multi-signature schemes [28, 34–37] have been proposed. The notion of identity-based cryptography was introduced by Shamir [38], and since the realization of elliptic curve pairings, there has been a huge increase in implementations of identity-based signatures. The majority of identity-based multi-proxy signature, proxy multi-signature and multi-proxy multi-signature schemes proposed have relied on pairings. While extensive research has led to vast improvements in implementation of pairings, their computational cost is necessarily higher than for more traditional public key algorithms which use exponentiation in various groups. Moreover, pairing-based cryptosystems rely on newer computational assumptions in their security analysis. There has been a proliferation of pairing-based assumptions whose difficulty is not widely understood and whose connection to established assumptions, and to each other, remains unknown. Therefore, when designing new identity-based multi-proxy signatures, proxy multi-signatures and multi-proxy multi-signatures it is desirable to diversify the computational assumptions and to use widely accepted assumptions where possible.

Without incorporating bilinear pairings, Tiwari and Padhye proposed an identity-based proxy multi-signature scheme [33] in 2011, and an identity-based multi-proxy multi-signature scheme [37] was proposed by Tiwari et al. in 2012. In this study, first we show that identity-based proxy multi-signature [33] and multi-proxy multi-signature [37] are not secure in their security models. Then, we propose the first identity-based multi-proxy multi-signature scheme from RSA, without bilinear pairings, where identity-based multi-proxy signature and proxy multi-signature schemes are special cases of it. The proposal and its special cases are the sequential aggregation of GQ identity-based signature [39] or GQ identity-based multi-signature with GQ identity-based multi-signature, which is a different version of identity-based multi-signature from RSA [40] proposed by Neven and Bellare in the number of interactions and random oracles. Furthermore, we show that the scheme is secure under one-wayness of the RSA problem in the random oracle model [41] by proposing a Forking Lemma suitable for our construction. We should highlight that the general Forking Lemma [42] cannot be applied directly into our scheme since this scheme is the result of sequential aggregation of two different signatures such that we have two different types of random oracle responses. Hence, we need to consider the probability of happening some random responses before the forking point in the proposed forking lemma, and this is the main difference of our Forking Lemma from previous ones.

The rest of this paper is organized as follows. Section 2 presents the RSA complexity assumption employed as the signature foundation and the model of identity-based multi-proxy signature, proxy multi-signature and multi-proxy multi-signature schemes including their outline and security properties. Review and security analysis of the identity-based proxy multi-signature [33] and the multi-proxy multi-signature [37] without bilinear pairings are given in Section 3. Our proposed scheme and its formal security proof are presented in Section 4. The generalization of our security proof is given in Section 5. Sections 6 and 7 present the concluding remarks and conclusion, respectively.

2 Background

In this section, we review the RSA assumption and then present the outline and the security properties of identity-based multi-proxy signature, proxy multi-signature and multi-proxy multi-signature schemes.

2.1 The RSA assumption

An RSA key generator KG_{rsa} is an algorithm that generates triplets (N, e, d) such that N is the product of two large primes p and q and $ed = 1 \bmod \varphi(N)$, where $\varphi(N) = (p-1)(q-1)$. The advantage of B in breaking the one-wayness of RSA related to KG_{rsa} is defined as

$$Adv_{KG_{rsa}}^{ow-rsa}(B) = \Pr\left[\gamma^e = y \bmod N \mid (N, e, d) \longleftarrow KG_{rsa}; y \longleftarrow \mathbb{Z}_N^*; \gamma \longleftarrow B(N, e, y)\right] \tag{1}$$

We say that B (t', ϵ')-breaks the one-wayness of RSA with respect to KG_{rsa} if it runs in time at most t' and has advantage $Adv_{KG_{rsa}}^{ow-rsa}(B) \geq \epsilon'$. We say that the RSA function associated to KG_{rsa} is (t', ϵ')-one-way if no algorithm B (t', ϵ')-breaks it.

2.2 Outline of identity-based multi-proxy signature, proxy multi-signature and multi-proxy multi-signature schemes

An original signer with identity ID_o and a group of proxy signers with identities $\langle ID_{p_1}, ..., ID_{p_n}\rangle$, a group of original signers with identities $\langle ID_{o_1}, ..., ID_{o_d}\rangle$ and a proxy signer with identity ID_p and a group of original signers with identities $\langle ID_{o_1}, ..., ID_{o_d}\rangle$ and a group of proxy signers with identities $\langle ID_{p_1}, ..., ID_{p_n}\rangle$ are participants of identity-based multi-proxy signature, proxy multi-signature and multi-proxy multi-signature schemes, respectively. Each warrant-based proxy signature scheme consists of ParaGen, KeyExtract, StandardSign (MSign), StandardVer (MVer), DelegationGen, MProxySign (ProxyMSign or MProxyMSign) and MProxyVer (ProxyMVer or MProxyMVer) as follows. We note that 11 is concatenated to each warrant to differentiate ordinary signatures from delegations, and also 01 is concatenated to each message to differentiate proxy signatures from aggregate signatures.

– ParaGen: This algorithm takes as input the system security parameter l and outputs system's parameters $Para$ and the system's master key (msk, mpk), i.e. $(Para, (msk, mpk)) \leftarrow ParaGen(l)$.

– KeyExtract: This algorithm takes as input the system's parameter $Para$, master public key mpk, master secret key msk, and an identity ID_u. Then, it outputs the corresponding secret key x_u, i.e. $x_u \leftarrow KeyExtract(Para, mpk, msk, ID_u)$.

– StandardSign: This algorithm takes as input the system's parameter $Para$, the master public key mpk, the signer's secret key x_u and the message m, then, it outputs the standard signature σ_u, i.e. $\sigma_u \leftarrow StandardSign(Para, mpk, m, x_u)$.

– StandardVer: This algorithm takes as input the system's parameter $Para$, the master public key mpk, the signer's identity ID_u, the message m and the standard signature σ_u, then, it outputs 1 if σ_u is a valid standard signature of the message m under the identity ID_u and outputs 0 otherwise, i.e. $\{0, 1\} \leftarrow StandardVer(Para, mpk, ID_u, m, \sigma_u)$.

– MSign: This interactive protocol is run by a group of signers with identities $\langle ID_{u_1}, ...ID_{u_d}\rangle$ who intend to sign the same message m, each signer takes as input the system's parameter $Para$, the master public key mpk, the signer's secret key, x_{u_j}, $1 \leq j \leq d$, the identity set $\langle ID_{u_1}, ...ID_{u_d}\rangle$ of all participated signers in the protocol, and additional inputs Inp_{co-us} generated by other co-signers, then, it outputs the multi-signature σ_u after a number of interactions, i.e. $\sigma_u \leftarrow MSign(Para, mpk, m, x_{u_j}, \langle ID_{u_1}, ...ID_{u_d}\rangle, Inp_{co-us})$.

– MVer: This algorithm takes as input the system's parameter $Para$, the master public key mpk, the signers' identities $\langle ID_{u_1}, ..., ID_{u_d}\rangle$, the message m and the multi-signature σ_u, then, it outputs 1 if σ_u is a valid multi-signature of the message m under the identities $\langle ID_{u_1}, ..., ID_{u_d}\rangle$ and outputs 0 otherwise, i.e. $\{0, 1\} \leftarrow MVer(Para, mpk, \langle ID_{u_1}, ..., ID_{u_d}\rangle, m, \sigma_u)$.

– DelegationGen: This algorithm employs the StandardSign or MSign algorithm depending on the type of signature scheme to generate a delegation after some interactions between original signer(s) and proxy signer(s). In case of having (multi)-proxy multi-signature, the delegation is a multi-signature generated by original signers in the original group, otherwise, it is a standard signature of an original signer on the warrant. In case of StandardSign algorithm, x_u is the secret key of the original signer, x_o, and the message m is warrant $(w||11)$, while in case of MSign algorithm x_{u_j} is the secret key of an original signer x_{o_j}, for $1 \leq j \leq d$, the message m is warrant $(w||11)$ and additional inputs Inp_{co-os} generated by other co-original signers, where w includes the identity (identities) of proxy signer(s), the type of the delegated information and the period of delegation.

– The three following pairs of interactive protocols (algorithms) and algorithm are MProxySign and MProxyVer for identity-based multi-proxy signature scheme, ProxyMSign and ProxyMVer for proxy multi-signature or MProxyMSign and MProxyMVer for multi-proxy multi-signature scheme, depending on the type of signature scheme.

- MProxySign: This is an interactive protocol in which each proxy signer with identity ID_{p_i}, $1 \leq i \leq n$ takes as input the system's parameter $Para$, the master public key mpk, the proxy signers' identities $\langle ID_{p_1}, ..., ID_{p_n} \rangle$, original signer's identity ID_o, the warrant w, the delegation σ_o, its secret key x_{p_i}, additional inputs Inp_{co-ps} of other proxy signers in the proxy agent and the message m to be signed, then, it outputs the identity-based multi-proxy signature θ on behalf of the original signer after a number of interactions with other proxy signers, i.e. $\theta \leftarrow MProxySign(Para, mpk, ID_o, \langle ID_{p_1}, ..., ID_{p_n} \rangle, w, \sigma_o, x_{p_i}, Inp_{co-ps}, m||01)$.

- MProxyVer: This algorithm takes as input the system's parameter $Para$, the master public key mpk, the original signer's identity ID_o, the proxy signers' identities $\langle ID_{p_1}, ..., ID_{p_n} \rangle$, the warrant w, the signed message m and the multi-proxy signature θ, then, it outputs 1 if θ is a valid identity-based multi-proxy signature of the message m and outputs 0 otherwise, i.e. $\{0, 1\} \leftarrow MProxyVer(Para, mpk, ID_o, \langle ID_{p_1}, ..., ID_{p_n} \rangle, w, m||01, \theta)$.

- ProxyMSign: This algorithm takes as input the system's parameter $Para$, the master public key mpk, original signers' identities $\langle ID_{o_1}, ..., ID_{o_d} \rangle$, proxy signer's identity ID_p, the warrant w, the delegation σ_o, secret key x_p of the proxy signer with identity ID_p and the message m to be signed, then, it outputs the identity-based proxy multi-signature θ on behalf of the original signers, i.e. $\theta \leftarrow ProxyMSign(Para, mpk, \langle ID_{o_1}, ..., ID_{o_d} \rangle, ID_p, w, \sigma_o, x_p, m||01)$.

- ProxyMVer: This algorithm takes as input the system's parameter $Para$, the master public key mpk, the original signers' identities $\langle ID_{o_1}, ..., ID_{o_d} \rangle$, the proxy signer's identity ID_p, the warrant w, the signed message m and the proxy multi-signature θ, then, it outputs 1 if θ is a valid identity-based proxy multi-signature of the message m and outputs 0 otherwise, i.e. $\{0, 1\} \leftarrow ProxyMVer(Para, mpk, \langle ID_{o_1}, ..., ID_{o_d} \rangle, ID_p, w, m||01, \theta)$.

- MProxyMSign: This is an interactive protocol in which each proxy signer with identity ID_{p_i}, $1 \leq i \leq n$ takes as input the system's parameter $Para$, the master public key mpk, original signers' identities $\langle ID_{o_1}, ...ID_{o_d} \rangle$, proxy signers' identities $\langle ID_{p_1}..., ID_{p_n} \rangle$, the warrant w, the delegation σ_o, its secret key x_{p_i}, additional inputs Inp_{co-ps} of other proxy signers in the proxy group and the message m to be signed, then, it outputs the identity-based multi-proxy multi-signature θ on behalf of original signers after a number of interactions with other proxy signers, i.e. $\theta \leftarrow MProxyMSign(Para, mpk, \langle ID_{o_1}, ...ID_{o_d}, ID_{p_1}..., ID_{p_n} \rangle, w, \sigma_o, x_{p_i}, Inp_{co-ps}, m||01)$.

- MProxyMVer: This algorithm takes as input the system's parameter $Para$, the master public key mpk, the original signers' identities $\langle ID_{o_1}, ..., ID_{o_d} \rangle$, the proxy signers' identities $\langle ID_{p_1}, ..., ID_{p_n} \rangle$, the warrant w, the signed message m and the multi-proxy multi-signature θ, then, it outputs 1 if θ is a valid identity-based multi-proxy multi-signature of the message m and outputs 0 otherwise, i.e. $\{0, 1\} \leftarrow MProxyMVer(Para, mpk, \langle ID_{o_1}, ...ID_{o_d}, ID_{p_1}..., ID_{p_n} \rangle, w, m||01, \theta)$.

2.3 Security models of identity-based multi-proxy signature, proxy multi-signature and multi-proxy multi-signature schemes

In a warrant-based identity-based multi-proxy signature, proxy multi-signature and multi-proxy multi-signature, the delegation is the original signer's standard signature (original signers' multi-signature) on the warrant w which contains information regarding the proxy agent such as the proxy agent's identity (identities), the period of validity, the restriction on the class of messages for which the warrant is valid. Therefore, the properties of strong identifiability, strong undeniability, verifiability, and prevention of misuse are satisfied naturally. Therefore, the signature scheme should be secure against existential forgery under an adaptive-chosen-message, an adaptive-chosen warrant and chosen identity attack.

To have the strongest security notion possible and at the same time avoid making security proof unnecessarily complicated, we use single-signer setup, in which each signing oracle simulates the role of only one honest signer, with only one honest signer in the security model as it is a folklore and well-accepted in the literature [13, 29–33, 36, 37]. The adversary A can choose the identities on which it wants to forge a proxy signature and can request the secret keys corresponding to them (corrupted users) except for the honest signer, and also A can make delegation and proxy signature queries on arbitrary warrants and messages under arbitrary identities including only one honest identity.

To achieve existential unforgeability, three types of potential adversaries are considered. Adversaries of type I which only have identities of original and proxy signers, adversaries of type II which have secret keys of all proxy signers in addition to capabilities of adversaries of type I and adversaries of type III which have secret keys of all original signers in addition to identities of original and proxy signers.

Since an identity-based proxy signature scheme secure against type II (or type III) adversaries is also secure against type I adversaries we will henceforth only consider type II and type III adversaries. To have a formal definition for strong unforgeability, the following game between the challenger C and the adversary A is considered to be played [13].

1. Setup: C runs the $ParaGen$ algorithm with a security parameter l to obtain system's parameter $para$ and the master key (mpk, msk), then it sends $(mpk, para)$ to A.

2. The adversary A issues a polynomially bounded number of queries to the following oracles adaptively.
   – KeyExtract queries: A can ask for the secret key corresponding to an identity ID_u, then C returns the private key x_u with running the KeyExtract algorithm.

   – DelegationGen queries: If the scheme is proxy multi-signature or multi-proxy multi-signature, this kind of query is MSign query, otherwise, this kind of query is StandardSign query.

     • StandardSign queries: Adversary A can request the StandardSign algorithm under the identity ID_u on the message m of its choice. Then, C returns $\sigma_u \leftarrow StandardSign(Para, mpk, m, x_u)$ to A. Especially, if $ID_u = ID_o$, A can choose $m = (w\|11)$ to attain a delegation σ_o on a warrant w.

     • MSign queries: Adversary A can request the multi-signature σ_u of m w.r.t. identities $\langle ID_{u_1}, ..., ID_{u_d} \rangle$ to C. In response, C firstly runs the KeyExtract algorithm to obtain the secret key, x_{u_t}, corresponding to the identity of the honest signer, ID_{u_t}. Next, C runs MSign protocol to generate a multi-signature $\sigma_u \leftarrow MSign(Para, mpk, m, x_{u_t}, \langle ID_{u_1}, ...ID_{u_d} \rangle, Inp_{co-us})$ for $1 \leq t \leq d$, and returns σ_u to the adversary A, where Inp_{co-us} is generated by A since it is assumed that other signers are corrupted. Especially, if the multi-set of identities are identities of original signers, A can choose $m = (w\|11)$, and also plays the role of other co-original signers to attain a delegation σ_o on a warrant w under identities $\langle ID_{o_1}, ..., ID_{o_d} \rangle$.

   – In case of multi-proxy signature, proxy multi-signature and multi-proxy multi-signature, we have items (a), (b) and (c), respectively.

     (a) MProxySign queries: Adversary A can request the multi-proxy signature of (w, m) w.r.t. original signer's identity ID_o and proxy signers' identities $\langle ID_{p_1}, ..., ID_{p_n} \rangle$ including one honest proxy

signer, where m and identities of proxy signers are in the warrant w. In response, C firstly runs the KeyExtract algorithm to obtain the secret key, x_{p_t}, corresponding to the identity of the honest proxy signer with identity ID_{p_t}. Next, C receives σ_o from A (A obtains it from DelegationGen oracle or simulates it by itself), and runs MProxySign protocol for the honest proxy signer to generate $\theta \leftarrow MProxySign(Para, mpk, ID_o, \langle ID_{p_1}, ..., ID_{p_n} \rangle, w, \sigma_o, x_{p_t}, Inp_{co-ps}, m||01)$ after a number of interactions, where Inp_{co-ps} is generated by A since it is assumed that other proxy signers are corrupted. Then C returns θ to the adversary A.

(b) ProxyMSign queries: Adversary A can request the proxy multi-signature of (w, m) w.r.t. original signers' identities $\langle ID_{o_1}, ..., ID_{o_d} \rangle$ and honest proxy signer's identity ID_p, where m and the proxy signer's identity are in the warrant w. In response, C firstly runs the KeyExtract algorithm to obtain the secret key, x_p, corresponding to the identity of the proxy signer.
In addition, C receives σ_o from A (A obtains it from DelegationGen oracle or simulates it by itself), then, C generates proxy multi-signature $\theta \leftarrow ProxyMSign(Para, mpk, \langle ID_{o_1}, ..., ID_{o_d} \rangle, ID_p, w, \sigma_o, x_p, m||01)$ and returns it to the adversary A.

(c) MProxyMSign queries: Adversary A can request the multi-proxy multi-signature of (w, m) w.r.t. original signers' identities $\langle ID_{o_1}, ..., ID_{o_d} \rangle$ and proxy signers' identities $\langle ID_{p_1}, ...ID_{p_n} \rangle$ including one honest user, where m and proxy signers' identities are in the warrant w. In response, C firstly runs the KeyExtract algorithm to obtain the secret key, x_{p_t}, corresponding to the identity of the honest proxy signer with identity ID_{p_t}.
In addition, C receives σ_o from A (A obtains it from DelegationGen oracle or simulates it by itself), then, C runs MProxyMSign protocol for the honest proxy signer to generate $\theta \leftarrow MProxyMSign(Para, mpk, \langle ID_{o_1}, ...ID_{o_d}, ID_{p_1}..., ID_{p_n} \rangle, w, \sigma_o, x_{p_t}, Inp_{co-ps}, m||01)$, where Inp_{co-ps} is generated by A since it is assumed that other proxy signers are corrupted. Then, C returns θ to the adversary A.

3. Finally, A outputs a valid identity-based proxy signature (m^*, w^*, θ^*) w.r.t. original signers' identities (original signer's identity) and proxy signers' identities (proxy signer's identity), and wins the game if the following conditions hold.

– In case of having identity-based multi-proxy signature scheme :
   1. For adversaries of type II, we have
      • E_0: Identity of the original signer in the warrant w^* has not been requested to the KeyExtract algorithm.
      • E_1: The warrant w^* has not been requested as one of the DelegationGen queries under the identity of the original signer.
   2. For adversaries of type III, we have
      • E_0: One of the identities of the proxy signers in the warrant w^* has not been requested to the KeyExtract algorithm.
      • E_1: The pair (m^*, w^*) has not been requested as one of the MProxySign queries under the identity set of the proxy signers in the warrant w^*.
– In case of having identity-based proxy multi-signature scheme :
   1. For adversaries of type II, we have
      • E_0: One of the identities of the original signers in the warrant w^* has not been requested to the KeyExtract algorithm.
      • E_1: The warrant w^* has not been requested as one of the DelegationGen queries under the original signer's identity set.
   2. For adversaries of type III, we have
      • E_0: Identity of the proxy signer in the warrant w^* has not been requested to the KeyExtract algorithm.
      • E_1: The pair (m^*, w^*) has not been requested as one of the ProxyMSign queries under the identity of the honest proxy signer in the warrant w^*.
– In case of having identity-based multi-proxy multi-signature scheme:

1. For adversaries of type II, we have
   - E_0: One of identities of the original signers in the warrant w^* has not been requested to the KeyExtract algorithm.
   - E_1: The warrant w^* has not been requested as one of the DelegationGen queries under the original signers' identity set.
2. For adversaries of type III, we have
   - E_0: One of the identities of the proxy signers in the warrant w^* has not been requested to the KeyExtract algorithm.
   - E_1: The pair (m^*, w^*) has not been requested as one of the MProxyMSign queries under the identity set of the proxy signers in the warrant w^*.

The formal definition of existential unforgeability for adversaries of type II is expressed in Definition 1.

Definition 1. An identity-based proxy signature is $(t, q_h, q_H, q_E, q_d, \epsilon)$- existentially unforgeable against adaptive chosen message (chosen warrant) attack and chosen identity attack if there is no adversary which runs in time at most t, (makes at most $q_H + q_h$ queries to hash functions), makes at most q_E KeyExtract queries, q_d DelegationGen queries can win the aforementioned game with probability at least ϵ.

The formal definition of existential unforgeability for adversaries of type III is expressed in Definition 2.

Definition 2. An identity-based proxy signature is $(t, q_h, q_H, q_E, q_s, \epsilon)$- existentially unforgeable against adaptive chosen message (chosen warrant) attack and chosen identity attack if there is no adversary which runs in time at most t, (makes at most $q_H + q_h$ queries to hash functions), makes at most q_E KeyExtract queries and q_s MProxyMSign (MProxySign or ProxyMsign) queries, can win the aforementioned game with probability at least ϵ.

3 Identity-based (multi)-proxy multi-signature schemes without bilinear pairings

In this section, first we review efficient identity-based (multi)-proxy multi-signature schemes [33, 37] without bilinear pairings, then, we show that these schemes are not secure in their security model.

3.1 Overview of Tiwari et al.'s identity-based multi-proxy multi-signature scheme

In 2012, Tiwari et al. [37] proposed an identity-based multi-proxy multi-signature without bilinear pairings to improve the efficiency of these kinds of schemes. This scheme includes a set of d original signers $\langle ID_{o_1}, ..., ID_{o_d} \rangle$, a set of n proxy signers $\langle ID_{p_1}, ..., ID_{p_n} \rangle$ and a clerk. Their scheme consists of following algorithms:

1. Setup: The system parameters are as follows. Let $H_1 : \{0, 1\}^* \times G \to \mathbb{Z}_\alpha^*$ and $H_2 : \{0, 1\}^* \to \mathbb{Z}_i^*$ be random oracles, where p is a l-bit prime determining tuples $F_p, E/F_p, G, P$, where E/F_p denotes an elliptic curve E over a prime finite field F_p defined by equation $y^2 = x^3 + ax + b$ for a and $b \in F_p$ and discriminant $\Delta = 4a^3 + 27b^2 \neq 0$. Let G be a cyclic additive group with order α, and P be the generator of G. The key distribution center chooses $x \in_R \mathbb{Z}_\alpha^*$ and computes the master public key $P_{pub} = xP$. It publishes $mpk = P_{pub}$ as the master public key, and keeps the master secret key $msk = x$ secret. Therefore, public parameters are $Para = \{H_1, H_2, F_p, E/F_p, G, P\}$ and mpk.

2. KeyExtract: On input master secret key $msk = x$ and the user identity ID_u, the key distribution center chooses at random $r_u \in \mathbb{Z}_\alpha^*$, computes $R_u = r_u P$, $h_u = H_1(ID_u, R_u)$ and $x_u = r_u + h_u x$, and sends the user secret key (x_u, R_u) over a secure and authenticated channel to the user with identity ID_u.

3. DelegationGen: Let w be the warrant to be signed by all original signers with identities $\langle ID_{o_1}, ..., ID_{o_d} \rangle$ who want to delegate their signing right to a group of proxy signers with the identities $\langle ID_{p_1}, ..., ID_{p_n} \rangle$, the delegation is $((K_{o_j}, R_{o_j}), (K_{p_i}, R_{p_i}), K, \sigma)$, $1 \leq i \leq n$ and $1 \leq j \leq d$, which is generated as follows.

- For $1 \leq j \leq d$, the original signer with identity ID_{o_j} chooses $k_{o_j} \xleftarrow{\$} \mathbb{Z}_\alpha^*$, computes $K_{o_j} = k_{o_j}P$, and broadcasts K_{o_j} to $d-1$ original signers, n proxy signers and the clerk C. Similarly, for $1 \leq i \leq n$, the proxy signer with identity ID_{p_i} chooses $k_{p_i} \xleftarrow{\$} \mathbb{Z}_\alpha^*$, computes $K_{p_i} = k_{p_i}P$, and broadcasts K_{p_i} to $n-1$ proxy signers, d original signers and the clerk C.
- The clerk C and all signers compute $K = \sum_1^n K_{p_i} + \sum_1^d K_{o_j}$.
- For $1 \leq i \leq n$, the proxy signer with identity ID_{p_i} computes $\sigma_{p_i} = e_{p_i}x_{p_i} + k_{p_i}$, where $e_{p_i} = H_1(w, K_{p_i}, K)$, and broadcast σ_{p_i} to the clerk C. Similarly, for $1 \leq j \leq d$, the original signer with identity ID_{o_j} computes $\sigma_{o_j} = e_{o_j}x_{o_j} + k_{o_j}$, where $e_{o_j} = H_1(w, K_{o_j}, K)$, and broadcast σ_{o_j} to the clerk C.
- The clerk checks if $\sigma_{p_i}P = e_{p_i}[R_{p_i} + h_{p_i}P_{pub}] + K_{p_i}$ for $1 \leq i \leq n$ and if $\sigma_{o_j}P = e_{o_j}[R_{o_j} + h_{o_j}P_{pub}] + K_{o_j}$ for $1 \leq j \leq d$. If all of them are valid, the clerk C computes $\sigma = \sum_1^n \sigma_{p_i} + \sum_1^d \sigma_{o_j}$, and broadcasts σ to all original and proxy signers. Therefore, the delegation is $((K_{o_j}, R_{o_j}), (K_{p_i}, R_{p_i}), K, \sigma)$, $1 \leq i \leq n$ and $1 \leq j \leq d$.

4. MProxyMSign: The proxy agent with identities $\langle ID_{p_1}, ... ID_{p_n} \rangle$ can sign a message m under the warrant w and the delegation $((K_{o_j}, R_{o_j}), (K_{p_i}, R_{p_i}), K, \sigma)$, $1 \leq i \leq n$ and $1 \leq j \leq d$ as follows:

- For $1 \leq i \leq n$, the proxy signer with identity ID_{p_i} chooses $a_i \xleftarrow{\$} \mathbb{Z}_\alpha^*$, computes $N_i = a_iP$, and broadcasts N_i to $n-1$ proxy signers.
- For $1 \leq i \leq n$, the proxy signer with identity ID_{p_i} first computes $N = \sum_1^n N_i$, $h = H_2(m, N, K)$, $s_i = h\sigma + a_i$, and sends (N_i, s_i) to the clerk.
- For $1 \leq i \leq n$, a proxy signer with identity ID_{p_i} sends the delegation $((K_{o_j}, R_{o_j}), (K_{p_i}, R_{p_i}), K, \sigma)$, $1 \leq i \leq n$ and $1 \leq j \leq d$ on the warrant w to the clerk, the clerk first verifies the validity of the delegation by checking if $\sigma P = \sum_1^d e_{o_j}(R_{o_j} + h_{o_j}P_{pub}) + K + \sum_1^n e_{p_i}(R_{p_i} + h_{p_i}P_{pub})$ holds. If so, it continues; otherwise, it rejects the delegation $((K_{o_j}, R_{o_j}), (K_{p_i}, R_{p_i}), K, \sigma)$, $1 \leq i \leq n$ and $1 \leq j \leq d$.
- The clerk C computes $N = \sum_1^n N_i$, then checks if $s_iP = h[\sum_1^d(e_{o_j}(R_{o_j} + h_{o_j}P_{pub})) + K + \sum_1^n(e_{p_i}(R_{p_i} + h_{p_i}P_{pub}))] + N_i$ holds for $1 \leq i \leq n$. When all individual proxy signatures are valid, the multi-proxy multi-signature can be generated as $\theta = ((K_{o_j}, R_{o_j}), (K_{p_i}, R_{p_i}), K, \sigma, N, s)$, $1 \leq i \leq n$ and $1 \leq j \leq d$ by computing $s = \sum_1^n s_i$.

5. MProxyMVer: Given identities $\langle ID_{o_1}, ... ID_{o_d} \rangle$ of original signers and identities $\langle ID_{p_1}, ... ID_{p_n} \rangle$ of proxy signers, a warrant w, a message m, and a signature $\theta = ((K_{o_j}, R_{o_j}), (K_{p_i}, R_{p_i}), K, \sigma, N, s)$, a verifier operates as follows:

- Checks if the message m conforms to the warrant w, otherwise, it stops.
- Checks if n proxy signers with identities $\langle ID_{p_1}, ..., ID_{p_n} \rangle$ are authorized by original signers with identities $\langle ID_{o_1}, ..., ID_{o_d} \rangle$ in the warrant w, otherwise, it stops.
- Accepts the multi-proxy multi-signature if and only if $sP = hn[\sum_1^d(e_{o_j}(R_{o_j} + h_{o_j}P_{pub}) + K_{o_j}) + \sum_1^n(e_{p_i}(R_{p_i} + h_{p_i}P_{pub}) + K_{p_i})] + N$ holds, where $e_{o_j} = H_1(w, K_{o_j}, K)$, $e_{p_i} = H_1(w, K_{p_i}, K)$, $h_{o_j} = H_1(ID_{o_j}, R_{o_j})$, $h_{p_i} = H_1(ID_{p_i}, R_{p_i})$ and $h = H_2(m, N, K)$.

3.2 Security analysis of Tiwari et al.'s identity-based multi-proxy multi-signature scheme

Tiwari et al.'s scheme is forgeable by original signers, the clerk or everyone who has a valid delegation. Note that delegations are public in the scheme. This security drawback is the result of not employing proxy signers' secret keys in MProxyMSign algorithm to generate multi-proxy multi-signatures on different messages in the warrant w.

The scenario of this attack is as follows. Original signers, the clerk or an adversary with having a valid delegation $((K_{o_j}, R_{o_j}), (K_{p_i}, R_{p_i}), K, \sigma)$, $1 \leq i \leq n$ and $1 \leq j \leq d$ can generate valid multi-proxy multi-signatures on arbitrary messages in the warrant w. The adversary first chooses $a' \in_R \mathbb{Z}_\alpha^*$, then, computes $N' = a'P$, $h' = H_2(m', N', K)$ and $s' = a' + nh'\sigma$. The forged signature is $\theta' = ((K_{o_j}, R_{o_j}), (K_{p_i}, R_{p_i}), K, \sigma, N', s')$, $1 \leq i \leq n$ and $1 \leq j \leq d$ under the warrant w on the message m' in the warrant w. Hence, the first two

conditions in the MProxyMVer algorithm are held since the forgery was performed for the arbitrary message m' in the warrant w, and also the valid delegation shows authorization of proxy signers with identities $\langle ID_{p_1}, ..., ID_{p_n}\rangle$ by original signers with identities $\langle ID_{o_1}, ..., ID_{o_d}\rangle$ in the warrant w. Furthermore, the verification equation $s'P = (a' + nh'\sigma)P = N' + nh'[\sum_1^d(e_{o_j}(R_{o_j} + h_{o_j}P_{pub}) + K_{o_j}) + \sum_1^n(e_{p_i}(R_{p_i} + h_{p_i}P_{pub}) + K_{p_i})]$ holds for the the forged signature θ' since it is generated according to the scheme, where $e_{o_j} = H_1(w, K_{o_j}, K)$, $e_{p_i} = H_1(w, K_{p_i}, K)$, $h_{o_j} = H_1(ID_{o_j}, R_{o_j})$, $h_{p_i} = H_1(ID_{p_i}, R_{p_i})$ and $h' = H_2(m', N', K)$. Note that in the verification equation, we have $\sigma P = \sum_1^d(e_{o_j}(R_{o_j} + h_{o_j}P_{pub}) + K_{o_j}) + \sum_1^n(e_{p_i}(R_{p_i} + h_{p_i}P_{pub}) + K_{p_i})$ since the delegation is valid and was obtained by the adversary or the clerk, or was generated by the original signers.

If we consider a condition in MProxyMVer algorithm such that each verifier checks if N equals the summation of N_i to prevent from the aforementioned attack, where N_i indicates the identity of each proxy signer through an authenticated broadcast primitive, we cannot make the scheme resistant against this attack.

With this condition in the verification of each multi-proxy multi-signature, the clerk can still forge valid signatures on different messages in the warrant after generation of a valid multi-proxy multi-signature. Since for this forgery attack the clerk needs some components (s_i, N), $1 \le i \le n$ of a valid multi-proxy multi-signature in addition to the delegation. After creation of a valid multi-proxy multi-signature, the clerk knows the tuple (m, N, σ, K, s_i). Next, the clerk computes $a_i = s_i - \sigma H_2(m, N, K)$ for $1 \le i \le n$. Then, it can generate valid multi-proxy multi-signatures on arbitrary messages in the warrant w with computing $a = \sum_i a_i$, $N = aP$, $h' = H_2(m', N, K)$ and $s' = a + nh'\sigma$. Hence, the forged signature is $\theta' = ((K_{o_j}, R_{o_j}), (K_{p_i}, R_{p_i}), K, \sigma, N, s')$. Sine the delegation is valid, we have $\sigma P = \sum_1^d(e_{o_j}(R_{o_j} + h_{o_j}P_{pub}) + K_{o_j}) + \sum_1^n(e_{p_i}(R_{p_i} + h_{p_i}P_{pub}) + K_{p_i})$ and consequently, the verification equation $s'P = (a + nh'\sigma)P = N + nh'[\sum_1^d(e_{o_j}(R_{o_j} + h_{o_j}P_{pub}) + K_{o_j}) + \sum_1^n(e_{p_i}(R_{p_i} + h_{p_i}P_{pub}) + K_{p_i})]$ holds for the forged signature θ', where $e_{o_j} = H_1(w, K_{o_j}, K)$, $e_{p_i} = H_1(w, K_{p_i}, K)$, $h_{o_j} = H_1(ID_{o_j}, R_{o_j})$, $h_{p_i} = H_1(ID_{p_i}, R_{p_i})$ and $h' = H_2(m', N, K)$. Since previous N_i of a valid signature are used by the clerk to generate a forgery, the new condition is held. Therefore, with the condition of checking if N in the signature equals the summation of N_i, where N_i indicates the identity of each proxy signer through an authenticated broadcast primitive, the proposed scheme cannot be secure against the mentioned forgery attack.

3.3 Overview of Tiwari and Padhye's identity-based proxy multi-signature scheme

In 2011, Tiwari and Padhye [33] proposed an identity-based proxy multi-signature without bilinear pairings to improve efficiency of these kinds of schemes. This scheme includes a set of d original signers with identities $\langle ID_{o_1}, ..., ID_{o_d}\rangle$, a proxy signer with identity ID_p and a clerk. Their scheme consists of following algorithms:

1. Setup: The system parameters are as follows. Let $H_1 : \{0,1\}^* \to \mathbb{Z}_\alpha^*$ and $H_2 : \{0,1\}^* \times G \to \mathbb{Z}_l^*$ be random oracles, where p is a l-bit prime determining tuples $F_p, E/F_p, G, P$, where E/F_p denotes an elliptic curve E over a prime finite field F_p defined by equation $y^2 = x^3 + ax + b$ for a and $b \in F_p$ and discriminant $\Delta = 4a^3 + 27b^2 \ne 0$. Let G be a cyclic additive group with order α, and P be the generator of G. The key distribution center chooses $x \in_R \mathbb{Z}_\alpha^*$ and computes the master public key $P_{pub} = xP$. It publishes $mpk = P_{pub}$ as the master public key, and keeps the master secret key $msk = x$ secret. Therefore, public parameters are $Para = \{H_1, H_2, F_p, E/F_p, G, P\}$ and mpk.

2. KeyExtract: On input master secret key $msk = x$ and an identity ID_u, the key distribution center chooses at random $r_u \in \mathbb{Z}_\alpha^*$, computes $R_u = r_uP$, $h_u = H_1(ID_u, R_u)$ and $x_u = r_u + h_ux$, and sends the user secret key (x_u, R_u) over a secure and authenticated channel to the user with identity ID_u.

3. DelegationGen: Let w be the warrant to be signed by all original signers with identities $\langle ID_{o_1}, ..., ID_{o_d}\rangle$ who want to delegate their signing right to a proxy signer with identity ID_p, the delegation is $y = \{\sigma_{o_j}, K_{o_j}\}$, $1 \le j \le d$ which is generated as follows.

   - For $1 \le j \le d$, the original signer with identity ID_{o_j} chooses $k_{o_j} \xleftarrow{\$} \mathbb{Z}_\alpha^*$, and computes $K_{o_j} = k_{o_j}P$.
   - For $1 \le j \le d$, the original signer with identity ID_{o_j} computes $\sigma_{o_j} = e_{o_j}x_{o_j} + k_{o_j} \mod \alpha$, where $e_{o_j} = H_1(w, K_{o_j}, ID_p)$, and sends $y = \{\sigma_{o_j}, K_{o_j}\}$, $1 \le j \le d$ to the proxy signer with identity ID_p.

4. PKGen. The proxy signer with identity ID_p generates the proxy signing key D_p with computing $D_p = \sum_1^d(\sigma_{o_j} + x_p e_{p_j})$, where $e_{p_j} = H_1(w, K_{o_j}, ID_{o_j})$.

5. ProxyMSign: The proxy signer with identity ID_p can sign a message m under the warrant w with his secret key D_p as follows:

   – The proxy signer with identity ID_p chooses $b \xleftarrow{\$} \mathbb{Z}_\alpha^*$, computes $R = bP$ and $h = H_2(m, R)$, and checks if $gcd(b + h, \alpha) = 1$ holds. If it does, the proxy signer continues; otherwise, chooses another b.
   – The proxy signer with identity ID_p computes $s = (b + h)^{-1} D_p \bmod \alpha$ and the resulting signature is $\theta = (R_{o_j}, K_{o_j}, R_p, R, s)$, $1 \le j \le d$.

6. ProxyMVer: Given the identities $\langle ID_{o_1}, ..., ID_{o_d}\rangle$ of the original signers and identity ID_p of the proxy signer, a warrant w, a message m, and a signature $\theta = (R_{o_j}, K_{o_j}, R_p, R, s)$, $1 \le j \le d$, a verifier operates as follows:
   – Checks if the message m conforms to the warrant w, otherwise, it stops.
   – Checks if the proxy signer with identity ID_p is authorized by the original signers with identities $\langle ID_{o_1}, ..., ID_{o_d}\rangle$ in the warrant w, otherwise, it stops.
   – Accepts the proxy multi-signature if and only if $s(R+hP) = \sum_1^d e_{o_j}(R_{o_j}+h_{o_j}P_{pub})+K+\sum_1^d(e_{p_j}(R_p+h_pP_{pub}))$ holds, where $e_{o_j} = H_1(w, K_{o_j}, ID_p)$, $e_{p_j} = H_1(w, K_{o_j}, ID_{o_j})$, $h_{o_j} = H_1(ID_{o_j}, R_{o_j})$, $h_p = H_1(ID_p, R_p)$, $K = \sum_1^d K_{o_j}$ and $h = H_2(m, R)$.

3.4 Security analysis of Tiwari and Padhye's identity-based proxy multi-signature scheme

Tiwari and Padhye's scheme is forgeable in their proposed security model [33]. Since in their security model [33] they assumed that adversaries (original signers) in addition to having access to the KeyExtract algorithm on input ID_u, the DelegationGen algorithm on input $(w, ID_p, \langle ID_{o_1}, ..., ID_{o_d}\rangle)$ and the ProxyMSign algorithm on input $(w, m, ID_p, \langle ID_{o_1}, ..., ID_{o_d}\rangle)$, have access to the proxy signing key generation oracle, PKGen, of proxy signers on input $(ID_p, y', .)$ to obtain D_p', where ID_p is proxy signer's identity, $\langle ID_{o_1}, ..., ID_{o_d}\rangle$ are identities of original signers, y' is a valid delegation, and w and m are the warrant and the message in the warrant, respectively. Note that outputs of the KeyExtract, DelegationGen and ProxyMSign algorithm are x_u, y and θ, respectively. They also assumed that for a forged proxy multi-signature θ' under identities ID_p and $\langle ID_{o_1}, ..., ID_{o_d}\rangle$ including a delegation y, the adversaries (the original signers) that do not have proxy signer's long term secret key x_p, are not allowed to make query to the proxy signing key generation oracle, PKGen, on input $(ID_p, y, .)$ to obtain D_p, make KeyExtract query on input ID_p to obtain x_p, and make a ProxyMSign query on input $(w, m, ID_p, \langle ID_{o_1}, ..., ID_{o_d}\rangle)$ to attain θ'. Note that the delegation y is generated by original signers or adversaries obtained it with making query to the DelegationGen oracle.

   To forge the proxy multi-signature θ' including the delegation y regarding to adversary's capabilities in having access to oracles in their security model, the adversaries (the original signers) first make a query to the proxy signing key generation oracle, PKGen, on input $(ID_p, y', .)$ to obtain D_p', where $y' = \{\sigma_{o_j}', K_{o_j}'\}$, $1 \le j \le d$ is a valid delegation on w' and $y \ne y'$. Next, the adversaries (the original signers) can extract long term secret key, x_p, of the proxy signer with identity ID_p with computing $x_p = \frac{D_p' - \sum_{1 \le j \le d} \sigma_{o_j}'}{\sum_{1 \le j \le d} e_{p_j}'}$, where D_p' is the response of PKGen on input $(ID_p, y', .)$ and $e_{p_j}' = H_1(w', K_{o_j}', ID_{o_j}')$, $1 \le j \le d$.

   Then, the original signers or the adversaries with having the delegation y and long term secret key, x_p, of the proxy signer with identity ID_p can generate valid identity-based proxy multi-signatures on different messages in the warrant as follows. The original signers or the adversaries with the delegation $y = \{\sigma_{o_j}, K_{o_j}\}$, $1 \le j \le d$ generate the proxy signing key with computing $D_p = \sum_1^d(\sigma_{o_j} + x_p e_{p_j})$, where $e_{p_j} = H_1(w, K_{o_j}, ID_{o_j})$ for $1 \le j \le d$. Next, they choose $b' \xleftarrow{\$} \mathbb{Z}_\alpha^*$, and compute $R' = b'P$ and $h' = H_2(m', R')$, then check if $gcd(b' + h', \alpha) = 1$ holds. If it does, they continue, otherwise, they choose another b'. Then, they compute $s' = (b' + h')^{-1} D_p \bmod \alpha$ and the resulting signature is $\theta' = (R_{o_j}, K_{o_j}, R_p, R', s')$, $1 \le j \le d$ which passes the verification equation since it is generated according to the scheme. Note that in the proposed forgery attack the original signers or the adversaries with the delegation $y = \{\sigma_{o_j}, K_{o_j}\}$, $1 \le j \le d$ did not make a query to the proxy signing key generation oracle, PKGen, on input $(ID_p, y, .)$ to obtain D_p, did

not make KeyExtract query on input ID_p to obtain x_p, and did not make a ProxyMSign query on input $(w, m, ID_p, \langle ID_{o_1}, ..., ID_{o_d} \rangle)$ to attain θ'.

4 Our identity-based multi-proxy signature, proxy multi-signature and multi-proxy multi-signature schemes

In this section, we present an identity-based multi-proxy multi-signature scheme based on the GQ identity-based signature scheme [39]. Since identity-based proxy multi-signature and multi-proxy signature schemes are special cases of the proposed scheme such that $d = 1$ for the former and $n = 1$ in the latter, we omit their details. Then, we prove that the identity-based multi-proxy multi-signature scheme is secure under one-wayness of RSA in the random oracle model. Similarly, one may show that its special cases are secure in the random oracle model. To give some intuition into our schemes, we briefly recall the GQ scheme [39] here. The key distribution center generates an RSA module N and exponents (e, d) such that $ed = 1 \mod \varphi(N)$. The master public key is the pair (N, e), while d is the master secret key. The signature on a message m by identity ID_u is a pair (R_u, s_u) such that $R_u = r_u^e \mod N$, where r_u is a random number, and $s_u^e = R_u H(ID_u)^c \mod N$, where $c = H_1(R_u, m)$. Functions $\{H, H_1\}$ are random oracles.

Our scheme employs GQ identity-based multi-signature scheme, which is a different version of identity-based multi-signature from RSA [40] proposed by Neven and Bellare in the number of interactions and random oracles, as a building block such that the delegation is the original signers' GQ multi-signature on a warrant and the proxy signature is sequential aggregation of delegation and proxy agent's GQ identity-based multi-signature on a message m.

4.1 Details of identity-based multi-proxy multi-signature scheme

In this section, we present the details of identity-based multi-proxy multi-signature scheme. There are $n+d+1$ participants in the system, a group of original signers with identity set $\widehat{ID_o} = \langle ID_{o_1}, ..., ID_{o_d} \rangle$, a group of proxy signers with identity set $\widehat{ID_p} = \langle ID_{p_1}, ..., ID_{p_n} \rangle$ and a clerk, where d is the number of original signers in the original signers' group and n is the number of proxy signers in the proxy group. Our scheme consists of seven algorithms as follows.

1. Setup: The system parameters are as follows. Let l_1 and $l_N \in \mathbb{N}$ and let $H_1 : \{0,1\}^* \to \{0,1\}^{l_1}$, and $H : \{0,1\}^* \to \mathbb{Z}_N^*$ be random oracles, where H depends on the master public key of the scheme. Let KG_{rsa} be a RSA key pair generator that outputs triplets (N, e, d) such that $\varphi(N) > 2^{l_N}$ and with prime encryption exponent e of length strictly greater than $l_1 + 2\log_2^n + \log_2^d$ bits. The key distribution center runs KG_{rsa} to generate RSA parameters (N, e, d). It publishes $mpk = (N, e)$ as the master public key, and keeps the master secret key $msk = d$ secret. Therefore, public parameters are $Para = \{H_1, H\}$ and mpk.

2. KeyExtract: On input master secret key $msk = d$ and the user identity ID_u, the key distribution center computes $x_u = H(ID_u)^d \mod N$, and sends the user secret key x_u over a secure and authenticated channel to the user with identity ID_u.

3. MSign: Let m be a message to be signed with signers whose identity set is $\widehat{ID_u} = \langle ID_{u_1}, ..., ID_{u_d} \rangle$, the signature σ_u is generated as follows:

   - For $1 \leq j \leq d$, the signer with identity ID_{u_j} chooses $r_j \overset{\$}{\leftarrow} \mathbb{Z}_N^*$, computes $R_j = r_j^e \mod N$, and broadcasts R_j to $d-1$ signers and the clerk.
   - For $1 \leq j \leq d$, the signer with identity ID_{u_j} computes $R_u = \prod_{j=1}^d R_j$ and $c_0 = H_1(R_u || \widehat{ID_u} || m)$, and computes $s_j = r_j(x_{u_j})^{c_0} \mod N$.
   - For $1 \leq j \leq d$, the signer with identity ID_{u_j} sends (c_0, s_j) to the clerk as her partial signature on the message m.

– The clerk checks if $R_j = s_j^e H(ID_{u_j})^{-c_0}$ holds for $1 \le j \le d$.

When all the partial signatures are valid, the identity-based multi-signature of the message m w.r.t. identity set $\widehat{ID_u} = \langle ID_{u_1}, ..., ID_{u_d} \rangle$ of signers is generated as $\sigma_u = (R_u, s_u)$ by computing $s_u = \prod_{j=1}^{d} s_j$.

4. MVer: Given the identity set $\widehat{ID_u}$ of signers, the message m and a signature $\sigma_u = (R_u, s_u)$, a verifier checks if $s_u^e = R_u[\prod_{j=1}^{d} H(ID_{u_j})]^{c_0} \bmod N$ holds, where $c_0 = H_1(R_u||\widehat{ID_u}||m)$.

5. DelegationGen: Let w be a warrant to be signed by original signers with identity set $\widehat{ID_o} = \langle ID_{o_1}, ..., ID_{o_d} \rangle$ who want to delegate their signing right to a proxy agent with identity set $\widehat{ID_p} = \langle ID_{p_1}, ..., ID_{p_n} \rangle$, the delegation σ_o is generated with employing MSign algorithm on the message $m = (w||11)$, where the identity set of signers in the algorithm are identity set $\widehat{ID_o}$ of original signers.

6. MProxyMSign: The proxy agent with identity set $\widehat{ID_p}$ can sign a message m under the warrant w with having a valid delegation $\sigma_o = (R_o, s_o)$ as follows:
   – For $1 \le i \le n$, the proxy signer with identity ID_{p_i} chooses $r_i \xleftarrow{\$} \mathbb{Z}_N^*$, computes $R_i = r_i^e \bmod N$, and broadcasts R_i to $n-1$ proxy signers and the clerk.
   – For $1 \le i \le n$, the proxy signer with identity ID_{p_i} computes $R_p = \prod_{i=1}^{n} R_i$, $c_1 = H_1(R_p||R_o||\widehat{ID_o}||\widehat{ID_p} ||w||m||01)$, $s_i = r_i(x_{p_i})^{c_1} \bmod N$ and $\hat{s}_i = s_i s_o \bmod N$.
   – For $1 \le i \le n$, the proxy signer with identity ID_{p_i} sends $(c_1, \hat{s}_i, R_o)$ to the clerk as his partial proxy signature on the message m under the warrant w.
   – The clerk checks if $R_i = \hat{s}_i^e([H(ID_{p_i})]^{c_1} R_o[\prod_{j=1}^{d} H(ID_{o_j})]^{c_0})^{-1}$ holds for $1 \le i \le n$, where $c_0 = H_1(R_o||\widehat{ID_o}||w||11)$.

   When all partial proxy signatures are valid, the identity-based multi-proxy multi-signature of the message m under the warrant w w.r.t. identity sets $\widehat{ID_o}$ for original signers and $\widehat{ID_p}$ for proxy signers is generated as $\theta = (R_p, R_o, s_p)$ by computing $s_p = \prod_{i=1}^{n} \hat{s}_i$.

7. MProxyMVer: Given identity set $\widehat{ID_o}$ of original signers and identity set $\widehat{ID_p}$ of proxy signers, a warrant w, a message m, and a signature $\theta = (R_p, R_o, s_p)$, a verifier operates as follows:
   – Checks if the message m conforms to the warrant w, otherwise, it stops.
   – Checks if n proxy signers with identity set $\widehat{ID_p}$ are authorized by original signers with identity set $\widehat{ID_o}$ in the warrant w, otherwise, it stops.
   – Accepts the multi-proxy multi-signature if and only if $s_p^e = R_p[\prod_{i=1}^{n} H(ID_{p_i})]^{c_1}(R_o[\prod_{j=1}^{d} H(ID_{o_j})]^{c_0})^n$ holds, where $c_1 = H_1(R_p||R_o||\widehat{ID_o}||\widehat{ID_p}||w||m||01)$ and $c_0 = H_1(R_o||\widehat{ID_o}||w||11)$.

4.2 Analysis of the proposed schemes

In this section, we verify the correctness and prove existential unforgeability of the new identity-based multi-proxy multi-signature scheme in the random oracle model (see [41] for the background). Note that one may similarly verify the correctness and prove unforgeability of other schemes, namely proxy multi-signature and multi-proxy signature schemes, since they are special cases of the identity-based multi-proxy multi-signature scheme.

In order to prove unforgeability of the proposed scheme, we need to show that it is unforgeable against adversaries of types II and III (as defined in Section 2.3). Since our security proofs are quite similar in both cases, we have parameterized these proofs to prevent unnecessary repetitions of arguments. Hence, just for notational settings, we refer to the adversary as $A_{(1-k)II+kIII}$ in which the parameter $k \in \{0, 1\}$ makes the difference between adversaries of types II and III (i.e. notationally we assume that we have an adversary of type II, A_{II}, when $k = 0$ and an adversary of type III, A_{III}, when $k = 1$). Note that, the proofs for different values of k are independent.

In the security proof, it is assumed that there is only one honest signer, and note that, this does not have any effect on the generality of our proof. Also, this assumption in different security models is in coherence

with what is usually considered in the current literature (e.g. see [13, 29–33, 36, 37] for similar issues) since security of the general case of having multiple honest signers follows from the security of the extreme case of having only one honest signer in a more or less straightforward way. Although, this fact is not usually explicitly stated in this literature, to be precise, we have decided to elaborate on the details of this proof in Section 5.

To prove the security of our proposed scheme, and by contradiction, assuming an adversary $A_{(1-k)II+kIII}$, we show that there is a solver (algorithm B) that can solve a random instance of the RSA problem with a nonnegligible probability. To do this, we first show that there exists a simulator called $C_{A_{(1-k)II+kIII}}$ (see Algorithm 2) that can simulate the signature scheme without knowing the secret key of the honest signer, and runs the adversary $A_{(1-k)II+kIII}$ as its sub-routine. In this regard, we compute the run-time and a lower-bound for the success (returning a *useful output* $(R_p, R_o, s_p, c_k, c_{1-k}, x_o, x_p, m, w)$ (see Definition 3)) probability of this simulator in terms of the run-time and success (returning a valid forgery $\theta = (R_p, R_o, s_p, c_k, c_{1-k})$ on a message m under the warrant w with respect to the original signers' identity set $\widehat{ID_o}$ and the proxy signers' identity set $\widehat{ID_p}$) probability of the adversary and the number of queries to the oracles (see Lemma 1).

At the final stage, we use a forking strategy to solve an instance (N, e, y) of the RSA problem, using a *useful pair* (see Definition 4) of the simulator $C_{A_{(1-k)II+kIII}}$ when the random string used in both simulations are the same. Hence, we concentrate on computing a lower bound for the probability of producing such a *useful pair* and solving the RSA instance as the main body of the solver algorithm B (see Lemma 3). We should highlight that the general Forking Lemma [42] cannot be applied directly into our scheme since this scheme is the result of sequential aggregation of two different signatures such that we have two different types of random oracle responses. Hence, we need to consider the probability of happening some random responses before the forking point in the proposed forking lemma, and this is the main difference of our Forking Lemma from previous ones.

Our main result on the security of the proposed scheme is summarized in Theorem 1, where the parameter k is used to code the result for both adversaries of types II and III.

To start let us verify the correctness of the proposed scheme. Note that, all computations are done modulo N, but we omit this for simplicity.

$$
\begin{aligned}
s_p^e &= [\textstyle\prod_{i=1}^n \hat{s}_i]^e \\
&= \textstyle\prod_{i=1}^n [r_i^e x_{p_i}{}^{ec_1} [\textstyle\prod_{j=1}^d r_j^e x_{o_j}{}^{ec_0}]] \\
&= [\textstyle\prod_{i=1}^n r_i^e][\textstyle\prod_{i=1}^n H(ID_{p_i})]^{c_1} ([\textstyle\prod_{j=1}^d r_j^e][\textstyle\prod_{j=1}^d H(ID_{o_j})]^{c_0})^n \\
&= R_p[\textstyle\prod_{i=1}^n H(ID_{p_i})]^{c_1} (R_o[\textstyle\prod_{j=1}^d H(ID_{o_j})]^{c_0})^n.
\end{aligned}
\tag{2}
$$

Definition 3. Let $k \in \{0, 1\}$ be a constant and the algorithm $C_{A_{(1-k)II+kIII}}$ return $(R_p, R_o, s_p, c_k, c_{1-k}, x_o, x_p, m, w)$ derived from a valid forgery $((\widehat{ID_o}, \widehat{ID_p}, m, w), \theta = (R_p, R_o, s_p, c_k, c_{1-k}))$ produced by an adversary $A_{(1-k)II+kIII}$ when $C_{A_{(1-k)II+kIII}}$ simulates the signature scheme. The tuple $(R_p, R_o, s_p, c_k, c_{1-k}, x_o, x_p, m, w)$ is a *useful output* if $s_p^e = R_p[x_p^e y^k]^{c_1} (R_o[x_o^e y^{1-k}]^{c_0})^n$ holds.

Lemma 1. *Let $k \in \{0, 1\}$, $n \geq 1$ and $d \geq 1$ be constants and l_N be a security parameter. Assuming the existence of an adversary, $A_{(1-k)II+kIII}$, with success probability at least ϵ and run-time t, there exists a simulator $C_{A_{(1-k)II+kIII}}$ for the signature scheme that does not use the secret key of the honest signer, and produces a useful output $(R_p, R_o, s_p, c_k, c_{1-k}, x_o, x_p, m, w)$ (see Algorithm 2 for the pseudocode and the definitions) such that,*

a) *the success probability of $C_{A_{(1-k)II+kIII}}$ is greater than*

$$
\varepsilon \overset{\text{def}}{=} \frac{\epsilon}{4q_E} - (2(1-k)q_d^2 + 2kq_s^2 + ((1-k)q_d + kq_s)q_H)2^{-l_N},
$$

b) *the run-time of $C_{A_{(1-k)II+kIII}}$ is less than*

$$
\tau \overset{\text{def}}{=} t + (1q_E + 1q_h + 2(1-k)dq_d + 2k(n+1)q_s)t_{exp},
$$

where t_{exp} is the time of one exponentiation in $\mathbb{Z}_N^*$, and q_H, q_h, q_E, q_d and q_s are the number of queries to the oracles H_1, H, KeyExtract, DelegationGen and MProxyMSign, respectively.

Proof. Assume the existence of an adversary $A_{(1-k)II+kIII}$ on the public data $mpk = (N, e)$ which runs in time at most t, makes q_H queries to the random oracle H_1, q_h queries to the random oracle H, q_E queries to the KeyExtract, $(1 - k)q_d$ queries to the DelegationGen and kq_s queries to the MProxyMSign algorithm, and can win the unforgeability game with probability at least ϵ. The algorithm $C_{A_{(1-k)II+kIII}}$ maintains initially empty associative arrays $T_1[.]$ and $T[.]$, and answers $A_{(1-k)II+kIII}$'s oracle queries as described below (see Algorithm 2 and note that this algorithm uses the adversary $A_{(1-k)II+kIII}$ and Algorithm 1 as its sub-routines).

- $H_1(Q)$ queries: If $T_1[Q]$ is defined then $C_{A_{(1-k)II+kIII}}$ returns its value, otherwise $C_{A_{(1-k)II+kIII}}$ chooses $T_1[Q] \xleftarrow{\$} \{0, 1\}^{l_1}$, and returns $T_1[Q]$ to $A_{(1-k)II+kIII}$. Note that, in DelegationGen $Q = (R_o||\widehat{ID_o}||w||11)$ and in MProxyMSign $Q = (R_p||R_o||\widehat{ID_o}||\widehat{ID_p}||w||m||01)$.

- $H(ID_u)$ queries: We employ Coron's technique [43] to obtain a tighter security bound when simulating H. If $T[ID_u] = (b, x_u, X_u)$ then $C_{A_{(1-k)II+kIII}}$ returns X_u. If this entry is not yet defined, it chooses $x_u \xleftarrow{\$} \mathbb{Z}_N^*$ and tosses a biased coin b so that $b = 0$ with probability β and $b = 1$ with probability $1 - \beta$. If $b = 0$, then $C_{A_{(1-k)II+kIII}}$ sets $X_u = x_u^e \bmod N$; if $b = 1$, it sets $X_u = x_u^e y \bmod N$. It stores $T[ID_u] \leftarrow (b, x_u, X_u)$ and returns X_u to $A_{(1-k)II+kIII}$. Note that, the value of β is determined later.

- KeyExtract queries for ID_u: The algorithm $C_{A_{(1-k)II+kIII}}$ looks up $T[ID_u] = (b, x_u, X_u)$, if this entry is not yet defined, it performs a query $H(ID_u)$. If $b = 0$, then $C_{A_{(1-k)II+kIII}}$ returns x_u; otherwise, it sets $bad_{KE} \leftarrow true$ and aborts the execution of $A_{(1-k)II+kIII}$ and returns $\perp$.

- DelegationGen queries for a warrant w w.r.t. the multi-set $\widehat{ID_o}$ of original signers including one honest original signer whose identity is denoted as ID_{o_t}: The algorithm $C_{A_{II}}$ first makes query to H oracle to obtain $H(ID_{o_j}) = X_{o_j}$ for $1 \leq j \leq d$, next, $C_{A_{II}}$ chooses $c_0 \xleftarrow{\$} \{0, 1\}^{l_1}$ and $s_t \xleftarrow{\$} \mathbb{Z}_N^*$, computes $R_t \leftarrow s_t^e (X_{o_t})^{-c_0} \bmod N$, and broadcasts the value of R_t of the honest original signer. The algorithm $C_{A_{II}}$ at the same time receives R_j, $1 \leq j \neq t \leq d$, of other corrupted original signers from A_{II} (the adversary plays the role of the corrupted co-original signers with having their secret keys in the group of original signers), and computes $R_o = \prod_{j=1}^d R_j$. If $T_1[R_o||\widehat{ID_o}||w||11]$ has already been defined, then $C_{A_{II}}$ sets $bad_{DG} \leftarrow true$ and halts returning $\perp$; otherwise, it sets $T_1[R_o||\widehat{ID_o}||w||11] \leftarrow c_0$. After having received s_j, $1 \leq j \neq t \leq d$ from adversary A_{II}, $C_{A_{II}}$ checks if $R_j = s_j^e H(ID_{o_j})^{-c_0}$ holds for $1 \leq j \neq t \leq d$. If not, it ends the signing protocol. Otherwise, it computes $s_o = \prod_{j=1}^d s_j$, and returns the signature $\sigma_o = (R_o, s_o, c_0)$ as a delegation to the adversary A_{II}.

- MProxyMSign queries for a message m under the warrant w w.r.t. proxy signers' identity set $\widehat{ID_p}$ including one honest proxy signer whose identity is denoted as ID_{p_t} such that m and $\widehat{ID_p}$ are in the warrant w: The algorithm $C_{A_{III}}$ receives $\sigma_o = (R_o, s_o, c_0)$ as a delegation from A_{III}, which generates it by itself since it has secret keys of all original signers. Next, if the delegation is valid, $C_{A_{III}}$ makes query to the H oracle to obtain $H(ID_{p_i}) = X_{p_i}$ for $1 \leq i \leq n$. Then, $C_{A_{III}}$ chooses $c_1 \xleftarrow{\$} \{0, 1\}^{l_1}$ and $s_t \xleftarrow{\$} \mathbb{Z}_N^*$, computes $R_t \leftarrow (s_t s_o)^e X_{p_t}^{-c_1} \bmod N$ for the honest proxy signer, and broadcasts R_t. The algorithm $C_{A_{III}}$ at the same time receives R_i, $1 \leq i \neq t \leq n$, of corrupted proxy signers from A_{III}, which has their secret keys, and plays the role of the corrupted co-proxy signers, and computes $R_p = \prod_{i=1}^n R_i$. If $T_1[R_p||R_o||\widehat{ID_o}||\widehat{ID_p}||w||m||01]$ has already been defined, then, $C_{A_{III}}$ sets $bad_{MP} \leftarrow true$ and halts returning $\perp$; otherwise, it sets $T_1[R_p||R_o||\widehat{ID_o}||\widehat{ID_p}||w||m||01] \leftarrow c_1$. After having received $\hat{s}_i$ for corrupted proxy signers from A_{III}, $C_{A_{III}}$ verifies that if $R_i = \hat{s}_i^e (H(ID_{p_i})^{c_1} R_o [\prod_{j=1}^d H(ID_{o_j})]^{c_0})^{-1}$ holds for $1 \leq i \neq t \leq n$. If not, it ends the signing protocol. Otherwise, it computes $s_p = \prod_{i=1}^n \hat{s}_i$, and returns the signature $\theta = (R_p, R_o, s_p, c_k, c_{1-k})$ on the message m under the warrant w w.r.t. $\widehat{ID_o}$ and $\widehat{ID_p}$ to the adversary A_{III}.

To lower-bound the probability that $C_{A_{(1-k)II+kIII}}$ does not abort at answering to queries of $A_{(1-k)II+kIII}$, we need to compute $\eta = \Pr[\neg bad_{KE}]((1 - k) \Pr[\neg bad_{DG}|\neg bad_{KE}] + k \Pr[\neg bad_{MP}|\neg bad_{KE}])$, where events bad_{KE}, bad_{DG} and bad_{MP} indicate that $C_{A_{(1-k)II+kIII}}$ aborts in signature simulation as a result of any of

---

Algorithm 1 $H(ID_u)$ (Global variables (N, e, y))

Choose $x_u \xleftarrow{\$} \mathbb{Z}_N^*$ and toss a biased coin b ($b = 0$ with probability β and $b = 1$ with probability $1 - \beta$) **and** compute X_u (i.e. $X_u = x_u^e \bmod N$ for $b = 0$, $X_u = x_u^e y \bmod N$ for $b = 1$ **and** set $T[ID_u] \leftarrow (b, x_u, X_u)$

Algorithm 2 $C_{A_{(1-k)II+kIII}}(N, e, y)$

1: $Done = 0$.
2: $(Done, \xi_1 = (Question, Oracle), \xi_2) \leftarrow A_{(1-k)II+kIII}(N, e)$
3: **while** $\neg Done$ **do**
4:    **if** $\xi_1 = (Q, H_1)$ **and** $\xi_2 = \perp$ **then**
5:       **if** $T_1[Q]$ is defined **then**
6:          **return** $T_1[Q]$
7:       **else**
8:          Set $T_1[Q] \xleftarrow{\$} \{0, 1\}^{l_1}$ **and**
9:          **return** $T_1[Q]$
10:       **end if**
11:    **end if**
12:    **if** $\xi_1 = (ID_u, H)$ **and** $\xi_2 = \perp$ **then**
13:       **if** $T[ID_u] = (b, x_u, X_u)$ is defined **then**
14:          **return** X_u
15:       **else**
16:          Run $H(ID_u)$ **and**
17:          **return** X_u
18:       **end if**
19:    **end if**
20:    **if** $\xi_1 = (ID_u, KeyExtract)$ **and** $\xi_2 = \perp$ **then**
21:       **if** $T[ID_u] = (b, x_u, X_u)$ is not defined **then**
22:          Run $H(ID_u)$
23:          **if** $b = 0$ **then**
24:             **return** x_u
25:          **else**
26:             Set $bad_{KE} \leftarrow true$ **and**
27:             **return** $\perp$
28:          **end if**
29:       **end if**
30:    **end if**
31:    **if** $\xi_1 = ((w, \widehat{ID_o}), DelegationGen)$ **and** $\xi_2 = \perp$ **then**
32:       Run $H(ID_{o_j})$ for $1 \leq j \leq d$
33:       Choose $c_0 \xleftarrow{\$} \{0, 1\}^{l_1}$, $s_t \xleftarrow{\$} \mathbb{Z}_N^*$, compute $R_t \leftarrow s_t^e(X_{o_t})^{-c_0} \bmod N$ (broadcast R_t and receive R_js for $1 \leq j \neq t \leq d$ from A_{II}) and compute $R_o = \prod_{j=1}^{d} R_j$.
34:       **if** $T_1[R_o||\widehat{ID_o}||w||11]$ has already been defined **then**
35:          Set $bad_{DG} \leftarrow true$ **and**
36:          **return** $\perp$
37:       **else**
38:          Set $T_1[R_o||\widehat{ID_o}||w||11] \leftarrow c_0$ **and** compute $s_o = \prod_{j=1}^{d} s_j$ (s_js for $1 \leq j \neq t \leq d$ are received from A_{II} and checks their correctness through relation $R_j = s_j^e H(ID_{o_j})^{-c_0}$) **and**
39:          **return** $\sigma_o = (R_o, s_o, c_0)$ on w w.r.t. $\widehat{ID_o}$
40:       **end if**
41:    **end if**
42:    **if** $\xi_1 = ((m, \sigma_o, \widehat{ID_p}), MProxyMSign)$ **and** $\xi_2 = \perp$ **then**
43:       Run $H(ID_{p_i})$ for $1 \leq i \leq n$
44:       Choose $c_1 \xleftarrow{\$} \{0, 1\}^{l_1}$ and $s_t \xleftarrow{\$} \mathbb{Z}_N^*$, compute $R_t \leftarrow (s_t s_o)^e X_{p_t}^{-c_1} \bmod N$ (broadcast R_t and receive R_is for $1 \leq i \neq t \leq n$ from A_{III}) and compute $R_p = \prod_{i=1}^{n} R_i$.
45:       **if** $T_1[R_p||R_o||\widehat{ID_o}||\widehat{ID_p}||w||m||01]$ has already been defined **then**
46:          Set $bad_{MP} \leftarrow true$ **and**
47:          **return** $\perp$
48:       **else**
49:          Set $T_1[R_p||R_o||\widehat{ID_o}||\widehat{ID_p}||w||m||01] \leftarrow c_1$ **and** compute $s_p = \prod_{i=1}^{n} \hat{s_i}$ ($\hat{s_i}$s for $1 \leq i \neq t \leq n$ are received from A_{III} and their correctness are checked through relation $R_i = \hat{s_i}^e (H(ID_{p_i})^{c_1} R_o[\prod_{j=1}^{d} H(ID_{o_j})]^{c_0})^{-1})$ for $1 \leq i \neq t \leq n$ **and**
50:          **return** $\theta = (R_p, R_o, s_p, c_k, c_{1-k})$ on m under w w.r.t. $\widehat{ID_o}$ and $\widehat{ID_p}$
51:       **end if**
52:    **end if**
53: **end while**
54: **if** $Done$ **then**
55:    $A_{(1-k)II+kIII}$ returns $(\xi_1 = \perp, \xi_2 = ((\widehat{ID_o}, \widehat{ID_p}, m, w)), \theta = (R_p, R_o, s_p, c_k, c_{1-k}))$
56:    Look up $T[ID_{p_i}]$ for $1 \leq i \leq n$ **and** $T[ID_{o_j}]$ for $1 \leq j \leq d$ **and** compute $x_o = \prod_{j=1}^{d} x_{o_j}$ and $x_p = \prod_{i=1}^{n} x_{p_i}$ **and**
57:    **return** $(R_p, R_o, s_p, c_k, c_{1-k}, x_o, x_p, m, w)$
58: **end if**

$A_{(1-k)II+kIII}$'s KeyExtract, DelegationGen and MProxyMSign queries, respectively. These probabilities are computed as follows.

Claim 1. $\Pr[\neg bad_{KE}] \geq \beta^{q_E}$.

Proof. $\Pr[\neg bad_{KE}]$ is the probability that $C_{A_{(1-k)II+kIII}}$ does not abort as a result of $A_{(1-k)II+kIII}$'s KeyExtract queries. The algorithm $C_{A_{(1-k)II+kIII}}$ aborts at answering to a KeyExtract query when bad_{KE} is set to true (Algorithm 2, Line 26) which means that $b = 1$ for a given identity. The probability of this event is $1 - \beta$, so the probability that $C_{A_{(1-k)II+kIII}}$ does not abort for one KeyExtract query is β. Since $A_{(1-k)II+kIII}$ makes at most q_E KeyExtract queries, the probability that $C_{A_{(1-k)II+kIII}}$ does not abort as a result of q_E KeyExtract queries is at least β^{q_E}.

Claim 2. $\Pr[\neg bad_{DG}|\neg bad_{KE}] \geq 1 - q_d((q_d + q_H)2^{-l_N}) - q_d^2 2^{-l_N}$.

Proof. Events $\neg bad_{KE}$ and $\neg bad_{DG}$ are independent, so $\Pr[\neg bad_{DG}|\neg bad_{KE}] = \Pr[\neg bad_{DG}]$. The value of $\Pr[\neg bad_{DG}]$ is the probability that $C_{A_{II}}$ does not abort as a result of DelegationGen queries. The algorithm $C_{A_{II}}$ aborts at answering to a DelegationGen query if bad_{DG} is set to true (Algorithm 2, Line 35) which means that there is a conflict in the table $T_1[.]$ for these kinds of queries. The probability of finding a conflict in $T_1[.]$ for one DelegationGen query $(w, \widehat{ID_o})$ equals the probability that $(R_o||\widehat{ID_o}||w||11)$ generated in a DelegationGen simulation has been occurred by chance in a previous query to the oracle H_1. Since there are at most $q_H + q_d$ entries in the table $T_1[.]$ for these kinds of queries and the number of R_o, uniformly distributed in $\mathbb{Z}_N$, is 2^{l_N}, the probability of this event for one DelegationGen query is at most $(q_d + q_H)2^{-l_N}$. Hence, the probability of this event for q_d queries is at most $q_d(q_d + q_H)2^{-l_N}$. In addition, this probability includes the probability that $C_{A_{II}}$ previously used the same randomness R_t, uniformly distributed in $\mathbb{Z}_N$, in one DelegationGen simulation. Since there are at most q_d DelegationGen simulations, this probability is at most $q_d 2^{-l_N}$. Therefore, for q_d DelegationGen queries the probability of this event is at most $q_d^2 2^{-l_N}$.

Claim 3. $\Pr[\neg bad_{MP}|\neg bad_{KE}] \geq 1 - q_s((q_s + q_H)2^{-l_N}) - q_s^2 2^{-l_N}$.

Proof. Events $\neg bad_{KE}$ and $\neg bad_{MP}$ are independent, so $\Pr[\neg bad_{MP}|\neg bad_{KE}] = \Pr[\neg bad_{MP}]$. The value of $\Pr[\neg bad_{MP}]$ is the probability that $C_{A_{III}}$ does not abort as a result of MProxyMSign queries. The algorithm $C_{A_{III}}$ aborts at answering to a MProxyMSign query if bad_{MP} is set to true (Algorithm 2, Line 46) which means that there is a conflict in table $T_1[.]$ for these kinds of queries. The probability of finding a conflict in $T_1[.]$ for one MProxyMSign query equals the probability that $(R_p||R_o||\widehat{ID_o}||\widehat{ID_p}||w||m||01)$ generated in MProxyMSign simulation has been occurred by chance in a previous query to the oracle H_1. Since there are at most $q_H + q_s$ entries in the table $T_1[.]$ for these kinds of queries and the number of R_p, uniformly distributed in $\mathbb{Z}_N$, is 2^{l_N}, the probability of this event for one MProxyMSign is at most $(q_s + q_H)2^{-l_N}$. Hence, the probability of this event for q_s queries is at most $q_s(q_s + q_H)2^{-l_N}$. In addition, this probability includes the probability that $C_{A_{III}}$ previously used the same randomness R_t, uniformly distributed in $\mathbb{Z}_N$, in one MProxyMSign simulation. Since there are at most q_s MProxyMSign simulations, this probability is at most $q_s 2^{-l_N}$. Therefore, for q_s MProxyMSign queries the probability of this event is at most $q_s^2 2^{-l_N}$.

Finally, it is assumed that $A_{(1-k)II+kIII}$ outputs a valid forgery $\theta = (R_p, R_o, s_p, c_k, c_{1-k})$ on a message m under a warrant w w.r.t. original signers' identity set $\widehat{ID_o}$ and proxy signers' identity set $\widehat{ID_p}$ with probability at least ϵ in time bound t. Since the forgery is valid, we have

$$s_p^e = R_p[\textstyle\prod_{i=1}^n H(ID_{p_i})]^{c_1}(R_o[\textstyle\prod_{j=1}^d H(ID_{o_j})]^{c_0})^n, \tag{3}$$

and A_{II} has not asked the warrant w from DelegationGen algorithm under original signer's identity set $\widehat{ID_o}$ and A_{III} has not asked the message m from MProxyMSign algorithm under proxy signer's identity set $\widehat{ID_p}$. In addition, a valid forgery has to contain at least one uncorrupted identity. The probability that $A_{(1-k)II+kIII}$ outputs a valid forgery containing at least one uncorrupted identity is computed as follows.

Claim 4. The probability that $A_{(1-k)II+kIII}$ outputs a valid forgery including at least one uncorrupted identity is at least $\epsilon(1-\beta)$.

Proof. It is assumed that $A_{(1-k)II+kIII}$ outputs a valid forgery with probability at least ϵ. The probability that a valid forgery contains at least one uncorrupted identity is at least $1-\beta$. The probability of existence of one honest identity with $b=1$ is $1-\beta$, and for adversaries of type II the probability of existence of at least one uncorrupted identity among $d+n$ identities is $1-\beta^d$ (since A_{II} has secret keys of n proxy signers, we have at most d uncorrupted identities in the forgery). Since we have $1-\beta \le 1-\beta^d$ for $d \ge 1$, this probability is at least $1-\beta$. Similarly we have the same claim for A_{III}. Therefore, the probability that $A_{(1-k)II+kIII}$ outputs a valid forgery containing at least one uncorrupted identity is at least $\epsilon(1-\beta)$.

Therefore, the probability that $C_{A_{(1-k)II+kIII}}$ returns a useful output is at least $\epsilon(1-\beta)\eta \ge \epsilon(1-\beta)\beta^{q_E} - ((1-k)q_d((2q_d+q_H)2^{-l_N}) + kq_s(2q_s+q_H)2^{-l_N})$. The value of $\beta^{q_E}(1-\beta)$ is maximized for $\beta = \frac{q_E}{q_E+1}$. With substituting the value of β, we obtain $\beta^{q_E}(1-\beta) = (\frac{q_E}{q_E+1})^{q_E}\frac{1}{q_E+1} = \frac{1}{q_E}(1-\frac{1}{q_E+1})^{1+q_E}$. If $q_E = 0$, this value is 1 and $(1-\frac{1}{q_E+1})^{1+q_E}$ is a monotonically increasing sequence for $q_E \ge 1$. Therefore, the lower bound of $\beta^{q_E}(1-\beta)$ is $\frac{1}{4q_E}$.

To estimate the required time of $C_{A_{(1-k)II+kIII}}$ in returning a useful output, the required time t_C in which $C_{A_{(1-k)II+kIII}}$ answers $A_{(1-k)II+kIII}$'s queries is computed as follows. Since it is assumed that a (multi-) exponentiation in $\mathbb{Z}_N$ takes time t_{exp}, while all other operations take zero time, each random oracle or KeyExtract query takes at most one exponentiation, a delegation simulation takes $2d$ exponentiations, and a multi-proxy multi-signature simulation takes $2(n+1)$ exponentiations, we therefore have $t_C \le (1q_E + 1q_h + 2(1-k)dq_d + 2k(n+1)q_s)t_{exp}$.

Finally, $C_{A_{(1-k)II+kIII}}$ performs additional random oracle queries $H(ID_u)$ for identities in the forgery to find $T[ID_u] = (b, x_u, X_u)$ for them, computes $x_o = \prod_{j=1}^{d} x_{o_j}$ and $x_p = \prod_{i=1}^{n} x_{p_i}$, and returns $(R_p, R_o, s_p, c_k, c_{1-k}, x_o, x_p, w, m)$ with probability at least $\varepsilon = \frac{\epsilon}{4q_E} - (2(1-k)q_d^2 + 2kq_s^2 + ((1-k)q_d + kq_s)q_H)2^{-l_N}$ in time bound $\tau = t + (1q_E + 1q_h + 2(1-k)dq_d + 2k(n+1)q_s)t_{exp}$. Substituting the values of $H(ID_u) = x_u^e$ for corrupted identities and $H(ID_u) = x_u^e y$ for the honest identity in Equation 3, we obtain

$$s_p^e = R_p([\prod_{i=1}^{n}(x_{p_i})^e]y^k)^{c_1}(R_o([\prod_{j=1}^{d}(x_{o_j})^e]y^{1-k})^{c_0})^n.$$

Since $x_o = \prod_{j=1}^{d} x_{o_j}$ and $x_p = \prod_{i=1}^{n} x_{p_i}$, the above relation converts to

$$s_p^e = R_p[x_p^e y^k]^{c_1}(R_o[x_o^e y^{1-k}]^{c_0})^n.$$

$\square$

Also, in what follows we will be needing the following splitting lemma.

Lemma 2. [44] Let $A \subset X \times Y$ such that $\Pr[(x,y) \in A] \ge \delta$. For any $\alpha < \delta$, define $B = \{(x,y) \in X \times Y \mid \Pr_{y' \in Y}[(x,y') \in A] \ge \delta - \alpha\}$ and $\bar{B} = (X \times Y) \setminus B$, then the following statements hold:

– $\Pr[B] \ge \alpha$
– $\forall(x,y) \in B, \Pr_{y' \in Y}[(x,y') \in A] \ge \delta - \alpha\}$
– $\Pr[B|A] \ge \frac{\alpha}{\delta}$.

Definition 4. Let $k \in \{0,1\}$ be a constant. A pair of useful outputs $(R_p, R_o, s_p, c_k, c_{1-k}, x_o, x_p, m, w)$ and $(R_p', R_o', s_p', c_k', c_{1-k}', x_o', x_p', m', w')$ is said to be a *useful pair* if $R_p = R_p'$, $R_o = R_o'$, $s_p \ne s_p'$, $c_k \ne c_k'$, $c_{1-k} = c_{1-k}'$, $x_o = x_o'$, $x_p = x_p'$, $m = m'$ and $w = w'$ hold.

Definition 5. The probabilistic polynomial time algorithm $C_{A_{(1-k)II+kIII}}$ at each run proceeds based on a random string ω and answers $\rho \overset{\text{def}}{=} (\rho_1, ..., \rho_{q_t})$ to the queries $\mathcal{Q} \overset{\text{def}}{=} (Q_1, ..., Q_{q_t})$ made to the random oracle H_1. A pair of (ω, ρ) is said to be a *successful pair* if $C_{A_{(1-k)II+kIII}}$ produces a useful output $(R_p, R_o, s_p, c_k, c_{1-k}, x_o, x_p, w, m)$ based on them.

Lemma 3. *(A Forking Lemma).* Let $k \in \{0, 1\}$ be a constant, l_1 be a security parameter, H_1 be a random oracle, and q_t be the total number of queries to H_1. It is assumed that $C_{A_{(1-k)II+kIII}}$ returns a useful output $(R_p, R_o, s_p, c_k, c_{1-k}, x_o, x_p, m, w)$ with probability at least ε in time bound τ. Then, a replay of $C_{A_{(1-k)II+kIII}}$ with the same random string and a different random oracle gives a useful pair in time $t' \le 2\tau$ with probability $\epsilon' \ge \frac{\epsilon_1^2(1-2^{-l_1})}{8q_t(q_t-1)}$, where $\epsilon_1 \ge (\varepsilon - 2^{-(l_1-1)})$.

Proof. Consider the probabilistic polynomial time Turing machine $C_{A_{(1-k)II+kIII}}$ with a random string ω, that answers to the queries $\mathcal{Q} \overset{\text{def}}{=} (Q_1, ..., Q_{q_t})$ made to the random oracle H_1, and stores these queries and the corresponding answers $\rho \overset{\text{def}}{=} (\rho_1, ..., \rho_{q_t})$ in the table $T_1[.]$. For a given query Q, let index of Q be defined as $Ind(Q) \overset{\text{def}}{=} \min\{i | Q_i = Q\}$. By hypothesis, for a random choice of (ω, ρ), $C_{A_{(1-k)II+kIII}}$ produces a useful output $(R_p, R_o, s_p, c_k, c_{1-k}, x_o, x_p, w, m)$ with probability at least ε in time bound τ.

Since H_1 is a random oracle, the probability of the event

$$c_k = H_1((1-k)(R_o||\widehat{ID_o}||w||11) + k(R_p||R_o||\widehat{ID_o}||\widehat{ID_p}||w||m||01))$$

and

$$c_{1-k} = H_1(k(R_o||\widehat{ID_o}||w||11) + (1-k)(R_p||R_o||\widehat{ID_o}||\widehat{ID_p}||w||m||01))$$

is less than $2^{-(l_1-1)}$, unless they are asked during the attack. Hence, in what follows it is likely that queries

$$(1-k)(R_o||\widehat{ID_o}||w||11) + k(R_p||R_o||\widehat{ID_o}||\widehat{ID_p}||w||m||01)$$

and

$$k(R_o||\widehat{ID_o}||w||11) + (1-k)(R_p||R_o||\widehat{ID_o}||\widehat{ID_p}||w||m||01)$$

are asked during a successful attack. We assume that i_2 and i_1 are indices of these queries, respectively, and if these queries are never asked, we set $i_1 = \infty$ and $i_2 = \infty$.

We define set Υ as the set of successful pairs (ω, ρ), $\Upsilon = \{(\omega, \rho) | \ C_{A_{(1-k)II+kIII}}(\omega)$ produces a useful output & $(i_1, i_2) \ne (\infty, \infty)\}$. The lower bound of probability of producing a useful output is $\epsilon_1 = \Pr[\Upsilon] \ge \varepsilon - 2^{-(l_1-1)}$.

Since $C_{A_{(1-k)II+kIII}}$ makes query to the random oracle H_1 for Q_{i_1} and Q_{i_2} for a successful pair $(\omega, \rho) \in \Upsilon$, then we define set Υ_{i_1,i_2} as a subset of Υ in which query Q_{i_1} was made to the oracle H_1 before query Q_{i_2} which means $i_1 < i_2$.

This gives us a partition of Υ in exactly $\frac{(q_t)(q_t-1)}{2}$ classes. Let I be the set consisting of most likely indices (i_1, i_2), $I = \{(i_1, i_2) | \Pr[\Upsilon'_{i_1,i_2} | \Upsilon] \ge \frac{1}{2} \frac{2}{(q_t)(q_t-1)}\}$. Hence, for each $(i_1, i_2) \in I$, Υ_{i_1,i_2} is denoted as Υ'_{i_1,i_2}, we have $\Pr[\Upsilon'_{i_1,i_2}] = \Pr[\Upsilon'_{i_1,i_2} | \Upsilon] \Pr[\Upsilon] \ge \frac{\epsilon_1}{(q_t)(q_t-1)}$.

With splitting the randomness ρ related to the oracle H_1 as (ρ', c_k), where ρ' denotes answers of all queries to oracle H_1 except for query Q_{i_2}, whose answer is denoted as c_k (c_{1-k} is assigned to the response of Q_{i_1}). We employ splitting lemma, taking $X = (\omega, \rho')$, $Y = c_k$, $A = \Upsilon'_{i_1,i_2}$, $\delta = \frac{\epsilon_1}{(q_t)(q_t-1)}$ and $\alpha = \frac{\epsilon_1}{2(q_t)(q_t-1)}$. This lemma ensures the existence of a subset of executions Ω_{i_1,i_2} such that $\Pr[\Omega_{i_1,i_2} | \Upsilon'_{i_1,i_2}] \ge \frac{\alpha}{\delta} = \frac{1}{2}$ and for each $(\omega, \rho) \in \Omega_{i_1,i_2}$, $\Pr_{c'_k}[(\omega, \rho', c'_k) \in \Upsilon'_{i_1,i_2}] \ge \delta - \alpha = \frac{\epsilon_1}{2(q_t)(q_t-1)}$.

Since Υ'_{i_1,i_2} are disjoint, we have

$$
\begin{aligned}
\Pr_{(\omega,\rho)}[\exists(i_1,i_2) \in I \text{ s.t. } (\omega,\rho) \in \Omega_{i_1,i_2} \cap \Upsilon'_{i_1,i_2} \ | \Upsilon] &= \sum_{(i_1,i_2) \in I} \Pr[\Omega_{i_1,i_2} \cap \Upsilon'_{i_1,i_2} | \Upsilon] \\
&= \sum_{(i_1,i_2) \in I} \Pr[\Omega_{i_1,i_2} | \Upsilon'_{i_1,i_2}] \Pr[\Upsilon'_{i_1,i_2} | \Upsilon] \\
&\ge \frac{\sum_{(i_1,i_2) \in I} \Pr[\Upsilon'_{i_1,i_2} | \Upsilon]}{2} \ge \frac{1}{4}.
\end{aligned}
$$

Let $(\widetilde{i_1}, \widetilde{i_2})$ denote indices of a successful pair with probability at least $\frac{1}{4}$, $(\widetilde{i_1}, \widetilde{i_2}) \in I$ and $(\omega, \rho') \in \Omega_{i_1,i_2} \cap \Upsilon'_{i_1,i_2}$. If B replays the attack with fixed (ω, ρ') and a randomly chosen $c'_k \in \{0,1\}^{l_1}$, it gets another successful pair $((\omega, \rho'), c'_k)$ such that $c_k \ne c'_k$ with probability $\frac{\epsilon_1(1-2^{-l_1})}{2(q_t)(q_t-1)}$.

After two successful executions of $C_{A_{(1-k)II+kIII}}$, B obtains $((\omega, \rho'), c_k)$ and $((\omega, \rho'), c'_k)$, $c_k \neq c'_k$ which means that it obtains a useful pair $((R_p, R_o, s_p, c_k, c_{1-k}, x_o, x_p, m, w), (R_p, R_o, s'_p, c'_k, c_{1-k}, x_o, x_p, m, w))$ in time $t' \leq 2\tau$ with probability $\epsilon' \geq \frac{\epsilon_1^2(1-2^{-l_1})}{8(q_t)(q_t-1)}$, where $\epsilon_1 \geq \varepsilon - 2^{-(l_1-1)}$. Note that since query $Q_{\widetilde{i_1}}$ was made to the oracle H_1 before query $Q_{\widetilde{i_2}}$, we also have $c_{1-k} = c'_{1-k}$.

$\square$

Theorem 1. *If the RSA function associated to Kg_{rsa} is (t', ϵ')-one-way, then the proposed scheme is $(t, q_h, q_H, q_E, (1-k)q_d, kq_s, \epsilon)$-secure against the adversary $A_{(1-k)II+kIII}$ for a constant $k \in \{0,1\}$ such that*

$$\epsilon' \geq \frac{\epsilon_1^2(1-2^{-l_1})}{8(kq_s+(1-k)q_d+q_H)(kq_s+(1-k)q_d+q_H-1)},$$
$$t' \leq 2t + 2(1q_E + 1q_h + 2d(1-k)q_d + 2k(n+1)q_s)t_{exp}, \quad (4)$$

where $\epsilon_1 \geq (\frac{\epsilon}{4q_E} - 2^{-(l_1-1)} - (2(1-k)q_d^2 + 2kq_s^2 + ((1-k)q_d + kq_s)q_H)2^{-l_N})$, t_{exp}, l_1 and l_N are the time of one exponentiation in $\mathbb{Z}_N^*$ and two security parameters, respectively. In addition, q_H, q_h, q_E, q_d and q_s are the number of queries to oracles H_1, H, KeyExtract, DelegationGen and MProxyMSign, respectively.

Proof. In the proof, we consider two cases for the forgery depending on the type of adversaries. In the first case, without loss of generality it is assumed that there is one honest original signer (type II adversary plus $d-1$ corrupted original signers), while in the second one there is one honest proxy signer (type III adversary plus $n-1$ corrupted proxy signers). Then, we show that the algorithm B can solve a random instance of the RSA problem (N, e, y) such that $\gamma = y^{\frac{1}{e}} \mod N$.

Case 1. In this case, we consider adversaries of type II (i.e., $k = 0$). According to Lemma 1, $C_{A_{II}}$ returns a useful output $(R_p, R_o, s_p, c_0, c_1, x_o, x_p, m, w)$ in time bound $\tau = t + (1q_E + 1q_h + 2dq_d)t_{exp}$ with probability at least $\varepsilon = \frac{\epsilon}{4q_E} - (2q_d^2 + q_dq_H)2^{-l_N}$, where $x_o = \prod_{j=1}^d x_{o_j}$ and $x_p = \prod_{i=1}^n x_{p_i}$. Then, the algorithm B, the RSA solver, will produce a useful pair of $((R_p, R_o, s_p, c_0, c_1, x_o, x_p, m, w)$ and $(R_p, R_o, s'_p, c'_0, c_1, x_o, x_p, m, w))$ in time $t' \leq 2\tau$ with probability $\epsilon' \geq \frac{\epsilon_1^2(1-2^{-l_1})}{8q_t(q_t-1)}$, where $\epsilon_1 \geq (\varepsilon - 2^{-(l_1-1)})$ and $q_t = q_d + q_H$ (see Lemma 3). Since a useful pair contains two useful outputs, we have

$$R_p = s_p^e (x_p^e)^{-c_1} (R_o(x_o^e y)^{c_0})^{-n}$$

and

$$R_p = s'^e_p (x_p^e)^{-c_1} (R_o(x_o^e y)^{c'_0})^{-n}.$$

By dividing the two aforementioned equations, we have

$$\left(\frac{s_p}{s'_p}(x_o)^{n(c'_0-c_0)}\right)^e = y^{n(c_0-c'_0)}.$$

Since $c_0 \neq c'_0 \in \{0,1\}^{l_1}$, and e is a prime of length strictly greater than $l_1 + log_2^n$, we have $e > n(c_0-c'_0)$ and therefore $\gcd(e, n(c_0 - c'_0)) = 1$. Using the extended Euclidean algorithm, one can find $a, b \in \mathbb{Z}$ such that $ae + bn(c_0 - c'_0) = 1$. Hence, we have $y = y^{ae+bn(c_0-c'_0)} \mod N = (y^a(\frac{s_p}{s'_p}(x_o)^{n(c'_0-c_0)})^b)^e \mod N$. Therefore, B can output $y^a(\frac{s_p}{s'_p}(x_o)^{n(c'_0-c_0)})^b \mod N$ as the RSA inversion of y in time $t' \leq 2t+2(1q_E+1q_h+2dq_d)t_{exp}$ with probability $\epsilon' \geq \frac{\epsilon_1^2(1-2^{-l_1})}{8(q_d+q_H)(q_d+q_H-1)}$, where $\epsilon_1 \geq (\frac{\epsilon}{4q_E} - 2^{-(l_1-1)} - (2q_d^2 + q_dq_H)2^{-l_N})$. Note that, since A_{II} has secret keys of all proxy signers, there is no need to make any query to the MProxyMSign oracle.

Case 2. In this case, we consider adversaries of type III (i.e., $k = 1$). According to Lemma 1, $C_{A_{III}}$ returns a useful output $(R_p, R_o, s_p, c_0, c_1, x_o, x_p, m, w)$ in time bound $\tau = t + (1q_E + 1q_h + 2(n+1)q_s)t_{exp}$ with probability at least $\varepsilon = \frac{\epsilon}{4q_E} - (2q_s^2 + q_sq_H)2^{-l_N}$, where $x_o = \prod_{j=1}^d x_{o_j}$ and $x_p = \prod_{i=1}^n x_{p_i}$. Then, the algorithm B, the RSA solver, produces a useful pair of $((R_p, R_o, s_p, c_0, c_1, x_o, x_p, m, w)$ and $(R_p, R_o, s'_p, c_0, c'_1, x_o, x_p, m, w))$ in time $t' \leq 2\tau$ with probability $\epsilon' \geq \frac{\epsilon_1^2(1-2^{-l_1})}{8q_t(q_t-1)}$, where $\epsilon_1 \geq (\varepsilon - 2^{-(l_1-1)})$ and $q_t = q_s + q_H$ (see Lemma 3).
Since a useful pair contains two useful outputs, we have

$$R_p = s_p^e (x_p^e y)^{-c_1} (R_o(x_o^e)^{c_0})^{-n}$$

and

$$R_p = s_p'^e (x_p^e y)^{-c_1'} (R_o (x_o^e)^{c_0})^{-n}.$$

By dividing the two aforementioned equations, we have

$$(\frac{s_p}{s_p'} (x_p)^{(c_1' - c_1)})^e = y^{(c_1 - c_1')}.$$

Since $c_1 \neq c_1' \in \{0,1\}^{l_1}$, and e is a prime of length strictly greater than l_1, we have $e > (c_1 - c_1')$ and therefore $\gcd(e, (c_1 - c_1')) = 1$. Using the extended Euclidean algorithm, one can find $a, b \in \mathbb{Z}$ such that $ae + b(c_1 - c_1') = 1$. Hence, we have $y = y^{ae + b(c_1 - c_1')} \mod N = (y^a (\frac{s_p}{s_p'} (x_p)^{(c_1' - c_1)})^b)^e \mod N$. Therefore, algorithm B can output $y^a (\frac{s_p}{s_p'} (x_p)^{(c_1' - c_1)})^b \mod N$ as the RSA inversion of y in time $t' \leq 2t + 2(1 q_E + 1 q_h + 2(n+1) q_s) t_{exp}$ with probability $\epsilon' \geq \frac{\epsilon_1^2 (1 - 2^{-l_1})}{8(q_s + q_H)(q_s + q_H - 1)}$, where $\epsilon_1 \geq (\frac{\epsilon}{4 q_E} - 2^{-(l_1 - 1)} - (2 q_s^2 + q_s q_H) 2^{-l_N})$. Note that, since A_{III} has secret keys of all original signers, there is no need to make any query to the DelegationGen oracle. $\qquad\square$

5   Security extension to multiple honest signers

In general, one may consider two scenarios in the security analysis of our schemes for the single signer setup: security analysis with one honest signer and with multiple honest signers. At first glance, it seems that the latter is a stronger security model than the former, however, in what follows we show that for our schemes both of them are equivalent.

In the security proof with having multiple honest signers, assuming an adversary $A_{(1-k)II + kIII}$, with success probability at least ϵ and run-time t in returning a forgery containing $k n_{max} + (1 - k) d_{max}$ honest signers ($1 \leq d_{max} \leq d$ and $1 \leq n_{max} \leq n$), we need to estimate the run-time and a lower-bound for the success probability of a simulator in returning a useful output in terms of the run-time, the success probability of the adversary and the number of oracles' queries. Then, using the forking strategy, there exists an algorithm that can solve the RSA problem with a non-negligible probability, which is a function of the number of queries and the simulator's success probability.

In what follows, we examine if having multiple honest signers in the forgery has any effect on the number of oracle queries and the simulator's run-time.

In the single-signer setup, either one of the oracles DelegationGen or MProxyMSign simulates the role of one single honest signer, and the adversary plays the role of other co-signers. In the second scenario, since other co-signers are not necessarily assumed to be corrupted, the adversary has to play the role of other honest signers without knowing their secret keys. Since the adversary can engage in each of these oracles with any honest identity that it chooses, the adversary can interact concurrently by DelegationGen under each honest original signer's identity with the same original signers' identity set and the warrant to simulate the role of other co-original signers, or it can interact concurrently by the MProxyMSign under each honest proxy signer's identity with the same proxy signers' identity set, the warrant and the same message to simulate the role of other co-proxy signers.

We note that

– In both cases ($k = 0$ or $k = 1$), communication of the adversary with oracles (DelegationGen or MProxyMSign) is simulated in parallel (i.e. concurrent) time. Since the required time of signature simulation is the same as that in the first scenario, we have no extra penalty in the timing of returning a useful output.

– In both cases ($k = 0$ or $k = 1$), the result of making any number of queries to either one of the oracles DelegationGen or MProxyMSign on the same input under the identities of honest signers in the input to return a valid final output (a delegation or a proxy signature) to the adversary will give rise to at most one new entry in table $T_1[.]$, and consequently in the total number of queries. Since the number of queries are the same as those in the first scenario, we will have no extra penalty in the success probability of returning a useful output.

– Note that, the effect of having $kn_{max} + (1-k)d_{max}$ honest signers is only reflected in a useful output

$$(R_p, R_o, s_p, c_k, c_{1-k}, x_o, x_p, w, m, (1-k)d_{max} + kn_{max})$$

and consequently a useful pair

$$((R_p, R_o, s_p, c_k, c_{1-k}, x_o, x_p, m, w, (1-k)d_{max} + kn_{max}),$$
$$(R_p, R_o, s'_p, c'_k, c_{1-k}, x_o, x_p, m, w, (1-k)d_{max} + kn_{max}))$$

and parameters of the RSA problem $(e > l_1 + \log_2^n + k\log_2^{n_{max}} + (1-k)\log_2^{d_{max}})$.

It has essentially noting to do with the time and the success probability of returning a useful pair, since the simulator's probability and the total number of queries in the second scenario are the same as those in the first scenario, and with the RSA solutions, and one may verify them as follows.

A useful pair contains two useful outputs, we have

$$R_p = s_p^e (x_p^e y^{kn_{max}})^{-c_1} (R_o(x_o^e y^{(1-k)d_{max}})^{c_0})^{-n}$$

and

$$R_p = s_p'^e (x_p^e y^{kn_{max}})^{-(kc'_1 + (1-k)c_1)} (R_o(x_o^e y^{(1-k)d_{max}})^{(1-k)c'_0 + kc_0})^{-n}.$$

By dividing the two aforementioned equations, we have

$$\left(\frac{s_p}{s'_p}(x_p)^{((kc'_1+(1-k)c_1)-c_1)}(x_o)^{n((1-k)c'_0+kc_0-c_0)}\right)^e = y^{kn_{max}(c_1-(kc'_1+(1-k)c_1))+(1-k)d_{max}n(c_0-(1-k)c'_0-kc_0)}.$$

Since $kc_1 + (1-k)c_0 \neq kc'_1 + (1-k)c_0 \in \{0,1\}^{l_1}$, and e is a prime of length strictly greater than $l_1 + 2\log_2^n + \log_2^d$, $n_{max} \leq n$ and $d_{max} \leq d$, we have $e > k^2 n_{max}(c_1 - c'_1) + (1-k)^2 nd_{max}(c_0 - c'_0)$ and therefore $\gcd(e, k^2 n_{max}(c_1 - c'_1) + (1-k)^2 nd_{max}(c_0 - c'_0)) = 1$. Using the extended Euclidean algorithm, one can find $a, b \in \mathbb{Z}$ such that $ae + bk^2 n_{max}(c_1 - c'_1) + b(1-k)^2 nd_{max}(c_0 - c'_0) = 1$. Hence, we have

$$y = y^{ae+bk^2 n_{max}(c_1-c'_1)+b(1-k)^2 nd_{max}(c_0-c'_0)} \bmod N =$$

$$(y^a(\frac{s_p}{s'_p}(x_p)^{((kc'_1+(1-k)c_1)-c_1)}(x_o)^{n((1-k)c'_0+kc_0-c_0)})^b)^e \bmod N.$$

As a consequence, the RSA solutions are $y^a(\frac{s_p}{s'_p}(x_o)^{n(c'_0-c_0)})^b \bmod N$ and $y^a(\frac{s_p}{s'_p}(x_p)^{(c'_1-c_1)})^b \bmod N$ for $k = 0$ and $k = 1$, respectively.

6 Concluding remarks

In this section, first we discuss about the practicality of the scheme, and then compare it with existing schemes to show its efficiency.

Practicality. First, note that in our security proof, the probability estimates are far from being sharp. Hence, we expect higher security guarantees in real applications. However, to get a feeling about our main Theorem (Theorem 1), we analyze a real setup in what follows.

Considering adversaries of type III who can corrupt two users (i.e., $q_E = 2$ and $n = d = 3$), the security result becomes $\epsilon' \geq \frac{\epsilon^2(1-2^{-l_1})}{2^9(q_s+q_H)(q_s+q_H-1)}$. This result gives the exact relation between the security of the scheme and hardness of the RSA problem. This relation helps us choose the length of the keys to implement the scheme securely w.r.t. the best known algorithm which solves the RSA problem. The best way to solve the RSA problem is by factoring and the expected time to factor a l_N−bit integer with Number Field Sieve, NFS, is $O(\exp((\frac{64l_N}{9})^{\frac{1}{3}}(\ln l_N)^{\frac{2}{3}}))$. Consider an attacker that runs in time $t \simeq 2^{46}$ and makes at most 2^{46} random oracle and MProxyMSign queries, and can violate unforgeability of the scheme with probability at least 2^{-5}. According to Theorem 1, if such an adversary exists, the RSA problem can be solved in time $\frac{t'}{\epsilon'} \simeq \frac{2 \cdot 2^{46} \cdot 2^{92} \cdot 2^9}{2^{-10}} = 2^{158}$ which contradicts the best known result for factoring if the length of the RSA module is at least $2^{12} = 4096$ bits (the required time by NFS is 2^{167}). Therefore, to ensure that there is no adversary against the scheme with $t \simeq 2^{46}$, $\epsilon \simeq 2^{-5}$, $q_H \simeq 2^{46}$ and $q_s \simeq 2^{46}$, we should take the RSA module around 4096 bits which is a recommended key length.

Comparison. The comparison for some identity-based multi-proxy signature, proxy multi-signature and multi-proxy multi-signature schemes is summarized in Table 1. The comparison is in terms of DeleGen-Cost, DeleVer-Cost, PSign-Cost and PVer-Cost, dominating computational cost in delegation generation per an original signer, delegation verification per a verifier, proxy signature generation per a proxy signer and proxy signature verification per a verifier, respectively. In Table 1, P, E_T, m_{G_T}, m_G, E and MPS, PMS and MPMS denote the pairing evaluation, exponentiation in group G_T, multiplication in G_T, scalar multiplication in G, exponentiation in $\mathbb{Z}_N$, multi-proxy signature, proxy multi-signature and multi-proxy multi-signature, respectively. Note that, in comparison, it is assumed that other operations take zero time, n and d are the size of the proxy and original group, respectively, and also computational cost of the clerk is considered for users (original and proxy signers).

Scheme	DeleGen Cost	DeleVer Cost	PSign Cost	PVer Cost	Signature Size	Hard problem	Type of the scheme
Ours	$2E$	$2E$	$2nE$	$4E$	$3\mathbb{Z}_N^*$	RSA	MPS
[13, 29]	$2m_G$	$2P + 1m_{G_T}$	$n(4P + 1E_T + 2m_G)$	$4P$	$3G$	Pairing	MPS
Ours	$2dE$	$2E$	$2E$	$3E$	$3\mathbb{Z}_N^*$	RSA	PMS
[32]	$4m_G + (d-1)$ $(3P + 1m_G)$	$3P + 1m_{G_T}$ $+dm_G$	$4m_G$	$3P+$ $(d+1)m_G$	$3G$	Pairing	PMS
Ours	$2dE$	$2E$	$2nE$	$4E$	$3\mathbb{Z}_N^*$	RSA	MPMS
[36]	$(1m_{G_T}$ $+2P)$	$3P + 1m_{G_T}$ $(d-1) + 2m_G$	$(n-1)(5P + 1E_T$ $+3m_{G_T}) + 2m_G$	$5P + 3m_{G_T}$ $+1E_T$	$3G$	Pairing	MPMS

Table 1. Comparison between our schemes and some existing schemes

In Table 1, comparison of our schemes with some corresponding constructions, the latest provable secure schemes is given. As shown in Table 1, our identity-based multi-proxy signature, proxy multi-signature and multi-proxy multi-signature schemes are more efficient than identity-based multi-proxy signature schemes [28, 13, 29], identity-based proxy multi-signature schemes [28, 30–32] and identity-based multi-proxy multi-signature schemes [28, 34–36], respectively since the previous ones rely on elliptic curve pairings which are relatively expensive to implement (from [40], we know that the cost of each pairing is roughly that of $6 - 20$ exponentiations).

In addition, our schemes rely on well-understood assumption (RSA assumption), while previous schemes are based on recently derived computational assumptions.

Without incorporation bilinear pairings, two identity-based (multi)-proxy multi-signature schemes [33, 37] were proposed, while they are not secure in their security models as shown before, and also they are not efficient in terms of signature size.

7 Conclusion

In this paper, we showed that previous identity-based proxy multi- signature and multi-proxy multi-signature schemes without bilinear pairings are indeed insecure in their security models. Then, we presented the first provably secure identity-based multi-proxy multi-signature schemes from RSA assumption, without bilinear pairings, where identity-based multi-proxy signature and proxy multi-signature schemes are its special cases. To analyze security of our schemes, we proved a new Forking Lemma since the general Forking Lemma cannot be applied to them. These schemes are suitable for usage in resource-constraint devices to improve energy consumption which is a crucial factor for them.

References

1. Mambo, M., Usuda, K., and Okamoto, E. (1996) Proxy signatures: Delegation of the power to sign messages. *IEICE Transactions on Fundamentals of Electronics, Communications and Computer Sciences*, **79**, 1338–1354.

2. Shao, Z. (2009) Provably secure proxy-protected signature schemes based on RSA. *Computers & Electrical Engineering*, **35**, 497–505.

3. Shao, Z. (2003) Proxy signature schemes based on factoring. *Information Processing Letters*, **85**, 137–143.

4. Zhou, Y., Cao, Z., and Lu, R. (2005) Provably secure proxy-protected signature schemes based on factoring. *Applied Mathematics and Computation*, **164**, 83–98.

5. Park, J. H., Kang, B. G., and Han, J. W. (2005) Cryptanalysis of Zhou et al.'s proxy-protected signature schemes. *Applied Mathematics and Computation*, **169**, 192–197.

6. Liu, Y.-C., Wen, H.-A., Lin, C.-L., and Hwang, T. (2007) Proxy-protected signature secure against the undelegated proxy signature attack. *Computers & Electrical Engineering*, **33**, 177–185.

7. Hu, X., Xu, H., and Si, T. (2010) Analysis and improvement of proxy-protected signature secure against the undelegated proxy signature attack. *Computational Information Systems*, **6**, 2997–3002.

8. Gu, C. and Zhu, Y. (2005) Provable security of ID-based proxy signature schemes. *Proc. of the* 3*rd Int. Conf. on Networking and Mobile Computing (ICCNMC* 2005*)*, Zhangjiajie, China, 2-4 August, pp. 1277–1286. Springer-Verlag, Berlin.

9. Zhang, J. and Zou, W. (2007) Another ID-based proxy signature scheme and its extension. *Wuhan University Journal of Natural Sciences*, **12**, 33–36.

10. Wu, W., Mu, Y., Susilo, W., Seberry, J., and Huang, X. (2007) Identity-based proxy signature from pairings. *Proc. of the* 4*th Int. Conf. on Autonomic and Trusted Computing*, Hong Kong, China, 11-13 July, pp. 22–31. Springer-Verlag, Berlin.

11. Gu, C. and Zhu, Y. (2008) An efficient ID-based proxy signature scheme from pairings. *Proc. of* 3*rd SKLOIS Conf. on Information Security and Cryptology (Inscrypt* 2007*)*, Xining, China, 31 August- 5 September, pp. 40–50. Springer-Verlag, Berlin.

12. Ji, H., Wang, Y., Han, W., and Zhao, L. (2009) An identity-based proxy signature from bilinear pairings. *WASE Int. Conf. on Information Engineering (ICIE* 2009*)*, Taiyuan, Shanxi, 10-11 July, pp. 14–17. IEEE Xplore, NY.

13. Cao, F. and Cao, Z. (2009) A secure identity-based multi-proxy signature scheme. *Computers & Electrical Engineering*, **35**, 86–95.

14. Xu, J., Zhang, Z., and Feng, D. (2005) ID-based proxy signature using bilinear pairings. *Proc. of Parallel and Distributed Processing and Applications-ISPA* 2005 *Workshops*, Nanjing, China, 2-5 November, pp. 359–367. Springer-Verlag, Berlin.

15. Shim, K. (2006) An identity-based proxy signature scheme from pairings. *Proc. of* 8*th Int. Conf. on Information and Communications Security (ICICS* 2006*)*, Raleigh, NC, USA, 4-7 December, pp. 60–71. Springer-Verlag, Berlin.

16. Lu, R. and Cao, Z. (2005) Designated verifier proxy signature scheme with message recovery. *Applied Mathematics and Computation*, **169**, 1237–1246.

17. Yu, Y., Xu, C., Zhang, X., and Liao, Y. (2009) Designated verifier proxy signature scheme without random oracles. *Computers & Mathematics with Applications*, **57**, 1352–1364.

18. Shim, K.-A. (2011) Short designated verifier proxy signatures. *Computers & Electrical Engineering*, **37**, 180–186.

19. Huang, X., Mu, Y., Susilo, W., Zhang, F., and Chen, X. (2005) A short proxy signature scheme: efficient authentication in the ubiquitous world. *Proc. of Embedded and Ubiquitous Computing–EUC* 2005 *Workshops*, Nagasaki, Japan, 6-9 December, pp. 480–489. Springer-Verlag, Berlin.

20. Zhang, J., Liu, C., and Yang, Y. (2010) An efficient secure proxy verifiably encrypted signature scheme. *Journal of Network and Computer Applications*, **33**, 29–34.

21. Huang, X., Susilo, W., Mu, Y., and Wu, W. (2006) Proxy signature without random oracles. *Proc. of* 2*nd Int. Conf. on Mobile Ad-hoc and Sensor Networks (MSN* 2006*)*, Hong Kong, China, 13-15 December, pp. 473–484. Springer-Verlag, Berlin.

22. Hwang, S. and Shi, C. (2000) A simple multi-proxy signature scheme for electronic commerce. *Proc. of the* 10*th National Conf. on Information Security*, Hualien, Taiwan, China, 15-18 March, pp. 57–67. Springer-Verlag, Berlin.

23. Li, X., Chen, K., and Li, S. (2005) Multi-proxy signature and proxy multi-signature schemes from bilinear pairings. *Proc. of* 5*th Int. Conf. on Parallel and Distributed Computing: Applications and Technologies (PDCAT* 2005*)*, Singapore, Singapore, 8-10 December, pp. 61–62. Springer-Verlag, Berlin.

24. Wang, Q., Cao, Z., and Wang, S. (2005) Formalized security model of multi-proxy signature schemes. *Proc. of the* 5*th Int. Conf. on Computer and Information Technology (CIT* 2005*)*, Shanghai, China, 21-23 September, pp. 668–672. IEEE Xplore, NY.

25. Hwang, S.-J. and Chen, C.-C. (2004) New multi-proxy multi-signature schemes. *Applied Mathematics and Computation*, **147**, 57–67.

26. Guo, L. and Wang, G. (2007) Insider attacks on multi-proxy multi-signature schemes. *Computers & Electrical Engineering*, **33**, 88–93.

27. Chen, X., Zhang, F., and Kim, K. (2003) ID-based multi-proxy signature and blind multisignature from bilinear pairings. *Proc. of* 6*th Int. Conf. of Korea Institute on Information Security and Cryptology (KIISC* 2003 *)*, Seoul, Korea, 27-28 November, pp. 11–19. Springer-Verlag, Berlin.

28. Li, X. and Chen, K. (2005) ID-based multi-proxy signature, proxy multi-signature and multi-proxy multi-signature schemes from bilinear pairings. *Applied Mathematics and Computation*, **169**, 437–450.

29. Xiong, H., Hu, J., Chen, Z., and Li, F. (2011) On the security of an identity based multi-proxy signature scheme. *Computers & Electrical Engineering*, **37**, 129–135.

30. Wang, Q. and Cao, Z. (2007) Identity based proxy multi-signature. *Journal of Systems and Software*, **80**, 1023–1029.

31. Cao, F. and Cao, Z. (2009) A secure identity-based proxy multi-signature scheme. *Information Sciences*, **179**, 292–302.

32. Shao, Z. (2009) Improvement of identity-based proxy multi-signature scheme. *Journal of Systems and Software*, **82**, 794–800.

33. Tiwari, N. and Padhye, S. (2011) An ID-based proxy multi signature scheme without bilinear pairings. *Proc. of the First Int. Conf. on Security Aspects in Information Technology (InfoSecHiComNet* 2011*)*, Haldia, India, 19-22 October, pp. 83–92. Springer-Verlag, Berlin.

34. Guo, S., Cao, Z., and Lu, R. (2006) An efficient ID-based multi-proxy multi-signature scheme. *Proc. of the 1st Int. Multi-Symp. on Computer and Computational Sciences (IMSCCS* 2006*)*, Hangzhou, China, 20-24 June, pp. 81–88. IEEE Xplore, NY.

35. Sahu, R. and Padhye, S. (2010) An ID-based multi-proxy multi-signature scheme. *Proc. of Int. Conf. on Computer and Communication Technology (ICCCT* 2010*)*, Allahabad, Uttar Pradesh, 17-19 September, pp. 60–63. IEEE Xplore, NY.

36. Sahu, R. A. and Padhye, S. (2011) Efficient ID-based multi-proxy multi-signature scheme based on CDHP. *Journal of Applied Mathemtics and Informatics*, **5**, 275–282.

37. Tiwari, N., Padhye, S., and He, D. (2013) Efficient ID-based multiproxy multisignature without bilinear maps in ROM. *Annals of Telecommunications - Annales des tlcommunications*, **68**, 231–237.

38. Shamir, A. (1985) Identity-based cryptosystems and signature schemes. *Proc. of* 4*th Annual Int. Cryptology Conf. on Advances in Cryptology-CRYPTO* 1984, Santa Barbara, CA, USA, 19-22 August, pp. 47–53. Springer-Verlag, Berlin.

39. Guillou, L. and Quisquater, J. (1990) A paradoxical identity-based signature scheme resulting from zero-knowledge. *Proc. of* 8*th Annual Int. Cryptology Conf. on Advances in Cryptology-CRYPTO* 1988, Santa Barbara, CA, USA, 21-25 August, pp. 216–231. Springer-Verlag, Berlin.

40. Bellare, M. and Neven, G. (2006) Identity-based multi-signatures from RSA. *Proc. of the* 7*th Cryptographers' Track at the RSA Conf. on Topics in Cryptology (Topics in Cryptology–CT-RSA* 2007*)*, San Francisco, CA, USA, 5-9 February, pp. 145–162. Springer-Verlag, Berlin.

41. Bellare, M. and Rogaway, P. (1993) Random oracles are practical: A paradigm for designing efficient protocols. *Proc. of the* 1*st ACM Conf. on Computer and Communications Security (CCS* 1993*)*, Fairfax, VA, USA, 3-5 November, pp. 62–73. ACM, New York, NY.

42. Bellare, M. and Neven, G. (2006) Multi-signatures in the plain public-key model and a general forking lemma. *Proc. of the* 13*th ACM Conf. on Computer and Communications Security (CCS* 2006*)*, Alexandria, VA, USA, 30 October-3 November, pp. 390–399. ACM, New York, NY.

43. Coron, J. (2000) On the exact security of full domain hash. *Proc. of the* 20*th Annual Int. Cryptology Conf. on Advances in Cryptology-CRYPTO* 2000, Santa Barbara, CA, USA, 20-24 August, pp. 229–235. Springer-Verlag, Berlin.

44. Pointcheval, D. and Stern, J. (2000) Security arguments for digital signatures and blind signatures. *Journal of Cryptology*, **13**, 361–396.