1-out-of-n Signatures from a Variety of Keys

Masayuki Abe[1], Miyako Ohkubo[2], and Koutarou Suzuki[1]

[1] NTT Laboratories
1-1 Hikari-no-oka, Yokosuka-shi, Kanagawa-ken, 239-0847 Japan
[2] Chuo University
1-13-27, Kasuga, Bunkyo-ku, Tokyo 112-8551 JAPAN

Abstract. This paper addresses how to use public-keys of several different signature schemes to generate 1-out-of-n signatures. Previously known constructions are for either RSA-keys only or DL-type keys only. We present a widely applicable method to construct a 1-out-of-n signature scheme that allows mixture use of different flavors of keys at the same time. The resulting scheme is more efficient than previous schemes even if it is used only with a single type of keys. With all DL-type keys, it yields shorter signatures than the ones of the previously known scheme based on the witness indistinguishable proofs by Cramer, et. al. With all RSA-type keys, it reduces both computational and storage costs compared to that of the Ring signatures by Rivest, et. al.

1 Introduction

A 1-out-of-n signature convinces a verifier that a document is signed by one of n possible independent signers without allowing the verifier to identify which signer it was. It can be seen as a simple group signature that has no group manager who can revoke the identity of the signer in case of emergency. Such a signature can also be seen as a kind of non-interactive proof that the signer owns a witness (secret-key) that corresponds to one of n commitments (public-keys) or theorems without leaking which one it really is. Such a primitive, as a signature scheme and/or a proof system, plays a central role in variety of applications such as group signatures [8, 5], designated verifier signatures [17], mix-nets [1], electronic voting [10, 11] and so on.

In [9], Cramer, Damgård and Shoenmakers presented a widely applicable yet efficient construction of t-out-of-n witness indistinguishable proofs [13] based on secret sharing and public-coin honest verifier zero-knowledge proofs. It can be transformed into t-out-of-n signatures via the Fiat-Shamir technique [12]. It is especially suitable for converting Schnorr signatures [23] and Guillou-Quisquater signatures [16] into t-out-of-n signatures. It also allows to involve RSA signature scheme based on a zero-knowledge proof of knowledge about the factors of RSA modulus, e.g. [7, 6], but they are less efficient than the Schnorr or the GQ signatures both in computation and storage. [22] offers more intricate construction of t-out-of-n proofs for membership.

In [21], an efficient construction of 1-out-of-n signatures with RSA public-keys was introduced by Rivest, Shamir and Tauman. Called the Ring Signature

Scheme, it is based on trapdoor one-way permutations (TPs for short) and an ideal block cipher that is regarded as a perfectly random permutation. The name reflects its unique structure such that a signer who knows at least one witness (trapdoor information) can connect the head and tail of a series of n random permutations to shape the sequence into a ring. Since the trapdoor is essential in their construction, it is only for the keys like RSA's and the discrete-log keys are not supported.

There are other solutions that are more efficient but work only in non-separable models where all public-keys are related. For instance, when public-keys of the Schnorr signature scheme are chosen from a common group, one can construct an efficient 1-out-of-n signature scheme as shown in Appendix A. Such non-separable but highly efficient schemes may be useful when used within a specific members. In general, however, public-keys are selected independently by each signer. Even key-length would differ from user to user. Constructions based on [9] and [21] suit a separable model where no underlying group are assumed. Hence, they are 'setup-free'; if one utilizes an existing public-key infrastructure, the key-setup phase only for this purpose is unnecessary. Furthermore, each key can be freely updated whenever each user wishes.

As introduced in [21], one application of 1-out-of-n signatures is to involve somebody else's public-keys into one's signature without their agreement. Although there are pros and cons for such usage, it is surely useful for protecting privacy. Unfortunately, all above mentioned known schemes have particular shortcomings for this purpose; What if one is using a DL type public-key while others are using RSA? Generating a new RSA key only for this purpose is not a great idea. It is important to have wide freedom for choosing various public-keys to involve.

Our contribution. We present a widely applicable method of constructing 1-out-of-n signature schemes that allows to use several flavours of public-keys such as these for integer factoring based schemes and discrete-log based schemes at the same time. We describe two classes of signature schemes, which we call trapdoor-one-way type and three-move type, whose public-keys can be used for our construction.

Our approach also has several advantages even for the use with the same kind of keys like the former schemes:

- When our scheme is used only with public-keys of three-move type signature schemes converted from zero-knowledge proof system, it results in a more efficient scheme than previously known three-move based construction [9] with regard to the size of signatures. For large n, it saves signature length about *by half*. Since this type of schemes includes the discrete-log based public-keys, this can be seen as the first construction of a ring signature scheme based on the discrete logarithm problem.
- When our scheme is used only with the trapdoor-one way based public-keys such as RSA, it results in a simplified ring signature scheme. By eliminating the use of block cipher and costly domain adjustment from the former

scheme [21], our scheme offers shorter signature and less computation. In particular,

- The signature size of ours is about 20% less than that of the previous construction when RSA with modulus size 1024bits are used.
- Both size of signature and computation in our signature generation is proportional to the *average* size of the modulus while that of former scheme it is proportional to the *maximum* size of the modulus. Accordingly, one long modulus does not impact efficiency in our scheme unlike the previous scheme.

We will show several concrete examples following an abstract construction. The security is proven in the random oracle model [3] as well as previously known schemes.

The rest of this paper is organized as follows. Section 2 defines security of 1-out-of-n signatures. We review two constructions that work in the separable model in Section 3. Section 4 describes our construction in an abstract way. Some concrete examples are given in Section 5. It includes a discrete-log version of the ring signature scheme, improved and simplified version of the RSA-based ring signatures, and small case of mixture use of RSA and DL type signatures. In Section 6 the efficiency of some concrete instantiations are analyzed in detail.

2 Security Definitions

We first of all define 1-out-of-n signature scheme as follows.

Definition 1. (Syntax) *A 1-out-of-n signature scheme,* $\mathbf{S}^{\mathsf{1,n}}$*, is a triple of polynomial-time algorithms,* $\mathbf{S}^{\mathsf{1,n}} = (\mathcal{G}^{\mathsf{1,n}}, \mathcal{S}^{\mathsf{1,n}}, \mathcal{V}^{\mathsf{1,n}})$*:*

$(sk, vk) \leftarrow \mathcal{G}^{\mathsf{1,n}}(1^\kappa)$ *A probabilistic algorithm that takes security parameter* κ *and outputs private key sk and public-key vk.*

$\sigma \leftarrow \mathcal{S}^{\mathsf{1,n}}_{sk}(m, L)$ *A (probabilistic) algorithm that takes message* m*, and a list, say* L*, of public-keys including the one that corresponds to sk, outputs signature* σ*.*

$1/0 \leftarrow \mathcal{V}^{\mathsf{1,n}}_L(m, \sigma)$ *An algorithm that takes message* m *and signature* σ*, and outputs either* 1 *or* 0 *meaning* accept *and* reject*, respectively. We require that* $\mathcal{V}^{\mathsf{1,n}}_L(m, \mathcal{S}^{\mathsf{sig}}_{sk}(m, L)) = 1$ *for any message* m*, any* (sk, vk) *generated by* $\mathcal{G}^{\mathsf{1,n}}$*, and any* L *that includes vk.*

Note that $\mathcal{G}^{\mathsf{1,n}}$ does not generate L but each key pairs. Therefore, if L includes public-keys based on different security parameters, the security of $\mathbf{S}^{\mathsf{1,n}}$ is set to the smallest one among them. As we will see, L can include several types of public-keys all at the same time such as for RSA and Schnorr in a particular construction. $\mathcal{G}^{\mathsf{1,n}}$ may be extended to take a description of the key-type to support such variety flavour of key pairs. By $|L|$, we denote the number of public-keys in L hereafter.

The security of 1-out-of-n signature schemes has two aspects: *Signer ambiguity* and *Unforgeability*. Informally, the signer ambiguity is that it is infeasible to identify which signing key is used to generate a signature.

Definition 2. (Signer Ambiguity) *Let* $L = \{vk_1, \ldots, vk_n\}$ *where each key is generated as* $(vk_i, sk_i) \leftarrow \mathcal{G}^{1,n}(1^{\kappa_i})$. $\mathbf{S}^{1,n}$ *is perfectly signer-ambiguous if, for any* L*, any message* m*, and any* σ *generated by* $\sigma \leftarrow \mathcal{S}^{1,n}_{sk}(m, L)$ *where* $sk \leftarrow \{sk_1, \ldots, sk_n\}$*, given* (L, m, σ)*, any unbound adversary* $\mathcal{A}$ *outputs* i *such that* $sk = sk_i$ *with probability exactly* $1/|L|$.

Here, $a \leftarrow b$ denotes a uniform choice of an element from set b and its assignment to a. It is important to see that unbound adversary can compute all private keys from L. In practice, it means that when each public-key is owned by an independent party, they remain uncertain who else has issued a signature involving their public-keys.

Unforgeability of 1-out-of-n signature scheme is defined by naturally extending the notion of existential unforgeability against adaptive chosen message attacks (EUF-CMA) [15], which is the strongest security for ordinary signature schemes. In chosen message attacks, an adversary is given unbound access to the signing oracle and allowed to ask signatures for arbitrary messages. To adapt to our situation, we further allow the adversary to choose arbitrary set of public-keys as a subset of initially considered set of public-keys every time it access to the signing oracle. It is stressed that one can generate 1-out-of-n signatures for any message and any list of public-keys as long as it includes one's own key. So the definition of unforgeability should not treat "append-your-own-key-then-forge" activities as a forgery. Formal definition is as follows.

Definition 3. (Existential Unforgeability against Adaptive Chosen Message and Chosen Public-key Attacks) *Let* $(vk_i, sk_i) \leftarrow \mathcal{G}^{1,n}(1^{\kappa_i})$ *for* $i = 1, \ldots, n$. *Let* $\kappa = \min(\kappa_1, \ldots, \kappa_n)$ *and* $\mathcal{L} = \{vk_1, \ldots, vk_n\}$. *Let* $\mathcal{SO}^{1,n}_{\mathcal{L}}(m_i, L_i)$ *be a signing oracle that takes any message* $m \in \{0,1\}^*$ *and any* $L_i \subseteq \mathcal{L}$ *and outputs a valid signature* σ_i *that satisfies* $\mathcal{V}^{1,n}_{L_i}(m_i, \sigma_i) = 1$. *We say* $\mathbf{S}^{1,n}$ *is existentially unforgeable against adaptive chosen message and chosen public-key attacks if, for any polynomial-time oracle machine* $\mathcal{A}$ *such that* $(L, m, \sigma) \leftarrow \mathcal{A}^{\mathcal{SO}^{1,n}_{\mathcal{L}}(\cdot,\cdot)}(\mathcal{L})$*, its output satisfies* $\mathcal{V}^{1,n}_{L}(m, \sigma) = 1$ *only with negligible probability in* κ. *Restriction is that* $L \subseteq \mathcal{L}$ *and* $(L, m, \sigma) \notin \{(L_i, m_i, \sigma_i)\}$ *where* $\{(L_i, m_i, \sigma_i)\}$ *is the set of oracle queries and replies between* $\mathcal{A}$ *and* $\mathcal{SO}^{1,n}_{\mathcal{L}}$.

The above definition is a seamless extension of EUF-CMA since the case of $n = 1$ is equivalent to that. Note that the size of $\mathcal{L}$ can be a security parameter as well, though it is not our case. It is important to see that the above definition states that the list of public-keys must not be altered as well as the message. That is, one should not be able to add or remove public-keys associated to given signatures.

3 Previous Schemes

3.1 Witness Indistinguishable Signatures [9]

Here we review the witness indistinguishable signatures from [9] with a concrete discrete logarithm setting. Let p_i, q_i be large primes. Let $\langle g_i \rangle$ denote a prime subgroup of $\mathbb{Z}_{p_i}^*$ generated by g_i whose order is q_i. Let x_i, y_i be $y_i = {g_i}^{x_i} \bmod p_i$. Here, x_i is the secret-key and (y_i, p_i, q_i, g_i) is the public-key. Let L be a set of (y_i, p_i, q_i, g_i) for $i = 0, \ldots, n-1$. Let $H : \{0,1\}^* \to \{0,1\}^\ell$ be a publicly available hash function, where ℓ is larger than the largest $|q_i|$.

A signer who owns secret key x_k generates a signature for message m with public-key list L that includes his own public-key, in the following way.

W-1 (Simulation step)**:** For $i = 0, \ldots, n-1$, $i \neq k$, select $s_i, c_i \leftarrow \mathbb{Z}_{q_i}$ and compute $z_i = g_i^{s_i} y_i^{c_i} \bmod p_i$.

W-2 (Real proof step)**:** Select $r_k \leftarrow Z_{q_k}$ and compute

$$z_k = {g_k}^{r_k} \bmod p_k$$
$$c = H(L, m, z_0, \cdots, z_{n-1})$$
$$c_k = c \oplus (c_0 \oplus \cdots \oplus c_{k-1} \oplus c_{k+1} \oplus \cdots \oplus c_{n-1}) \text{ ($\oplus$: bitwise-XOR.)}$$
$$s_k = r_k - c_k \cdot x_k \bmod q_k.$$

The resulting signature is $\sigma = (c_0, s_0, \ldots, c_{n-1}, s_{n-1})$. A (L, m, σ) is valid if

$$c_0 \oplus \cdots \oplus c_{n-1} = H(L, m, g_0^{s_0} y_0^{c_0} \bmod p_0, \cdots, g_{n-1}^{s_{n-1}} y_{n-1}^{c_{n-1}} \bmod p_{n-1}).$$

The size of σ is $n\ell + \sum_{i=0}^{n-1} |q_i|$ bits. L does not necessarily contain whole public-keys but some identifiers of the keys. The security can be proven in the random oracle model by using the rewinding simulation technique [14, 20, 19].

3.2 Ring Signatures with trapdoor one-way permutations [21]

Let $f_i : \{0,1\}^\ell \to \{0,1\}^\ell$ be a trapdoor one-way permutation where its inverse, f_i^{-1}, can be computed only if the trapdoor information is known. Let E, D be a symmetric-key encryption and decryption function whose message space is $\{0,1\}^\ell$. Let H be a hash function whose output domain matches to the key-space of E, D.

Given $f_0, \ldots, f_{n-1}$, the signer who can compute ${f_k}^{-1}$ generates a signature, for message m in the following way.

R-1 (Initialization): Compute $r_{n-1} = D_K(c_0)$ where $K = H(m)$ and $c_0 \leftarrow \{0,1\}^\ell$.

R-2 (Forward sequence): For $i = 0, \ldots, k-1$, compute $c_{i+1} = E_K(c_i \oplus f_i(s_i))$ for $s_i \leftarrow \{0,1\}^\ell$.

R-3 (Backward sequence): For $i = n-1, \ldots, k+1$, compute $r_{i-1} = D_K(r_i \oplus f_i(s_i))$ for $s_i \leftarrow \{0,1\}^\ell$.

R-4 (Shaping into a ring): Compute $s_k = f_k^{-1}(c_k \oplus r_k)$

The resulting signature is $(c_0, s_0, s_1, \ldots, s_{n-1})$. A signature-message pair is verified by computing $K = H(m)$ and $c_{i+1} = E_K(c_i \oplus f_i(s_i))$ for $i = 0, \ldots, n-1$, and accept if $c_n = c_0$ holds.

In practice, each trapdoor permutations will be defined over individual domain such as $\mathbb{Z}_{N_i}$. In such a case, the above scheme need to transform such individual functions into common-domain trapdoor permutations. This transformation incurs some overhead. The following method is suggested in [21] to transform $\mathcal{E}_i$ into f_i defined over common-domain $\{0,1\}^\ell$ where $\ell = \max\{|N_i|\} + 160$. Let $\mathcal{E}_i$ be the RSA encryption function with modulus N_i. Let Q and S be positive integers such that $QN_i + S = s$ and $0 \leq S < N_i$. Define

$$f_i(s) = \begin{cases} QN_i + \mathcal{E}_i(S) & \text{if } (Q+1)N_i \leq 2^\ell \\ s & \text{otherwise.} \end{cases}$$

In order for the latter case to happen only with negligible probability, ℓ should be polynomially larger than the size of largest modulus. For instance, if the largest modulus is 2048 bits, ℓ will be $2048 + 160$ bits. Accordingly, the resulting signature size is $2208(n+1)$ bits. This would be a large overhead when other moduli are all 1024bits.

The above ring signature is existentially unforgeable against adaptive chosen message attacks in the ideal cipher model where E and D are modeled by truly random permutations.

3.3 Other related works

[4] extends the scheme of [21] to a threshold scheme with the cost of $O(2^t \log n)$ efficiency for threshold t. [18] considers deniable ring authentication that accepts variety of public-keys and a threshold of signers. It however, needs interaction between the signer and the verifier.

4 Our Scheme

4.1 Type of Keys and Signature Schemes

This section describes signature schemes whose public-key can be used to our construction of 1-out-of-n signature scheme. Let $\mathbf{S} = (\mathcal{G}^{\mathsf{sig}}, \mathcal{S}^{\mathsf{sig}}, \mathcal{V}^{\mathsf{sig}})$ be a signature scheme. We require that underlying signature scheme be secure (existentially unforgeable) against adaptive chosen message attacks. For this to be achieved, it must be at least hard to compute *sk* from *vk*.

We consider two types of signature schemes which we call *Hash-then-One-Way type* (type-H) and *Three-move type* (type-T) in the rest of this paper.

A representative of type-H is the Full-domain RSA signature scheme. Let F be a trapdoor one-way function and I be its inverse function. For any c taken from appropreate domain, computing $s = F_{vk}(c)$ is easy but any preimage of s cannot be computed in polynomial-time. Trapdoor information *sk* allows one to efficiently compute one of the pre-images of s. The signing function $\mathcal{S}^{\mathsf{sig}}$ involves I and a hash function $H : \{0,1\}^* \to \Delta$ that hashes message m and auxiliary information if any. Domain Δ is assumed to be an abelian group such as modulo an RSA composite that depends on the particular detail of the signature scheme and the security parameter. H can be a composition of hash functions. The verification function $\mathcal{V}^{\mathsf{sig}}$ of type-H consists of F and H. H is the same as that in $\mathcal{S}^{\mathsf{sig}}$. By F, signature σ is transformed into an element of Δ so that the result can be compared with the hashed message. In summary, type-H is as follows.

HASH-AND-ONE-WAY TYPE

SIGNING	VERIFICATION
$\mathcal{S}^{\mathsf{sig}}_{sk}(m) =$	$\mathcal{V}^{\mathsf{sig}}_{vk}(m, \sigma) =$
$c = H(m, aux)$ $s = I_{sk}(c)$ Return $\sigma = (s, aux)$	$\sigma \xrightarrow{p} (s, aux)$ ($\xrightarrow{p}$: parsing) $c = H(m, aux)$ $e = F_{vk}(s)$ Return 1 if $c = e$. Otherwise, 0.

The security of type-H requires that computing $I_{sk}(c)$ without sk be intractable. Precise description is as follows.

Assumption 1 (Intractability of Computing I) *For any probabilistic poly-time algorithm $\mathcal{A}$, for $(vk, sk) \leftarrow \mathcal{G}^{\mathsf{sig}}(1^\kappa)$, and for $c \leftarrow \Delta$, $F_{vk}(\mathcal{A}(c, vk)) = c$ happens only with negligible probability in κ. Probability is taken over coin flips of $\mathcal{A}$, $\mathcal{G}^{\mathsf{sig}}$, and the choice of c.*

To prevent $\mathcal{A}$ from being successful by random guess, I must not shrink Δ into exponentially small domain. Typically, I is one-to-one with regard to variable c. Finally, note that the above intractability assumption for computing I is stronger than that of for computing sk only from vk.

Next we describe type-T schemes. As the name implies, this type of schemes are from three-move honest verifier zero-knowledge proofs. Classical Fiat-Shamir signature scheme belongs to this type. Here, signing function $\mathcal{S}^{\mathsf{sig}}$ involves three functions, say A, H, and Z used in each stage of three-move honest verifier zero-knowledge proof system. A generates the first-move commitment a and with regard to randomness r. H is a hash function $\{0,1\}^* \to \Delta$ used to generate a challenge string c from message m and commitment a. Z is an answer generation function that generates an answer, say s, to the challenge. The verification function $\mathcal{V}^{\mathsf{sig}}$ involves two functions V and H. V is a checking predicate of the embedded zero-knowledge proof system. It converts s and c into z which is supposed to equal to a. If it is the case, hashing z with message m by using H in the same way as in signing procedure outputs e that matches to c. Abstract description follows.

Three-move Type

Signing	Verification
$\mathcal{S}^{\mathsf{sig}}_{sk}(m) =$	$\mathcal{V}^{\mathsf{sig}}_{vk}(m,\sigma) =$
$a \leftarrow A(sk;r)$	$\sigma \xrightarrow{p} (s,c)$
$c = H(m,a)$	$z = V(s,c,vk)$
$s = Z(sk,r,c)$	$e = H(m,z)$
Return $\sigma = (s,c)$	Return 1 if $c = e$. Otherwise 0.

The following property is assumed to type-T schemes.

Definition 4. (Collision Property) *There exists a polynomial-time algorithm that computes sk from* (c,s,c',s') *and vk where* (c,s) *and* (c',s') *are two unequal valid signatures that correspond to the same* (a,m) *given to hash function* H*.*

This property is frequently used for proving the security of type-T schemes such as Schnorr signatures and Modified ElGamal signatures and vast number of their variants.

Regardless of the types, we require that the signature scheme is simulatable in a particular model. Intuitively, it must be possible to construct a simulator that simulates the signing oracle without the signing key. In many schemes, this property is achieved in the random oracle model. Precise definition of this property is as follows.

Definition 5. (Simulatability of S in the random oracle model) *A signature scheme,* $\mathbf{S}$*, is* (t,ϵ,q_s,q_h)*-simulatable in the random oracle model if for any key-pair* (sk,vk) *generated by* $\mathcal{G}^{\mathsf{sig}}(1^\kappa)$ *and for any algorithm* $\mathcal{A}$ *that refers random oracle* H *at most* q_h *times and* $\mathcal{S}^{sig}_{sk}$ *at most* q_s *times, there exists a pair of interactive machines,* $\mathcal{M}^{sim} = (\mathcal{S}^{sim}, H^{sim})$*, that interacts with* $\mathcal{A}$ *in such a way that the total running time is at most* t*, and statistical distance of the probability distribution of* $\mathsf{view}_{\mathcal{A}}(vk,\mathcal{S}^{sig}_{sk},H)$ *and* $\mathsf{view}_{\mathcal{A}}(vk,\mathcal{M}^{sim}_{vk})$ *is at most* 2ϵ*. Here, the probability is taken over all coin flips of* $\mathcal{G}^{\mathsf{sig}}$*,* $\mathcal{S}^{\mathsf{sig}}$*,* H*,* $\mathcal{M}^{sim}$*, and* $\mathcal{A}$*.*

The above definition can be generalized to deal with multiple oracles for signature schemes that involves multiple hash functions if necessary. Simulatability is featured in many practical EUF-CMA signature schemes such as the Schnorr signature scheme, and FDH-RSA scheme. Given this property, one can say that an event that happens with probability μ in the real run also happens with probability at least $\mu - \epsilon$ in the simulation.

4.2 Description

[Key Generation]

A signer generates his own key pairs by using the signature generation function of a signature scheme of his choice: $(sk, vk) \leftarrow \mathcal{G}^{\mathsf{sig}}(1^\kappa)$

[Public-key Listing]

Collect public-keys and list them in L. Then, insert vk to the randomly chosen position of the list. Let $L = \{vk_0, \ldots, vk_{n-1}\}$ where $vk_k = vk$ for some $k \in \{0, \ldots, n-1\}$. (Corresponding signing key sk is referred as sk_k hereafter.)

For $a, b \in \Delta_i$, let $a + b$ denote the group operation of abelian group Δ_i and $a - b$ be the group operation with inverse of b. These binary operators are used without subscripts that denotes each group. Let $H_i : \{0,1\}^* \to \Delta_i$ be a hash function. Domain Δ_i depends on vk_i.

[Signature Generation]

G-1 (Initialization)**:** Compute

$$e_k = \begin{cases} A_k(sk_k\,;\alpha) & (vk_k \text{ is } \textsf{type-T}), \text{ or} \\ \beta & (vk_k \text{ is } \textsf{type-H}), \end{cases}$$

where $\alpha \leftarrow \Lambda_k$ and $\beta \leftarrow \Delta_k$ (Λ_k denotes an appropriate space of randomness defined by the algorithm of A_k and sk_k). Then compute $c_{k+1} = H_{k+1}(L, m, e_k)$.

G-2 (Forward sequence)**:** For $i = k+1, .., n-1, 0, ..., k-1$, compute

$$e_i = \begin{cases} V_i(s_i, c_i, vk_i) & (vk_i \text{ is } \textsf{type-T}), \text{ or} \\ c_i + F_i(s_i, vk_i) & (vk_i \text{ is } \textsf{type-H}), \end{cases}$$

where s_i is randomly chosen. Then compute $c_{i+1} = H_{i+1}(L, m, e_i)$.

G-3 (Forming the ring)**:**

$$s_k = \begin{cases} Z_k(sk_k, \alpha, c_k) & (vk_k \text{ is } \textsf{type-T}), \text{ or} \\ I_k(\beta - c_k, sk_k) & (vk_k \text{ is } \textsf{type-H}). \end{cases}$$

The resulting signature for m and L is $(c_0, s_0, s_1, \cdots, s_{n-1})$.

[Signature Verification]

For $i = 0, \cdots, n-1$, compute

$$e_i = \begin{cases} V_i(s_i, c_i, vk_i) & (vk_i \text{ is } \textsf{type-T}), \text{ or} \\ c_i + F_i(s_i, vk_i) & (vk_i \text{ is } \textsf{type-H}), \end{cases}$$

and then $c_{i+1} = H_{i+1}(L, m, e_i)$ if $i \neq n-1$. Accept if $c_0 = H_0(L, m, e_{n-1})$. Reject otherwise.

4.3 Remark on Compatibility of Keys

Some signature schemes are neither type-T nor type-H. For such schemes, we consider *compatibility* among signature schemes. Signature scheme A is compatible with scheme B if 1) A's private and public keys can be used to issue and verify signatures of scheme B, and 2) breaking B (in EUF-CMA sense) implies breaking A using the same key. For instance, DSS is not either type but it is compatible with the Schnorr signature scheme of type-T. Since breaking the Schnorr signature scheme implies that the discrete-log is tractable with regard to the key, it implies DSS is broken, too. Thus, DSS keys can be involved in our scheme with type-T.

With regard to type-H schemes, however, special care may be needed. Remember that type-H only shows the ability of computing $I_{sk}(\cdot)$ and does not necessarily imply possession of *sk*. Therefore, it is not sufficient that scheme A's keys can be used to scheme B, but it has to be true that ability of computing $I_{sk}(\cdot)$ of scheme B is sufficient to generate signatures of A.

The signature scheme in [2] is a curious scheme that belongs to type-H but its keys are also compatible with type-T ones. In such a case, one can select more efficient type to involve the keys.

5 Concrete Examples

Leaving out the security proof for the abstract scheme (which turns out to be similar to the one shown in Section 5.3), we present concrete examples and their security proofs in order to help readers who is familiar with RSA and Schnorr signatures grasp the ideas for our construction and the security proofs.

5.1 All Discrete-log Case

For $i = 0, \ldots, n-1$, let (y_i, p_i, q_i, g_i) be DL public-keys as described in Section 3.1 and $H_i : \{0,1\}^* \to \mathbb{Z}_{q_i}$ be hash functions. Let L be a list of these public-keys. A signer who has private key x_k generates a signature for message m as follows.

[Signature Generation]

- **D-1** (Initialization)**:** Select $\alpha \leftarrow \mathbb{Z}_{q_k}$ and compute $c_{k+1} = H_{k+1}(L, m, {g_k}^{\alpha} \bmod p_k)$.
- **D-2** (Forward sequence)**:** For $i = k+1, .., n-1, 0, ..., k-1$, select $s_i \leftarrow \mathbb{Z}_{q_i}$ and compute $c_{i+1} = H_{i+1}(L, m, g_i^{s_i} y_i^{c_i} \bmod p_i)$.
- **D-3** (Forming the ring)**:** Compute $s_k = \alpha - x_k c_k \bmod q_k$.

The resulting signature for m and L is $(c_0, s_0, s_1, \cdots, s_{n-1})$.

[Signature Verification]

For $i = 0, \ldots, n-1$, compute $e_i = g_i^{s_i} y_i^{c_i} \bmod p_i$ and then $c_{i+1} = H_{i+1}(L, m, e_i)$ if $i \neq n-1$. Accept if $c_0 = H_0(L, m, e_{n-1})$. Reject otherwise.

Intuitively, this scheme is a ring of the Schnorr signatures where each challenge is taken from the previous step. Indeed, it is the Schnorr signature scheme when $n = 1$.

Theorem 2. *The above all-DL scheme is unconditionally signer-ambiguous.*

Proof. Observe that all s_i are taken randomly from $\mathbb{Z}_{q_i}$ except for s_k at the closing point. At the closing point, s_k also distributes uniformly over $\mathbb{Z}_{q_k}$ since α is uniformly chosen from $\mathbb{Z}_{q_k}$. Therefore, for fixed (L, m), $(s_0, \ldots, s_{n-1})$ has $\prod_{i=0}^{n-1} q_i$ variation that are equally likely regardless of the closing point. Remaining c_0 in a signature is determined uniquely from (L, m) and s_i's. □

Note that the signer ambiguity does not rely on ideal randomness assumption on the hash function.

Next, we claim unforgeability. Let $\mathcal{A}$ be a $(\tau, \epsilon, q_s, q_h)$-adversary that requests signing oracle at most q_s times and accesses random oracles at most q_h times in total and output forged (L, m, σ) with probability at least ϵ and running time at most τ. The following theorem can be proven (see Appendix B).

Theorem 3. *If there exists $(\tau, \epsilon, q_s, q_h)$-adversary $\mathcal{A}$ for public-key set $\mathcal{L}$ of size n, then there exists (η, μ)-simulator* sim *that uses $\mathcal{A}$ as a black-box and computes discrete-logarithm x_i of $(y_i, p_i, q_i, g_i) \in \mathcal{L}$ for at least one i with probability at least μ within running time η. Here, $\eta < \frac{32q_h^2+4}{\epsilon} \cdot \tau$ and $\mu > \frac{9}{100}$ under the condition that $\epsilon > \frac{8q_h^2}{q}$ and $q > 2\,q_h q_s$ where q is the smallest q_i included in $\mathcal{L}$.*

We remark that the running time only concerns the number of black-box execution of $\mathcal{A}$. Note also that the condition $q > 2\,q_h q_s$ is not essential and used only for simplifying the presentation of the reduction cost so that the impact of each variable is comprehensible. If necessary, one can obtain detailed formula without the condition. These remarks apply to all the theorems in the rest of this paper.

5.2 All RSA Case

For $i = 0, \ldots, n-1$, let (e_i, N_i) be RSA public-keys and $H_i : \{0,1\}^* \to \mathbb{Z}_{N_i}$ be hash functions. Let L be a list of these public-keys. A signer who has private key d_k generates a signature for message m as follows.

[Signature generation]

- T-1 (Initialization)**:** Select $r_k \leftarrow \mathbb{Z}_{N_k}$ and compute $c_{k+1} = H_{k+1}(L, m, r_k)$.
- T-2 (Forward sequence)**:** For $i = k+1, .., n-1, 0, ..., k-1$, select $s_i \leftarrow \mathbb{Z}_{N_i}$, and compute $c_{i+1} = H_{i+1}(L, m, c_i + s_i^{e_i} \bmod N_i)$.
- T-3 (Shaping into a ring)**:** Compute $s_k = (r_k - c_k)^{d_k} \bmod N_k$

The resulting signature for m and L is $(c_0, s_0, s_1, \ldots, s_{n-1})$.

[Signature Verification]

For $i = 0, \ldots, n-1$, compute $r_i = c_i + s_i^{e_i} \bmod N_i$ and then $c_{i+1} = H_{i+1}(L, m, r_i)$ if $i \neq n-1$. Accept if $c_0 = H_0(L, m, r_{n-1})$. Reject, otherwise.

Unconditional signer-ambiguity can be proven in the same way as that for the all DL-based scheme. Unforgeability is also proven in the similar way. We wrap random oracles for each hash function in the same way as done in the proof for the DL version. The following theorem is proven (see Appendix C).

Theorem 4. *If there exists $(\tau, \epsilon, q_s, q_h)$-adversary $\mathcal{A}$ for public-key set $\mathcal{L}$ of size n, then there exists (η, μ)-simulator* sim *that, given $(w_0, \ldots, w_{n-1})$, uses $\mathcal{A}$ as a black-box, and computes $w_i^{d_i} \bmod N_i$ for some $i \in \{0, \ldots, n-1\}$ with probability at least μ and running-time within η. Here, $\eta \approx \tau$ and $\mu > \frac{1}{4{q_h}^2}\epsilon$ under the condition of $N > 2\, q_h q_s$ where N is the smallest modulus among all N_i in $\mathcal{L}$.*

5.3 Mixture Case: RSA and Schnorr

We finally show a small example for involving both RSA and DL keys. For simplicity, we consider the case $n = 2$, i.e., only two public-keys are involved. Let L be consists of RSA public-key (e, N) and one Schnorr public-key (y, g, p, q). Let $H_0 : \{0,1\}^* \to \mathbb{Z}_N$ and $H_1 : \{0,1\}^* \to \mathbb{Z}_q$ be hash functions. A signer who has the RSA private key, d, generates a signature for message m as follows.

[Signature generation]

- **M-1** (Initialization)**:** Select $\beta \leftarrow \mathbb{Z}_N$ and compute $c_1 = H_1(L, m, \beta)$.
- **M-2** (Forward sequence)**:** Select $s_1 \leftarrow \mathbb{Z}_q$ and compute $c_0 = H_0(L, m, g^{s_1} y^{c_1} \bmod p)$.
- **M-3** (Shaping into a ring)**:** Compute $s_0 = (\beta - c_0)^d \bmod N$

The resulting signature is (c_0, s_0, s_1).

[Signature Verification]
Given (L, m, c_0, s_0, s_1), compute $c_1 = H_1(L, m, c_0 + s_0^e \bmod N)$. Accept if $c_0 = H_0(L, m, g^{s_1} y^{c_1} \bmod p)$. Reject, otherwise.

The signature can be shorten by selecting (c_1, s_1, s_0) as a signature because $|c_0|$ is the size of RSA modulus typically > 1024 bits while $|c_1|$ is the size of q typically > 160 bits.

Unconditional signer-ambiguity can be proven as well as the former examples. Regarding unforgeability, we prove the following theorem by following the similar way as done in the proof of Theorem 3 and 4. Sketch of the proof is shown in Appendix D.

Theorem 5. *The above scheme is existentially unforgeable against adaptive chosen message and chosen public-key attacks.*

6 Efficiency

We compare our ring signature scheme with the existing schemes using DL, ECDL(elliptic curve DL) and RSA trapdoor functions, in terms of the length of a signature and the computational cost of signature generation and verification. We refer the scheme in Section 3.1 by "WI signatures" and the scheme in Section 3.2 with RSA trapdoor function by "RSA ring signatures", hereafter. Let n be the number of signers of ring signature.

Table 1 shows the comparison in terms of the length of signature. Here, L(DL) is the length of exponent of DL signature, and is typically 160-bit. L(RSA) is

Table 1. The table shows the length of signature and its typical value (bit).

	Length of signature	Typical value
WI signature	$(L(\text{DL}) + L(\text{DL})) \times n$	$320 \times n$
Ours with DL	$L(\text{DL}) + L(\text{DL}) \times n$	$160 + 160 \times n$
Ours with ECDL	$L(\text{EC}) + L(\text{EC}) \times n$	$160 + 160 \times n$
RSA ring signature	$(L(\text{RSA}) + 160) + (L(\text{RSA}) + 160) \times n$	$1184 + 1184 \times n$
Ours with RSA	$L(\text{RSA}) + L(\text{RSA}) \times n$	$1024 + 1024 \times n$

the length of modulus of RSA signature, and is typically 1024-bit. $L(\text{EC})$ is the length of the size of cyclic subgroup in elliptic curve, and is typically 160-bit. From the table, we can see that the length of our signature with DL is *one half* of WI signature for large n, and that the length of our signature with RSA is 0.8 of RSA ring signature.

Table 2. The table shows the computational costs of signature generation and verification and its typical value (arithmetic operation).

	Costs of generation	Typical value
WI signature	$T(\text{DL}) \times 5/4 \times n$	$2.0 \times 10^8 \times n$
Ours with DL	$T(\text{DL}) \times 5/4 \times n$	$2.0 \times 10^8 \times n$
Ours with ECDL	$T(\text{EC}) \times 5/4 \times n$	$7.1 \times 10^7 \times n$
RSA ring signature	$T(\text{RSA}^{-1}) + T(\text{RSA}) \times n$	$1.0 \times 10^9 + 1.6 \times 10^7 \times n$
Ours with RSA	$T(\text{RSA}^{-1}) + T(\text{RSA}) \times n$	$1.0 \times 10^9 + 1.6 \times 10^7 \times n$
	Costs of verification	Typical value
WI signature	$T(\text{DL}) \times 5/4 \times n$	$2.0 \times 10^8 \times n$
Ours with DL	$T(\text{DL}) \times 5/4 \times n$	$2.0 \times 10^8 \times n$
Ours with ECDL	$T(\text{EC}) \times 5/4 \times n$	$7.1 \times 10^7 \times n$
RSA ring signature	$T(\text{RSA}) \times n$	$1.6 \times 10^7 \times n$
Ours with RSA	$T(\text{RSA}) \times n$	$1.6 \times 10^7 \times n$

Table 2 shows the comparison in terms of the computational costs of signature generation and verification. Here, $T(DL)$, $T(EC)$, $T(RSA^{-1})$ and $T(RSA)$ are the computational costs of modular exponentiation, scalar multiplication on elliptic curve, inverse RSA function and RSA function, respectively. Typically, $T(\text{DL}) = T((1024)^{(160)})$, $T(\text{EC}) = T((160) \cdot (EC160))$, $T(\text{RSA}^{-1}) = T((1024)^{(1024)})$ and $T(\text{RSA}) = T((1024)^{(16)})$. Here, $T((x)^{(y)})$ is the number of (single precision) arithmetic operation of exponentiation with x-bit modulus and y-bit exponent, and is estimated $x^2 \times y$. Exponentiation with y-bit exponent needs y x-bit multiplications, using binary method and the fact costs of square is half of multiplication. x-bit multiplication needs x^2 (single precision) arithmetic operations. $T((y) \cdot (ECx))$ is the number of (single precision) arithmetic operation of scalar multiplication on elliptic curve with x-bit base field and y-bit scalar, and is estimated $x^2 \times 14 \times y$. Scalar multiplication on elliptic curve with y-bit scalar needs y additions of points, using binary method and the fact costs of doubling is half of costs of addition. Addition of points with x-bit base field needs 14 x-bit multiplications, using Jacobian coordinate. x-bit multiplication needs x^2 (single precision) arithmetic operations. Hence, we have

$$\begin{aligned} T((1024)^{(1024)}) &= 1024^2 \times 1024 \approx 1.07 \times 10^9, \\ T((1024)^{(160)}) &= 1024^2 \times 160 \approx 1.67 \times 10^8, \\ T((1024)^{(16)}) &= 1024^2 \times 16 \approx 1.67 \times 10^7, \\ T((160) \cdot (EC160)) &= 160^2 \times 14 \times 160 \approx 5.73 \times 10^7. \end{aligned}$$

The computational costs of exponentiation with two basis is 5/4 of exponentiation with single basis, using two basis binary method. From the table, we can see that the computational costs of our signature with DL is as same as WI signature, and that the computational costs of our signature with RSA is as same as RSA ring signature.

Notice that in known schemes the length and the computational costs of signature is proportional to the *maximum* of the length of DL exponent / RSA modulus. In our scheme, the length and the computational costs of signature is proportional to the *average* of the length of DL exponent / RSA modulus, since our scheme need not to round up the length to the maximum length.

Acknowledgements

The authors are grateful to Emmanuel Bresson and the anonymous referees for their helpful comments.

References

1. M. Abe and F. Hoshino. Remarks on mix-network based on permutation network. *PKC 2001*, LNCS 1992, pp. 317–324. Springer-Verlag, 2001.
2. D. Boneh, B. Lynn, and H. Shacham. Short signatures from the weil pairing. *Asiacrypt 2001*, LNCS 2248, pp. 514–532. Springer-Verlag, 2001.
3. M. Bellare, and P. Rogaway. Random Oracles are practical: a paradigm for designing efficient protocols. *1st ACM CCCS*, pp. 62–73. ACM, 1993.
4. E. Bresson, J. Stern, and M. Szydlo. Threshold ring signatures and applications to ad-hoc groups. *CRYPTO 2002*, LNCS 2442, pp. 465–480. Springer-Verlag, 2002.
5. J. Camenisch. Efficient and generalized group signatures. *EUROCRYPT '97*, LNCS 1233, pp. 465–479. Springer-Verlag, 1997.
6. J. Camenisch and M. Michels. Proving in zero-knowledge that a number is the product of two safe primes. *EUROCRYPT '99*, LNCS 1592, pp. 107–122. Springer-Verlag, 1999.
7. C. Chan, Y. Frankel, and Y. Tsiounis. Easy come - easy go divisible cash. *EUROCRYPT '98*, LNCS 1403, pp. 561–575. Springer-Verlag, 1998.

8. D. Chaum and E. Van Heyst. Group signatures. *EUROCRYPT '91*, LNCS 547, pp. 257–265. Springer-Verlag, 1991.
9. R. Cramer, I. Damgård, and B. Schoenmakers. Proofs of partial knowledge and simplified design of witness hiding protocols. *CRYPTO '94*, LNCS 839, pp. 174–187. Springer-Verlag, 1994.
10. R. Cramer, M. Franklin, B. Schoenmakers, and M. Yung. Multi-authority secret-ballot elections with linear work. *EUROCRYPT '96*, LNCS 1070, pp. 72–83. Springer-Verlag, 1996.
11. R. Cramer, R. Gennaro, and B. Schoenmakers. A secure and optimally efficient multi-authority election scheme. *EUROCRYPT '97*, LNCS 1233, pp. 103–118. Springer-Verlag, 1997.
12. U. Feige, A. Fiat, and A. Shamir. Zero-knowledge proofs of identity. *J. Cryptology*, 1:77–94, 1988.
13. U. Feige and A. Shamir. Witness indistinguishable and witness hiding protocols. STOC'90, pp. 416–426, 1990.
14. A. Fiat and A. Shamir. How to prove yourself: Practical solutions to identification and signature problems. *CRYPTO '86*, LNCS 263, pp. 186–199. Springer-Verlag, 1987.
15. S. Goldwasser, S. Micali, and R. Rivest. A digital signature scheme secure against adaptive chosen-message attacks. *SIAM J. Computing*, 17(2):281–308, April 1988.
16. L. C. Guillou and J.-J. Quisquater. A practical zero-knowledge protocol fitted to security microprocessor minimizing both transmission and memory. *EUROCRYPT '88*, LNCS 330 of *Lecture Notes in Computer Science*, pp. 123–128. Springer-Verlag, 1988.
17. M. Jakobsson, K. Sako, and R. Impagliazzo. Designated verifier proofs and their applications. *EUROCRYPT '96*, LNCS 1070, pp. 143–154. Springer-Verlag, 1996.
18. M .Naor. Deniable ring authentication. *CRYPTO 2002*, LNCS 2442, pp. 481–498. Springer-Verlag, 2002.
19. K. Ohta and T. Okamoto. On concrete security treatment of signatures derived from identification. *CRYPTO '98*, LNCS 1462, pp. 354–369. Springer-Verlag, 1998.
20. D. Pointcheval and J. Stern. Security arguments for digital signatures and blind signatures. *J. Cryptology*, 2000.
21. R. Rivest, A. Shamir, and Y. Tauman. How to leak a secret. *Asiacrypt 2001*, LNCS 2248, pp. 552–565. Springer-Verlag, 2001.
22. A. De Santis, G. Di Crescenzo, G. Persiano, and M. Yung. On monotone formula closure of SZK. *FOCS'94*, pp. 454–465, 1994.
23. C. P. Schnorr. Efficient signature generation for smart cards. *J. Cryptology*, 4(3):239–252, 1991.

Appendix A

The following an efficient 1-out-of-n signature scheme in non-separable model based on the representation problem.

Let p, q be large primes. Let $\langle g \rangle$ denote a prime subgroup in $\mathbb{Z}_p^*$ generated by g whose order is q. Let x_i, y_i be $y_i = g^{x_i} \bmod p$. Here x_i is the secret-key and (y_i, p, q, g) is the public-key. All member use common p, q, g in the non-separable model. So only y_i is different in public-keys for each member. Let L be a set of (y_i, p, q, g) for $i = 1, \ldots, n$. Let $H : \{0,1\}^* \to \mathbb{Z}_q$ be a hash function.

A signer who owns secret key x_k generates a signature for message m with public-key list L that includes y_k, in the following way.

S-1 Select $\alpha, c_i \leftarrow \mathbb{Z}_q$ for $i = 0, \ldots, n-1$, $i \neq k$, and compute $z = g^\alpha \prod_{i=0,i\neq k}^{n-1} y_i^{c_i} \bmod p$.

S-2 Compute

$$c = H(L, m, z)$$
$$c_k = c - (c_0 + \cdots + c_{k-1} + c_{k+1} + \cdots + c_{n-1}) \bmod q$$
$$s = \alpha - c_k \cdot x_k \bmod q.$$

The resulting signature is $\sigma = (s, c_0, \ldots, c_{n-1})$. (L, m, σ) is valid if

$$\sum_{i=0}^{n-1} c_i \equiv H(L, m, g^s y_0^{c_0} \cdots y_{n-1}^{c_{n-1}}) \bmod q.$$

The security of this scheme can also be reduced to the discrete-log problem by rewinding simulation. But the reduction is quite costly because we may have to have at most n successful rewinding simulations to extract only one secret-key (in this worst case, all the secret-keys are extracted at once).

Appendix B (*Proof of Theorem 3*)

To get the black-box $\mathcal{A}$ run properly, sim simulates the random oracles that corresponds to each hash function and the signing oracle. For simplicity, the random oracles are treated as a single oracle that takes $Q_j = (i, L_j, m_j, r_j)$ as j-th query and returns a random value that corresponds to $H_i(L_j, m_j, r_j)$ maintaining consistency against duplicated queries. The signing oracle receives the j-th query, say $R_j = (L_j, m_j)$, to sign. To avoid complicated suffixes, we describe a public-key with a suffix relative to L_j in the current context. So, $y_0 \in L_j$ and $y_0 \in L_{j'}$ could differ. We hope that this should cause no confusion. The corresponding answer is simulated in the following way.

D'-1: Choose $c_0 \leftarrow \mathbb{Z}_{q_0}$.

D'-2: For $i = 0, \ldots, |L_j| - 1$, select $s_i \leftarrow \mathbb{Z}_{q_i}$, compute $e_i = g_i^{s_i} y_i^{c_i} \bmod p_i$, and then compute $c_{i+1} = H_{i+1}(L_j, m_j, e_i)$ if $i \neq |L_j| - 1$.

D'-3: Assign c_0 to the value of $H_0(L_j, m_j, e_{|L_j|-1})$.

The simulation fails if Step D'-3 causes inconsistency in H_0. It happens with probability at most q_h/q where q is the smallest q_i in $\mathcal{L}$. Hence, the simulation is successful q_s times with probability at least $(1 - q_h/q)^{q_s} \geq 1 - q_h q_s/q$.

Let Θ, Ω be the random tapes given to the signing oracle and $\mathcal{A}$. The success probability of $\mathcal{A}$ is taken over the space defined by Θ, Ω and random oracle H. Let $\mathcal{S}$ be a set of (Θ, Ω, H) with which $\mathcal{A}$ is successful in forgery. From the definition of ϵ, we have $\Pr[(\Theta, \Omega, H) \in \mathcal{S}] \geq \epsilon$. Let $(L, m, c_0, s_0, \cdots, s_{n'-1})$ be a forged signature $\mathcal{A}$ outputs. Here $n' = |L|$. Define $r_i = {g_i}^{s_i} {y_i}^{c_i} \bmod p_i$ and $c_{i+1} = H_{i+1}(L, m, r_i)$ for $i = 0, \ldots, n'-1$ (indices are taken modulo n'). Then, with probability at least $1 - 1/q$, there exist queries $Q_j = (i+1, L, m, r_i)$ for all

$i = 0, \ldots, n' - 1$ due to the ideal randomness of H. Let $\mathcal{S}'$ be a subset of $\mathcal{S}$ where $(\Theta, \Omega, H) \in \mathcal{S}'$ leads $\mathcal{A}$ to output a signature that has corresponding queries as above with successfully simulated signing oracle. Then, $\Pr[(\Theta, \Omega, H) \in \mathcal{S}'] \geq (1 - q_h q_s/q)(1 - 1/q)\epsilon$. Let $\epsilon' = (1 - q_h q_s/q)(1 - 1/q)\epsilon$.

Since the queries form a ring, there exists at least one index, say k, in $\{0, \ldots, n' - 1\}$ such that $Q_u = (k + 1, L, m, r_k)$ and $Q_v = (k, L, m, r_{k-1})$ satisfy $u \leq v$. Namely, k is in between the gap of query order. We call such (u, v) a gap index. Note that $u = v$ happens only if $n' = 1$, which means that resulting L contains only one public-key. If there are two or more gap indices with regard to a signature, only the smallest one is considered. We classify $\mathcal{S}'$ by the gap indices. Let $\mathcal{S}'_{u,v}$ denote a class where $(\Theta, \Omega, H) \in \mathcal{S}'_{u,v}$ yields gap indices (u, v). There are at most $\binom{2}{q_h} + \binom{1}{q_h} = q_h(q_h + 1)/2$ classes. By invoking $\mathcal{A}$ with randomly chosen (Θ, Ω, H) at most $t_1 = 1/\epsilon'$ times, sim finds at least one $(\Theta, \Omega, H) \in \mathcal{S}'_{u,v}$ for some gap index (u, v) with probability $1 - \exp(-1) > 3/5$.

We consider a set of gap indices that is likely to appear when (Θ, Ω, H) is chosen randomly. Let $GI = \{(u, v) \mid |\mathcal{S}'_{u,v}|/|\mathcal{S}'| \geq \frac{1}{q_h(q_h+1)}\}$ and $B = \{(\Theta, \Omega, H) \in \mathcal{S}'_{u,v} | (u, v) \in GI\}$. Then, it holds that $\Pr[B|\mathcal{S}'] \geq \frac{1}{2}$. Due to this fact known as heavy-row lemma, (Θ, Ω, H) that yields the successful run of $\mathcal{A}$ is in B with probability at least $1/2$.

Split H as (H^-, c_k) where H^- corresponds to the answers to all queries except for Q_v answered with c_k. Due to the heavy-row lemma, again, with probability at least $1/2$, (Θ, Ω, H^-) satisfies $\Pr_{c'_k}[(\Theta, \Omega, H^-, c'_k) \in \mathcal{S}'_{u,v}] \geq \frac{\epsilon'}{2q_h(q_h+1)}$. Since we assume $\epsilon > \frac{8{q_h}^2}{q}$ and $q > 2q_h q_s$, it holds that $\frac{\epsilon'}{2q_h(q_h+1)} > 1/q$.

By running $\mathcal{A}$ up to $t_2 = (\frac{\epsilon'}{2q_h(q_h+1)} - \frac{1}{q})^{-1}$ times with (Θ, Ω, H^-) obtained in the first successful run and randomly chosen $c'_k (\neq c_k)$, then, with probability at least $3/5$, sim finds at least one c'_k such that $(\Theta, \Omega, H^-, c'_k) \in \mathcal{S}_{u,v}$. Since Q_u happens before Q_v, r_i is unchanged for both runs. Therefore, sim can compute the discrete-log, $x_k = (s_k - s'_k)/(c'_k - c_k) \bmod q_k$. Overall success probability is

$$\mu > \frac{3}{5} \cdot \frac{1}{2} \cdot \frac{1}{2} \cdot \frac{3}{5} = \frac{9}{100},$$

and the number of invocation of $\mathcal{A}$ is

$$t_1 + t_2 < \frac{1}{\epsilon'} + \frac{4q_h(q_h + 1)}{\epsilon'} < \frac{4}{\epsilon} + \frac{4 \cdot 4 \cdot 2q_h^2}{\epsilon} = \frac{32q_h^2 + 4}{\epsilon}.$$

Appendix C (*Proof of Theorem 4*)

The first half of the proof is the same as the one for Theorem 3 (the simulation of the singing oracle is different but can be done only with trivial changes). That is, there exists class $\mathcal{S}'$ such that $(\Theta, \Omega, H) \in \mathcal{S}'$ results in a successful simulation of the signing oracle and the forged signature has corresponding queries to the random oracle. Accordingly, $\Pr[(\Theta, \Omega, H) \in \mathcal{S}'] \geq (1 - q_h q_s/N)(1 - 1/N)\epsilon$.

At the beginning of the simulation, sim selects a pair of index (u, v) randomly so that $1 \leq u \leq v \leq q_h$. With probability $2/q_h(q_h + 1)$, the guess is correct and

sim receives $Q_u = (k+1, L, m, r_k)$ and $Q_v = (k, L, m, r_{k-1})$ so that (u, v) is a gap index. Let k' be an index such that $vk_{k'} \in \mathcal{L}$ corresponds to $vk_k \in L$. When query Q_v is made (u-th query has been already made by this moment), sim returns $c_k = r_k - w_{k'} \bmod N_k$ as a value of $H_k(L, m, r_{k-1})$. If $\mathcal{A}$ is successful in forgery, it outputs s_k that satisfies $r_k \equiv c_k + s_k^{e_k} \bmod N_k$. Since $r_k \equiv c_k + w_{k'} \bmod N_k$, we obtain s_k as the inverse of $w_{k'}$ with regard to the public-key $vk_{k'}$ in $\mathcal{L}$.

Overall success probability of sim is

$$\mu \geq \frac{(1 - q_h q_s/N)(1 - 1/N)}{q_h(q_h + 1)/2}\epsilon > \frac{1}{4{q_h}^2}\epsilon.$$

The rightmost term assumes $N > 2q_h q_s$. The running time η is almost the same as τ as sim runs $\mathcal{A}$ only once and the simulation cost for the signing oracle and the random oracle are assumed to be sufficiently smaller than τ.

Appendix D (*Proof of Theorem 5*)

We show that if there exists $(\tau, \epsilon, q_s, q_h)$-adversary $\mathcal{A}$ for a set of public-key, L, of size n, then there exists (τ', ϵ')-simulator sim such that using $\mathcal{A}$ as a black-box, it computes $w^d \bmod N$ for $w \leftarrow N$ with probability greater than $3/5$ and running time no more than $\frac{4{q_h}^2}{\epsilon}\tau$, or computes the discrete-logarithm of y with probability greater than $\frac{9}{100}$ and running time no more than $\frac{32q_h^2+4}{\epsilon}\tau$.

Let γ be $\min(q, N)$. We assume that $\gamma > 2\,q_h q_s$ and $\epsilon > \frac{8q_h^2}{q}$.

The proof is by combining the proofs for Theorem 3 and 4. Let $\epsilon' = (1 - q_h q_s/|\gamma|)(1 - 1/|\gamma|)\epsilon$.

Simulator sim guesses (u, v) and runs $\mathcal{A}$. If $Q_v = (0, L, m, r_1)$ for some L, m and r_1, sim checks if $Q_u = (1, L, m, r_0)$ for the same L, m and some r_0. If it is the case, sim chooses $t \leftarrow N$ and returns $c_0 = r_0 - wt^e \bmod N$ as the value of $H_0(L, m, r_1)$. If $\mathcal{A}$ succeeds and (u, v) forms a gap, it holds that $s_k = (r_0 - c_0)^d \equiv (wt^e)^d \equiv w^d t \bmod N$. Accordingly, $s_k/t = w^d \bmod N$. In this case, simulation ends here successfully. Such a successful case happens with probability greater than $3/5$ while repeating the simulation at most $\{\frac{(1-q_h q_s/N)(1-1/N)}{q_h(q_h+1)/2}\epsilon\}^{-1} < \frac{4{q_h}^2}{\epsilon}$ (this is straightforward from the proof of Theorem 4 in Appendix C).

For all other cases, sim proceeds as follows. Regardless of the initial guess of (u, v), sim completes an execution of $\mathcal{A}$. If $\mathcal{A}$ succeeds and the resulting gap index, say (u', v'), corresponds to the queries $Q_{u'} = (0, L, m, r_1)$ and $Q_{v'} = (1, L, m, r_0)$, namely, if the resulting gap index comes across the discrete-log key, sim proceeds to rewinding simulation with the forking point v' in the same way as done in the proof of Theorem 3. As a result, sim gets a collision and computes the discrete-log, x. The success probability and the running time for this case is the same as that in the proof of Theorem 3.