Trends in Mathematics
Information Center for Mathematical Sciences
Volume 8, Number 1, June, 2005, Pages 29–43

ACCOUNTABLE SIGNATURES

JUNG HEE CHEON, WOO-HWAN KIM, AND HYOJIN YOON

Abstract. We propose a new notion of *accountable signatures* in which related signatures form a signature chain consisting of a root signature and subsequent signatures. In a signature chain, each subsequent signature is indexed and if distinct messages are signed along with the same index, the signer's secret key is revealed. Accountable signature is an extension of *k-times signatures* in which a signer is enforced not to generate signatures more than k and the signature and the certificate size and the verification cost is constant with respect to k. We give an instance of accountable signatures based on a Gap Diffie-Hellman group and prove its security against adaptive chosen-[index, root signature, and message] attack in the random oracle model under the Diffie-Hellman assumption. Further, we show that our accountable signature admits secure batch verification of many signatures almost at one verification cost with small security loss.

1. Introduction

In the real world, we can easily meet the situation that only limited number of actions are approved: A depositor can withdraw money from ATM (Automated Teller Machine) by the predetermined times in a day. A book of coupons consists of fixed number of tickets. Some membership certificates are highly priced by limited issuing. In digital signatures, however, once a certificate for his public key is given, the signer can generate as many signatures as he wants. The valid time for the signing key may be restricted by inserting the terms of validity into the certificate by an authority. However, it is rather difficult to control the number of signatures by a certificate authority. A digital signature scheme with this functionality is desirable especially in group signatures and proxy signatures. Electronic coupons or admission tickets with limited use are another examples.

One-time signature [Rab78] is a digital signature mechanism which can be used to sign at most one message, otherwise signatures can be forged. This notion can be extended to *k-times signature* in which a signer can generate at most k signatures. If we use a Merkle hash tree for authenticating random commitments, we can get a k-times signature scheme with a certificate of a constant size [Mer89]. But its signature size and verification cost is proportional to $\log k$. In [OO89], Okamoto and Ohta introduced a disposable zero-knowledge authentication for untraceable electronic cash. Though not specified in [OO89], 'electronic coupon tickets' can be easily converted into a k-times signature scheme, whose certificate size, signature

size, and verification cost is constant with respect to k. The scheme can be considered as a variant of Guillou-Quisquater signature [GQ88] and so its security is based on the integer factorization problem (IFP). On the other hand, the previous schemes on the discrete logarithm problem (DLP) has efficiency proportional to k or $\log k$ [EO94, HKLL03]. There is no known k-times signature scheme based on the DLP whose efficiency is independent of k.

1.0.1. *Motivation.* We started from the Schnorr signature scheme: Let G be a cyclic group of order q and g a generator of G. Given a private/public key pair (x, g^x), a signature for a message m consists of (r, s) where $r = g^k$, $s = k + xh \pmod q$, k is a random value in $\mathbb{Z}_p^*$, and h is the hash value of (m, r). If the same r is used to generate two signatures with different messages, the secret key x is revealed. On the other hand, if many random r is to be certified, the certificate size increases. Thus one possible choice is to use the same random number in a different form. But if it is an algebraic relation, it is also easily solved. For example, if $k_i = k^i$ is used for i-th signature generation, the simultaneous quadratic equations $s_1 = k + xh_1$ and $s_2 = k^2 + xh_2$ with unknown k and x over $\mathbb{Z}_q$ can be solved easily [Mao96]. It is the different part from the signature scheme based on the IFP. Another choice is to use $r_i = H(r, i)$ for a one-way hash function H and an index i. But this r_i can not be used as a random commitment part in the Schnorr scheme without knowledge of the discrete logarithm k_i of r_i with respect to g. This problem may be overcome in the Gap-Diffie-Hellman Group.

1.0.2. *Our Contributions.* We introduce a new notion of *accountable signatures* consisting of root signatures and their corresponding subsequent signatures and present an instance based on a Gap Diffie-Hellman (GDH) group. Each subsequent signature for a root signature is indexed and if distinct messages are signed along with the same index, the signer's secret key is revealed. We remark that the Okamoto and Ohta's scheme [OO89] can be converted into an accountable signature based on the IFP.

Our scheme can be directly applied for a k-times signature. On the contrary to the previous schemes based on the DLP, the size of signature and certificate and the verification cost of our scheme are constant with respect to k. Further it may be also used to guarantee the order of previously issued signatures without using time stamping authority, or to audit any removal or replacement of signed messages in a public bulletin board. For example, when a customer gives an evaluation on the restaurant and uploads it with a token serially signed by the homepage manager in its homepage, any replacement of a bad reputation is not possible without issuing two tokens with the same serial number. Otherwise it reveals the secret key of the manager and can ruin all of the evaluation values in the homepage.

We propose a formal definition and a security model for accountable signatures and prove that our scheme is secure against adaptive chosen-[index, root signature, and message] attack in the random oracle model when the underlying group is a GDH group. Inthe security proof, we consider two types of forgers: One is to forge

a root signature and the other is to forge a subsequent signature with a chosen index from the root signature chosen by himself.

Also we realize the batch verifications of our signature scheme with provable security. Since our signature scheme has a fixed base on the contrary to RSA [Fia89, BGR98] or BGLS [BGLS03], precomputation technique can be applied to get more speed up [LL94]. However, a batch verifier should be taken carefully since it may disturb reveal of the secret key.

Finally, we note that our scheme is easily converted into an ID-based one, called an *ID-based accountable signature*.

1.0.3. *Organization.* The rest of paper is organized as follows: In Section 2, we introduce the formal definition and security model of the accountable signatures. In Section 3, we propose our accountable signature scheme and give a sketch of the security proof. Section 4 deals with the batch verification. We conclude the paper by introducing some further studies in Section 5. A rigorous security proof in the random oracle model is presented in Appendix.

2. Formal Definitions

2.1. **Definition of Accountable Signature Schemes.** We introduce the notion of an accountable signature scheme.

Definition 1. *An accountable signature scheme is a digital signature scheme which consists of the following six algorithms:*

- KeyGen.: Given a security parameter ℓ as input, output a public key and secret key pair (PK, SK).
- RootSign.: Given a message m and SK, output a signature $\sigma = (m, U, V)$, called a root signature in public, and k_σ in private. k_σ is used for generating subsequent signatures, as described below, from a root signature σ.
- SubSign.: Given an index $i \in \mathbb{N}$, a random string $x^{(i)}$, a message m_i, a root signature σ, k_σ and a secret key SK, output $\sigma^{(i)} = (\sigma, i, x^{(i)}, m^{(i)}, V^{(i)})$. $\sigma^{(i)}$ is called the i-th subsequent signature of $m^{(i)}$ from the root signature σ. We call $\sigma^{(I)} = (\sigma, I, x^{(I)}, m^{(I)}, V^{(I)})$ a signature chain from σ where I is an index set, $m^{(I)} = \{m^{(i)} \mid i \in I\}$, and $V^{(I)} = \{V^{(i)} \mid i \in I\}$.
- RootVerify.: Given a root signature σ and a public key PK, output Valid or Invalid.
- SubVerify.: Given a subsequent signature $\sigma^{(i)}$ from a valid root signature σ, an index $i \in \mathbb{N}$, a random string $x^{(i)}$ and a public key PK, output Valid or Invalid.
- Reveal.: Given two valid i-th subsequent signatures $\sigma^{(i)}$ and $\sigma^{(i)'}$ of distinct messages $m^{(i)}$ and $m^{(i)'}$ from the same root signature σ, output a secret key SK.

KeyGen, RootSign and SubSign are randomized algorithms and RootVerify, SubVerify and Reveal are deterministic algorithms.

2.2. **Security Model of Accountable Signatures.** Let $\mathcal{F}$ be a forger of the accountable signature scheme. $\mathcal{F}$ is allowed to make root signature queries for adaptively chosen messages, and subsequent signature queries for adaptively chosen indices, root signatures, and messages. $\mathcal{F}$ succeeds if it outputs a *new* root signature or a *new* subsequent signature from a valid root signature at the final stage. That is, $\mathcal{F}$ performs an existential forgery. The advantage of $\mathcal{F}$ in this model is defined to be the success probability of the following game.

Setup.: $(PK, SK) \leftarrow \mathsf{KeyGen}(1^\ell)$, where ℓ is the security parameter. The forger $\mathcal{F}$ is given PK.

Queries.: Proceeding adaptively, $\mathcal{F}$ can make $\mathsf{RootSign}$ queries, $\mathsf{SubSign}$ queries and hash queries.

- $\mathsf{RootSign}$ queries: $\mathcal{F}$ chooses a message m and requests a root signature of m. The additional output k_σ is not given to $\mathcal{F}$.
- $\mathsf{SubSign}$ queries: $\mathcal{F}$ chooses a root signature σ obtained by $\mathsf{RootSign}$ queries, an index $i \in \mathbb{N}$, and $m^{(i)}$. $\mathcal{F}$ requests the i-th subsequent signature of $m^{(i)}$ from the root signature σ.
- Hash queries: If necessary, $\mathcal{F}$ can make hash queries also.

Response.: Finally $\mathcal{F}$ wins if $\mathcal{F}$ outputs one of the following:

- A root signature σ of m such that the $\mathsf{RootSign}$ query on m has not been made and $\mathsf{RootVerify}(\sigma, PK) = \mathsf{Valid}$. This type of forger is denoted by $\mathcal{F}_R$ and its advantage $\mathsf{Adv}_{\mathcal{F}_\mathsf{R}}$ is defined by

$$\mathsf{Adv}_{\mathcal{F}_\mathsf{R}} \stackrel{\text{def}}{=} \Pr\left[\begin{array}{ll} \sigma = \mathsf{new}, & \mathsf{Verify}(\sigma, PK) = \mathsf{Valid} : \\ & (PK, SK) \stackrel{\mathrm{R}}{\leftarrow} \mathsf{KeyGen} \\ & \sigma \leftarrow \mathcal{F}^{RS,SS,H}(PK) \end{array}\right]$$

  where RS, SS and H are the oracles of $\mathsf{RootSign}$, $\mathsf{SubSign}$ and hash respectively. The probability is taken over the coin tosses of the KeyGen algorithm, oracle queries and the forger.
- A subsequent signature $\sigma^{(i)}$ of $m^{(i)}$ from a root signature σ such that $\mathsf{SubSign}$ query on $(\sigma, i, m^{(i)})$ has not been made and $\mathsf{SubVerify}(\sigma, i, \sigma_i, PK) = \mathsf{Valid}$. This type of forger is denoted by $\mathcal{F}_S$ and its advantage $\mathsf{Adv}_{\mathcal{F}_S}$ is defined by

$$\mathsf{Adv}_{\mathcal{F}_\mathsf{S}} \stackrel{\text{def}}{=} \Pr\left[\begin{array}{ll} \sigma^{(i)} = \mathsf{new}, & \mathsf{SubVerify}(\sigma, i, \sigma_i, PK) = \mathsf{Valid} : \\ & (PK, SK) \stackrel{\mathrm{R}}{\leftarrow} \mathsf{KeyGen} \\ & (\sigma, i, \sigma_i) \leftarrow \mathcal{F}^{RS,SS,H}(PK) \end{array}\right].$$

In the security analysis, we distinguish two forgers $\mathcal{F}_R$ and $\mathcal{F}_S$, and define the security notion against each forger.

Definition 2. *$\mathcal{F}_S$ ($\mathcal{F}_R$) is a $(t, q_{RS}, q_{SS}, q_H, \epsilon)$-subsequent signature (root signature) forger of an accountable signature scheme in the adaptive chosen-[index, root signature, and message] attack model if $\mathcal{F}_S$ ($\mathcal{F}_R$) runs in time t, it makes at most*

q_{RS} $\mathsf{RootSign}$ queries, q_{SS} $\mathsf{SubSign}$ queries, q_H hash queries and $\mathsf{Adv}_{\mathcal{F}_\mathsf{S}}$ ($\mathsf{Adv}_{\mathcal{F}_\mathsf{R}}$) is at least ϵ.

2.3. **Gap Diffie-Hellman Groups.** Let G be a cyclic group of prime order p with a generator P. To define a Gap Diffie-Hellman (GDH) group, we review some problems regarding the discrete logarithm problem.

Decisional Diffie-Hellman Problem (DDHP).: For given $P, aP, bP, cP \in G$, decide whether $ab \equiv c \pmod p$. If $ab \equiv c \pmod p$, (P, aP, bP, cP) is called a Diffie-Hellman (DH) tuple.

Computational Diffie-Hellman Problem (CDHP).: For given $P, aP, bP \in G$, compute abP.

The advantage of an algorithm $\mathcal{A}$ on the CDHP of G is defined as

$$\mathsf{AdvCDHP}_{\mathcal{A}} \stackrel{\text{def}}{=} \Pr\left[\mathcal{A}(P, aP, bP) = abP : a \stackrel{r}{\leftarrow} \mathbb{Z}_p, b \stackrel{r}{\leftarrow} \mathbb{Z}_p\right].$$

Definition 3. *A group G is a (t, ϵ)-Gap Diffie-Hellman (GDH) group if the DDHP of G can be efficiently computable and there is no algorithm $\mathcal{A}$ whose advantage $\mathsf{AdvCDHP}_{\mathcal{A}}$ is at least ϵ within running time t.*

Remark 4. *Joux and Nguyen* [JN03] *showed that if G has an efficient nondegenerate bilinear map e, called an admissible pairing, the DDHP in G can be solved efficiently by*

$$(P, aP, bP, cP) \text{ is a DH-tuple } \Leftrightarrow\ e(P, cP) = e(aP, bP).$$

3. An Accountable Signature Scheme

3.1. **A Basic Scheme.** We describe a basic scheme for our accountable signature, which is a variant of CC scheme [CC03].

KeyGen.: Given a security parameter ℓ, choose a GDH group G of prime order p where $p \geq 2^\ell$ and a generator P of G. Pick a random $s \in \mathbb{Z}_p$ and set $P_{pub} = sP$. Choose a full domain hash function $H_1 : \{0,1\}^* \times G \to \mathbb{Z}_p$. Choose a random element $Q \in G$ and compute $D = sQ$. The public key is $PK = (G, H_1, P, P_{pub}, Q)$ and the associated secret key is $SK = (D)$.

Sign.: Given a public key PK, a secret key SK, and a message m, pick a random $r \in \mathbb{Z}_p$ and output $\sigma = (m, U, V)$ where $U = rP$, $h = H_1(m, U)$, and $V = rQ + hD$.

Verify.: Given a signature $\sigma = (m, U, V)$ and a public key PK, check whether $(P, Q, U + hP_{pub}, V)$ is a valid DH tuple, where $h = H_1(m, U)$.

3.1.1. *Correctness.* Assume that $Q = aP$. Since $U + hP_{pub} = (r + hs)P$ and $V = rQ + hD = a(r + hs)P$, $(P, Q, U + hP_{pub}, V)$ is a valid DH tuple.

3.2. **An Accountable Signature Scheme.** We describe an accountable signature scheme based on the basic signature scheme. A hash function H_2 is added in KeyGen of the basic scheme.

KeyGen.: Given a security parameter ℓ, choose a GDH group G of prime order p where $p \geq 2^\ell$ and a generator P of G. Pick a random $s \in \mathbb{Z}_p$ and set $P_{pub} = sP$. Choose full domain hash functions $H_1 : \{0,1\}^\ell \times \{0,1\}^* \times G \to \mathbb{Z}_p$ and $H_2 : \{0,1\}^* \times \mathbb{N} \to G$. Choose an element $Q \in G$ and compute $D = sQ$. The public key is $PK = (G, H_1, H_2, P, P_{pub}, Q)$ and the associated secret key is $SK = (D)$.

RootSign.: Given a public key PK, a secret key SK and a message m, pick a random $k \in \mathbb{Z}_p$ and output a root signature $\sigma = (m, U, V)$ in public and k_σ in private where $U = kP$, $h = H_1(0^\ell, m, U)$, $V = kQ + hD$ and $k_\sigma = k$ where 0^ℓ is a ℓ-tuple in which all entries equal zero. k_σ is used for generating subsequent signatures from a root signature σ.

RootVerify.: Given a root signature $\sigma = (m, U, V)$ and a public key PK, check whether $(P, Q, U + hP_{pub}, V)$ is a valid DH tuple, where $h = H_1(0^\ell, m, U)$.

SubSign.: Given a root signature σ, k_σ, an index $i \in \mathbb{N}$, a message $m^{(i)}$ and a secret key D, generate a random string $x^{(i)} \in \{0,1\}^\ell \setminus \{0^\ell\}$ and output $\sigma^{(i)} = (\sigma, i, x^{(i)}, m^{(i)}, V^{(i)})$ where $V^{(i)} = k_\sigma H_2(\sigma, i) + h^{(i)} D$ and $h^{(i)} = H_1(x^{(i)}, m^{(i)}, H_2(\sigma, i))$.

SubVerify.: Given a subsequent signature $\sigma^{(i)} = (\sigma, i, x^{(i)}, m^{(i)}, V^{(i)})$ and a public key Q, check whether

$$(P, U, H_2(\sigma, i) - (h^{(i)}/h)Q, V^{(i)} - (h^{(i)}/h)V)$$

is a valid DH tuple where $h^{(i)} = H_1(x^{(i)}, m^{(i)}, H_2(\sigma, i))$ and $h = H_1(0^\ell, m, U)$.

Reveal.: Given two valid i-th subsequent signatures $V^{(i)} = k_\sigma H_2(\sigma, i) + h^{(i)} D$ and $V^{(i)'} = k_\sigma H_2(\sigma, i) + h^{(i)'} D$ with distinct $h^{(i)}$ and $h^{(i)'}$, output

$$D = (h^{(i)} - h^{(i)'})^{-1}(V^{(i)} - V^{(i)'}).$$

The Reveal algorithm enforces a signer not to generate i-th subsequent signatures more than once.

Correctness.: Assume that $U = k_\sigma P$, $Q = aP$ and $H_2(\sigma, i) = bP$. Then

$$\begin{aligned} H_2(\sigma, i) - (h^{(i)}/h)Q &= bP - (h^{(i)}/h)aP, \\ V^{(i)} - (h^{(i)}/h)V &= k_\sigma bP + h^{(i)} asP - (h^{(i)}/h)(k_\sigma aP + hasP) \\ &= k_\sigma (bP - (h^{(i)}/h)aP) \end{aligned}$$

holds and $(P, U, H_2(\sigma, i) - (h^{(i)}/h)Q, V^{(i)} - (h^{(i)}/h)V)$ is a valid DH tuple.

H_2 is a full-domain hash function from $\{0,1\}^* \times \mathbb{N}$ onto G. When G is a subgroup of an elliptic curve, we can use a MapToPoint in [BLS01].

Remark 5. *For a user A with the public key $PK = (G, H_1, H_2, P, P_{pub}, Q)$ and the associated secret key $SK = (D)$, the signing capability of A can be limited as following:*

(1) *A chooses an index set I and a message m that contains the index set I and some warrant information such as the expiry date and some constraints of this certificate. Then A generates a root signature $\sigma = (m, U, V)$ and keeps k_σ in secret.*

(2) *From the certificate authority (CA), A obtains the certificate* σ_{CA} *of* (PK, σ).
(3) *For the root signature* σ *and the certificate* σ_{CA}*, A generates subsequent signatures* $\sigma^{(i)} = (\sigma, \sigma_{CA}, i, x^{(i)}, m^{(i)}, V^{(i)})$ *for* $i \in I$ *using* k_σ.

For given i-th subsequent signature $\sigma^{(i)} = (\sigma, \sigma_{CA}, i, x^{(i)}, m^{(i)}, V^{(i)})$*, a verifier checks*

(1) σ *and* σ_{CA} *are valid.*
(2) $i \in I$.
(3) $V^{(i)}$ *is valid.*

If the signer signs at most one i-th subsequent signature for each $i \in I$*, the number of possible subsequent signatures of* σ *cannot exceed* k*. If the signer issues i-th subsequent signatures more than once, the secret key of the signer is revealed.*

If the CA is replaced by another signer B, it realize a simple proxy signature with the original signer B and the proxy signer A.

3.3. **Security Analysis.** We show the security of accountable signatures in the following tow theorems. Since we need different approaches for $\mathcal{F}_S$ and $\mathcal{F}_S$, we state the results separately.

Theorem 6. *If there exists a* $(t, q_{RS}, q_{SS}, q_{H_1}, q_{H_2}, \epsilon)$*-subsequent signature forger* $\mathcal{F}_S$*, then we can construct a simulator* $\mathcal{S}$ *which solves the CDHP in* G *with advantage* ϵ' *within* t' *in the random oracle model where*

$$\begin{aligned} \epsilon' &\geq \epsilon \cdot \frac{1}{e} \left\{ 1 - \frac{1}{2^\ell} \cdot q_{RS} \cdot (2q_{H_1} + q_{RS}) \right\} \cdot \frac{1}{(q_{SS}+1)} \\ t' &\leq t + c_G(q_{H_1} + q_{H_2} + 2q_{RS} + 3q_{SS}) \end{aligned}$$

for the security parameter ℓ.

Theorem 7. *If there exists a* $(t, q_{RS}, q_{SS}, q_{H_1}, q_{H_2}, \epsilon)$*-root signature forger* $\mathcal{F}_R$ *with advantage* $\epsilon \geq \frac{7e \cdot q_{H_1}}{2^\ell - A}$*, then we can construct a simulator* $\mathcal{S}$ *which solves the CDHP in* G *with advantage* ϵ' *within* t' *in the random oracle model where*

$$\epsilon' \geq \frac{1}{9} \text{ and } t' \leq \frac{16e \cdot q_{H_1}(t + c_G(q_{H_1} + q_{H_2} + 2q_{RS} + 3q_{SS}))}{\epsilon \cdot (1 - \frac{1}{2^\ell} \cdot A)}$$

and $A = q_{RS} \cdot (2q_{H_1} + q_{RS})$ *for the security parameter* ℓ.

In the above theorems, if ϵ, t, and the number of queries are polynomially bounded with respect to ℓ then we see that so are ϵ' and t'.

3.3.1. *Sketch of the proof.* We construct a simulator $\mathcal{S}$. Since the forger $\mathcal{F}$ is an adaptive chosen-[index, root signature, and message] attacker, it can access to the H_1, H_2, RootSign, and SubSign oracles. For success in the simulation, there should be no difference between $\mathcal{S}$ and the real signer from the view of $\mathcal{F}$. To simulate the RootSign oracle, $\mathcal{S}$ takes a hash value h at random in advance and sets $\sigma = (m, U, V)$, where $U = rP - hP_{pub} = (r - hs)P$ and $V = rQ$. And $\mathcal{S}$ responses with h for a H_1 query on $(0^\ell, m, U)$ if it is possible, reports Failure otherwise. We remark that $\mathcal{S}$ does not know k_σ of $U = k_\sigma P$.

When $\mathcal{F}$ asks the H_2 hash value of (σ, i) for some i, $\mathcal{S}$ responds

$$H_2(\sigma,i) = \begin{cases} t^{(i)}P & \text{with probability } \delta \text{ for a random } t^{(i)}, \\ (h^{(i)}/h)Q + r^{(i)}P & \text{with probability } 1-\delta \text{ for random } r^{(i)} \text{ and } h^{(i)}. \end{cases}$$

with a suitable $\delta \in [0,1]$, which is later adjusted to give the maximal success probability of $\mathcal{S}$. When $\mathcal{F}$ queries to the SubSign oracle, if $H_2(\sigma,i) = t^{(i)}P$ then the simulation fails. If $H_2(\sigma,i) = (h^{(i)}/h)Q + r^{(i)}P$ then $\mathcal{S}$ can produce a valid subsequent signature $\sigma^{(i)} = (\sigma, i, x^{(i)}, m^{(i)}, V^{(i)})$ where $V^{(i)} = (h^{(i)}/h)V + r^{(i)}U$, $h^{(i)} = H_1(x^{(i)}, m^{(i)}, H_2(\sigma,i))$, and $h = H_1(0^\ell, m, U)$.

Suppose $\mathcal{S}$ does not abort the simulation. If the forger $\mathcal{F}$ outputs a valid subsequent signature $\sigma^{(j)} = (\sigma, j, x^{(j)}, m^{(j)}, V^{(j)})$ where $\sigma = (m, U, V)$ and its corresponding $H_2(\sigma, j)$ is of the form $t^{(j)}P$, then $\mathcal{S}$ can compute $D = \frac{1}{h}(V^{(j)} - t^{(j)}U)$ from $V^{(j)} = t^{(j)}U + h^{(j)}D$, since $\mathcal{S}$ knows $t^{(i)}$ and $h^{(i)}$. Note that $V^{(j)} = k_\sigma H_2(\sigma,i) + h^{(j)}D = t^{(j)}U + h^{(j)}D$.

If the forger $\mathcal{F}$ outputs a valid root signature $\sigma = (m, U, V)$ then $\mathcal{S}$ replays the forger $\mathcal{F}$ to get another valid signature $\sigma' = (m, U, V')$. Since the probability that h is equal to h' is negligible, $\mathcal{S}$ can compute $D = (h-h')^{-1}(V - V')$ with overwhelming probability by the Forking lemma [PS00]. □

3.4. **An ID-based Variant.** We describe an ID-based accountable signature scheme. We divide KeyGen of the accountable signature scheme into two steps, Setup and Extract. A new hash function H_3 is introduced in Setup.

Setup.: Given a security parameter ℓ, choose a GDH group G of prime order p where $p \geq 2^\ell$ and a generator P of G. Pick a random $s \in \mathbb{Z}_p$ and set $P_{pub} = sP$. Choose full domain hash functions $H_1 : \{0,1\}^\ell \times \{0,1\}^* \times G \to \mathbb{Z}_p$, $H_2 : \{0,1\}^* \times \mathbb{N} \to G$, and $H_3 : \{0,1\}^* \to G$. The system parameter is $(P, P_{pub}, G, H_1, H_2, H_3)$. The master key is s.

Extract: Given an identity ID, put $Q = H_3(\text{ID})$ and compute $D = sQ$. Finally output Q as a public key and D as a secret key.

RootSign, RootVerify, SubSign, SubVerify: and Reveal As in the accountable signature scheme.

The ID-based variant of an accountable signature scheme can be easily discourage a malicious signer who generates two subsequent signatures of the same index from a root signature. Combining the Theorem 6, Theorem 7 and [CC03, Lemma 1], we can show that our ID-based scheme is secure against existential forgery in adaptive chosen-[ID, index, root signature, and message] attack model under the Diffe-Hellman assumption.

4. Batch Verification of a Signature Chain

Batch cryptography was introduced first by Fiat [Fia89] and batch verifications for DSS and RSA signature scheme have been studied [BGR98, BP00]. Roughly speaking, a batch verifier is an algorithm to verify a set of signatures at one time. So it can be more efficient than verifying signatures individually. If a set of signatures is accepted by the batch verifier, each signature of them should be valid.

On the other hand a weaker notion of batch verification is "screening" [BGR98]. The difference between the batch verification and the screening is that even though a set of signatures is accepted by a screening verifier, each of them is not necessarily a valid signature of its message. But it is guaranteed that every message is authenticated by the signer, which is enough for most of applications.

In our accountable signature scheme, there is a screening verifier, which enables the batch verification of subsequent signatures with the same root signature.

4.0.1. *Batch Verification Test of a Signature Chain.* For a given signature chain $\sigma^{(I)}$, a batch verification of $\sigma^{(I)}$ can be done by checking whether

$$\Big(P, U, \big(\sum H_2(\sigma, i)\big) - \big(\sum h^{(i)}/h\big)Q, \big(\sum V^{(i)}\big) - \big(\sum h^{(i)}/h\big)V\Big)$$

is a valid DH tuple.

When the underlying GDH group G is a subgroup of an elliptic curve and the Tate pairing of G is used for the DDHP, the time complexity of the signature verification is dominated by that of Tate pairing. One execution of the DDH oracle is enough for the batch verification, which is the same as the verification of a individual signature. Thus the batch verifier enables an efficient verification.

Even though $\sigma^{(I)} = (\sigma, I, x^{(I)}, m^{(I)}, V^{(I)})$ passes the batch verification, it does not imply that each signature is valid. For example, if

$$\Big(\sigma, \{1,2\}, x^{(1)}, x^{(2)}, m^{(1)}, m^{(2)}, V^{(1)}, V^{(2)}\Big)$$

is a signature chain with valid signatures $V^{(1)}$ and $V^{(2)}$,

$$\Big(\sigma, \{1,2\}, x^{(1)}, x^{(2)}, m^{(1)}, m^{(2)}, V^{(1)} + R, V^{(2)} - R\Big)$$

passes the batch verification for any random $R \in G$. A permutation of $V^{(I)}$ also passes the test. But we show that the above test provides a screening.

4.1. **Security analysis.** Let $\mathcal{F}_B$ be a forger of the batch verification test of the signature scheme. $\mathcal{F}_B$ can access the H_1, H_2 oracles, RootSign oracle, and SubSign oracle with adaptively chosen indices, root signatures, and messages. Let I_σ be the index set on which $\mathcal{F}_B$ makes subsequent signature queries from a root signature σ. Finally $\mathcal{F}_B$ outputs a signature chain $(\sigma, I, x^{(I)}, m^{(I)}, V^{(I)})$ such that $I \not\subset I_\sigma$. $\mathcal{F}_B$ succeeds if the output is accepted by the batch verifier and the success probability is denoted $\mathsf{Adv}_{\mathcal{F}_\mathsf{B}}$. $\mathcal{F}_B$ is a $(t, q_{RS}, q_{SS}, q_H, n, \epsilon)$-batch verification forger if the advantage $\mathsf{Adv}_{\mathcal{F}_\mathsf{B}}$ of $\mathcal{F}_B$ is at least ϵ within running time t after q_H hash queries, q_{RS} RootSign queries, q_{SS} SubSign queries, and the size of the index set of the output is n. The batch verification is $(t, q_{RS}, q_{SS}, q_H, n, \epsilon)$-secure if there is no $(t, q_{RS}, q_{SS}, q_H, n, \epsilon)$-batch verification forger.

Theorem 8. *If there exists a $(t, q_{RS}, q_{SS}, q_{H_1}, q_{H_2}, n, \epsilon)$-batch verification forger, we can construct a simulator $\mathcal{S}$ which solves the CDHP in G with advantage ϵ' in the random oracle model within t' where*

$$\begin{aligned} \epsilon' &\geq \epsilon \cdot \frac{1}{e}\left\{1 - \frac{1}{2^\ell} \cdot q_{RS} \cdot (2q_{H_1} + q_{RS})\right\} \cdot \frac{1}{(q_{SS} + n - 1)} \\ t' &\leq t + c_G(q_{H_1} + q_{H_2} + 2q_{RS} + q_{H_2} + 3q_{SS} + n + 2) \end{aligned}$$

for the security parameter ℓ.

In the above theorem, if ϵ, t, n and the number of queries are polynomially bounded with respect to ℓ then we see that so are ϵ' and t'.

Remark 9. *If there are two signature chains from the same root signature σ with same index set I, the signing key is revealed by*

$$D = \big(\sum (h^{(i)} - h^{(i)'})\big)^{-1} \big(\sum (V^{(i)} - V^{(i)'})\big)$$

when at least one $h^{(i)}$ is different from $h^{(i)'}$. It may not work if the index set is different. Thus the batch verification is more desirable in verifying the total chain or its segment of a fixed length.

5. Conclusion and Further Studies

We proposed a new notion of accountable signatures in which each subsequent signature corresponds to a unique index and so forms a signature chain. It can be applied to a k-times signature to restrict the signing ability by k-times. This property is desirable for many applications including electronic coupons, proxy signatures or group signatures. It would be an interesting problem to design a group signature in which each member can sign k-times, otherwise his identity is revealed.

When all the signatures issued by a signer are publicly obtainable as in bulletin board, accountable signatures can be used to enforce the signer to sign a message with a serial index. In that case, any removal, replacement, insertion, or order change is not allowed. Note that we do not have to restrict the number of signatures for this purpose, which is not covered by k-times signatures.

Accountable signatures are effective when the exposure of a private key does badly damage the owner of the public key. In most applications, only detection of misbehavior is enough by requiring a mechanism relying on external authority such as police or court to redeem someone's right. Our solution discourages a signer's misbehaving more efficiently. More concrete applications should be studied further.

References

[BGLS03] D. Boneh, C. Gentry, B. Lynn, and H. Shacham. Aggregate and verifiably encrypted signatures from bilinear maps. In *Advances in Cryptology - Eurocrypt 2003*, LNCS Vol. 2656, pp. 416–432, Springer-Verlag, 2003.

[BGR98] M. Bellare, J. Garay, and T. Rabin. Fast batch verification for modular exponentiation and digital signatures. In *Advances in Cryptology - Eurocrypt'98*, LNCS Vol. 1403, pp. 236–250, Sringer-Verlag, 1998.

[BLS01] D. Boneh, B. Lynn, and H. Shacham. Short signature from the Weil pairing. In *Advances in Cryptology - Asiacrypt 2001*, LNCS Vol. 2248, pp. 514–531, Springer-Verlag, 2001. The extended version is available at `http://crypto.stanford.edu/~dabo/abstracts/weilsigs.html`.

[BP00] C. Boyd and C. Pavlovski. Attacking and repairing batch verification schemes. In *Advances in Cryptology - Asiacrypt 2000*, LNCS Vol. 1976, pp. 58–71, Springer-Verlag, 2000.

[CC03] J. Cha and J. Cheon. An ID-based signature from gap-Diffie-Hellman groups. In *Public Key Cryptography - PKC 2003*, LNCS Vol. 2567, pp. 18–30, Springer-Verlag, 2003.

[EO94] T. Eng and T. Okamoto. Single-term divisible coins. *Advances in Cryptology - Eurocrypt '94* LNCS Vol. 950, pp. 306–319, Springer-Verlag, 1995.

[Fia89] A. Fiat. Batch RSA. *J. Cryptology*, Vol. 10, No. 2, pp. 75–88, Springer-Verlag, 1997. A preliminary version appeared in *Advances in Cryptology - Crypto'89*, LNCS Vol. 435, pp. 175–185, Springer-Verlag, 1989.

[GQ88] L. Guillou and J. Quisquater. A Practical Zero-Knowledge Protocol Fitted to Security Microprocessors Minimizing Both Transmission and Memory. In *Adcances in Cryptology - Eurocrypt '88*, LNCS Vol. 330, pp. 123-128, Springer-Verlag, 1988.

[HKLL03] J. Hwang, H. Kim, D. Lee, and J. Lim. Digital signature schemes with restriction on signing capability. In *Information Security and Privacy - ACISP 2003*, LNCS Vol. 2727, pp. 324–335, Springer-Verlag, 2003.

[JN03] A. Joux and K. Nguyen. Separating decision Diffie-Hellman from computational Diffie-Hellman in cryptographic groups. *J. Cryptology*, Vol. 16, No. 4, pp. 239–247, Springer-Verlag, 2003.

[LL94] C.H. Lim and P.J. Lee. More flexible exponentiation with precomputation. In *Advences in cryptology - Crypto'94*, LNCS Vol. 839, pp. 95–107, Springer-Verlag, 1994.

[Mao96] W. Mao. Lightweight micro-cash for the Internet. In *Computer Security - ESORICS'96*, LNCS Vol. 1146, pp. 15–32, Springer-Verlag, 1996.

[Mer89] R.C. Merkle. A certified digital signature. In *Advences in cryptology - Crypto'89*, LNCS Vol. 435, pp. 218–238, Springer-Verlag, 1990.

[OO89] T. Okamoto, K. Ohta. Disposable Zero-Knowledge Authentication and Their Applications to Untraceable Electronic Cash. *Advences in cryptology - Crypto'89*, LNCS Vol.435, pp.481-496, SpringerVerlag, Heidelberg 1990.

[PS00] D. Pointcheval and J. Stern. Security arguments for digital signatures and blind signatures. *J. Cryptology*, Vol. 13, No. 3, pp. 361–396, Springer-Verlag, 2000.

[Rab78] M.O. Rabin. Digitalized signatures. *Foundations of Secure Computations*, Academic Press, pp. 155–168, 1978.

Appendix: Security Proof

Proof of Theorem 6. Let an instance of CDHP, aP and bP be given. We construct an algorithm $\mathcal{S}$ which produces abP using the forger $\mathcal{F}$ of the signature scheme. The forger $\mathcal{F}$ can access the H_1, H_2, RootSign and SubSign oracles. $\mathcal{S}$ simulates a real signer to get a valid signature from the forger $\mathcal{F}$. If $\mathcal{S}$ does not abort the simulation and $\mathcal{F}$ outputs a valid signature, then $\mathcal{S}$ produces abP.[1]

In setup, $\mathcal{S}$ puts $P_{pub} = aP$ and $Q = bP$. $\mathcal{S}$ manages H_1-list, RootSign-list and SubSign-list. H_1-list consists of $\langle m, U, h = H_1(x, m, U)\rangle$. RootSign-list consists of $\langle m, U, h, r, V\rangle$ (or $\langle \sigma, r\rangle$) and SubSign-list consists of $\langle \sigma, i, c^{(i)}, r^{(i)}, x^{(i)}, m^{(i)}, U^{(i)}, h^{(i)}, V^{(i)}\rangle$ where $U^{(i)}$ and $h^{(i)}$ mean $H_2(\sigma, i)$ and $H_1(x^{(i)}, m^{(i)}, U^{(i)})$ respectively. At first, all lists are blank. On the queries from $\mathcal{F}$, $\mathcal{S}$ behaves as following.

[1]During the simulation, we consider a root signature σ as the form (m, U, h, V) containing the H_1 hash value $h = H_1(0^\ell, m, U)$ and a subsequent signature $\sigma^{(i)}$ from σ as the form $(\sigma, i, x^{(i)}, m^{(i)}, h^{(i)}, V^{(i)})$.

: H_1 **Queries**
- Input: (x, m, U)
- If (x, m, U) appears in H_1-list or RootSign-list or SubSign-list
  - Return the corresponding value h
- Otherwise
  - $h \xleftarrow{R} \mathbb{Z}_p$ return h
  - Store $\langle x, m, U, h\rangle$ in H_1-list
- Output : (x, m, U, h)

: RootSign **Queries**
- Input: a message m
- $h, r \xleftarrow{R} \mathbb{Z}_p$
- $U \leftarrow rP - hP_{pub}$
- If $(0^\ell, m, U)$ appears in H_1-list or RootSign-list
  - Report Failure and terminate
- Otherwise
  - $V \leftarrow rQ$ and return $\sigma = (m, U, h, V)$
  - Store $\langle m, U, h, r, V\rangle$ in RootSign-list
- Output : (m, U, h, V) or Failure

In H_2 queries, we may restrict the input of $\mathcal{F}$ to root signatures obtained from RootSign queries. Similarly the input in SubSign query may be restricted.

: H_2 **Queries**
- Input: (σ, i) where $\sigma = (m, U, h, V)$ appears in RootSign-list
- If (σ, i) appears in SubSign-list, return the corresponding value $U^{(i)}$
- Otherwise
  - $c^{(i)} \leftarrow \{0, 1\}$ with $\Pr[c^{(i)} = 0] = \delta$ and $\Pr[c^{(i)} = 1] = 1 - \delta$ for some $\delta \in [0, 1]$
  - If $c^{(i)} = 0$
    - $t^{(i)} \xleftarrow{R} \mathbb{Z}_p$
    - $U^{(i)} \leftarrow t^{(i)}P$, return $U^{(i)}$
    - Store $\langle \sigma, i, 0, t^{(i)}, -, -, U^{(i)}, -, -\rangle$ in SubSign-list
  - If $c^{(i)} = 1$
    - $r^{(i)}, h^{(i)} \xleftarrow{R} \mathbb{Z}_p$
    - $U^{(i)} \leftarrow (h^{(i)}/h)Q + r^{(i)}P$, return $U^{(i)}$
    - $V^{(i)} \leftarrow (h^{(i)}/h)V + r^{(i)}U$
    - Store $\langle \sigma, i, 1, r^{(i)}, -, -, U^{(i)}, h^{(i)}, V^{(i)}\rangle$ in SubSign-list
- Output : $(\sigma, i, U^{(i)})$

: SubSign **Queries**
- Input: $(\sigma, i, m^{(i)})$ where σ appears in RootSign-list and $\mathcal{F}$ made H_2 query on (σ, i) ahead
- If $c^{(i)} = 1$
  - If $(x, m^{(i)}, U^{(i)})$ appears in H_1-list, choose a random $x^{(i)} \in \{0,1\}^\ell \setminus \{0^\ell\}$ such that $x^{(i)} \neq x$
  - Otherwise choose a random $x^{(i)} \in \{0,1\}^\ell \setminus \{0^\ell\}$
  - Return $(\sigma, i, x^{(i)}, m^{(i)}, h^{(i)}, V^{(i)})$ where $h^{(i)} = H_1(x^{(i)}, m^{(i)}, H_2(\sigma\ , i))$
  - Fill in $x^{(i)}$ and $m^{(i)}$ in SubSign-list
- If $c^{(i)} = 0$
  - Report Failure and terminate
- Output : $(\sigma, i, x^{(i)}, m^{(i)}, h^{(i)}, V^{(i)})$ or Failure

Now we consider some collisions of this simulation. If a collision occurs with any queries, the simulation reports failure and terminates. We know that the two collisions may happen: at first, when $\mathcal{S}$ makes the (m, U) pair in RootSign queries, the (m, U) pair may appear in H_1-list. In this case, a collision H_1 query with RootSign query happens with probability at most $q_{H_1} \cdot q_{RS} \cdot \frac{2}{2^\ell}$, where ℓ is the security parameter. Secondly, when $\mathcal{S}$ makes the (m, U) pair in RootSign queries (m, U) may appear in RootSign-list. In this case, (m, U) collides with the answer of other RootSign queries. The probability of this collision is at most $\frac{q_{RS}^2}{2} \cdot \frac{2}{2^\ell}$.

Suppose $\mathcal{S}$ does not abort during the above simulation and $\mathcal{F}$ outputs a new subsequent signature $\sigma^{(i)} = (\sigma, i, x^{(i)}, m^{(i)}, h^{(i)}, V^{(i)})$ where $\sigma = (m, U, h, V)$ is its root signature obtained from RootSign queries. (σ, i) should be in SubSign-list and $\mathcal{S}$ checks whether $c^{(i)}$ corresponding to $\sigma^{(i)}$ is zero or one.

- If $c^{(i)} = 1$, then report Failure and terminate.
- If $c^{(i)} = 0$, then $\mathcal{S}$ can compute

$$D = \frac{1}{h^{(i)}}(V^{(i)} - k_\sigma U^{(i)}) = \frac{1}{h^{(i)}}(V^{(i)} - t^{(i)}U).$$

Remark $k_\sigma U^{(i)} = k_\sigma t^{(i)} P = t^{(i)} U$ and $\mathcal{S}$ knows $t^{(i)}$.

Now we discuss the success probability and the running time of $\mathcal{S}$. To solve the CDHP the following should be hold: $\mathcal{S}$ does not abort the simulation, $\mathcal{F}$ outputs a new valid subsequent signature and the value $c^{(i)}$ in the output of $\mathcal{F}$ is zero. Let each event be as following:

Event $\mathcal{E}_1$.: No collision and failure happens during the simulation.
Event $\mathcal{E}_2$.: $\mathcal{F}$ generates a valid subsequent signature.
Event $\mathcal{E}_3$.: $\mathcal{E}_2$ holds and the value of the random coin corresponding to the output of $\mathcal{F}$ is zero.

The probability for $\mathcal{S}$ to solve the CDHP equals $\Pr[\mathcal{E}_1 \wedge \mathcal{E}_3]$ and

$$\Pr[\mathcal{E}_1 \wedge \mathcal{E}_3] = \Pr[\mathcal{E}_1] \cdot \Pr[\mathcal{E}_2|\mathcal{E}_1] \cdot \Pr[\mathcal{E}_3|\mathcal{E}_1 \wedge \mathcal{E}_2].$$

To happen the event $\mathcal{E}_1$ there is no collision between the queries and $c^{(i)}$ should be one, for all SubSign queries. Thus the probability if the event $\mathcal{E}_1$ is

$$\Pr[\mathcal{E}_1] \geq \left(1 - q_{H_1} \cdot q_{RS} \cdot \frac{2}{2^\ell}\right) \cdot \left(1 - \frac{q_{RS}^2}{2} \cdot \frac{2}{2^\ell}\right) \cdot (1-\delta)^{q_{SS}}.$$

We can simplify the lower bound of success probability value,

$$\Pr[\mathcal{E}_1] \geq (1-\delta)^{q_{SS}} \left\{1 - \frac{1}{2^\ell} \cdot q_{RS} \cdot (2q_{H_1} + q_{RS})\right\}.$$

If the simulator $\mathcal{S}$ does not abort during the queries, anyone can not distinguish the simulator $\mathcal{S}$ from the real signer, thus $\Pr[\mathcal{E}_2|\mathcal{E}_1] \geq \epsilon$. Remark that ϵ is the advantage of the forger $\mathcal{F}$. If the event $\mathcal{E}_1$ holds and the forger $\mathcal{F}$ outputs a valid subsequent signature, a matter of concern is only whether the value of random coin is zero or one. Since the hash values are independent to the random coin of $\mathcal{F}$'s view, thus $\Pr[\mathcal{E}_3|\mathcal{E}_1 \wedge \mathcal{E}_2] \geq \delta$. To summarize this, we get the probability for $\mathcal{S}$ to solve the CDHP:

$$\Pr[\mathcal{E}_1 \wedge \mathcal{E}_3] \geq \epsilon \cdot (1-\delta)^{q_{SS}} \left\{1 - \frac{1}{2^\ell} \cdot q_{RS} \cdot (2q_{H_1} + q_{RS})\right\} \cdot \delta.$$

To maximize this probability, $\mathcal{S}$ chooses $\delta = 1/(q_{SS}+1)$. Finally, we get $\epsilon' \geq \epsilon \cdot \frac{1}{e}\left\{1 - \frac{1}{2^\ell} \cdot q_{RS} \cdot (2q_{H_1} + q_{RS})\right\} \cdot \frac{1}{(q_{SS}+1)}$.

The running time of $\mathcal{S}$ is longer than that of the forger $\mathcal{F}$ since $\mathcal{S}$ performs computations on the group G to respond queries of $\mathcal{F}$. In simulation, $\mathcal{S}$ responds at most $(q_{H_1}+q_{RS}+q_{SS})$ H_1 queries, $(q_{H_2}+q_{SS})$ H_2 queries, q_{RS} RootSign queries, and q_{SS} SubSign queries to $\mathcal{F}$. So, the total running time of $\mathcal{S}$ is at most $t+c_G(q_{H_1}+q_{H_2}+2q_{RS}+3q_{SS}) \leq t'$ where c_G means the time for the group operation in G. □

Proof of Theorem 7. Suppose $\mathcal{S}$ does not abort the same simulation with Theorem 6 and $\mathcal{F}$ outputs a new root signature $\sigma = (x, m, U, h, V)$, then $\mathcal{S}$ replays this simulation except only the H_1 hash value h' of (m, U) which is distinct from h given before. Then $\mathcal{F}$ outputs another valid signature $\sigma' = (m, U, x, h', V')$. Finally, $\mathcal{S}$ can compute

$$D = (h - h')^{-1}(V - V').$$

Differently from the no-message attack case, there is some risk of abort in the simulation: collisions between queries and Failure in SubSign queries. According to the proof of Theorem 6, if $\epsilon \geq \frac{7e \cdot q_{H_1}}{2^\ell - A}$

$$\begin{aligned}\Pr[\mathcal{S} \text{ succeeds and no-fails}] &\geq \epsilon \cdot \frac{1}{e}\left\{1 - \frac{1}{2^\ell} \cdot q_{RS} \cdot (2q_{H_1} + q_{RS})\right\} \\ &\geq 7q_{H_1}/2^\ell\end{aligned}$$

So we can apply the Forking lemma [PS00, Lemma 2]. Finally, $\mathcal{S}$ can solve the CDHP with probability at least $\frac{1}{9}$ within $t' \leq 16eq_{H_1} \cdot (t + c_G(q_{H_1} + 2q_{RS} + q_{H_2} + 3q_{SS}))/\{\epsilon \cdot \left(1 - \frac{1}{2^\ell} \cdot q_{RS} \cdot (2q_{H_1} + q_{RS})\right)\}$ where c_G means the time for the group operation in G. □

Proof of Theorem 8. $\mathcal{S}$ executes a simulation as the same simulation in the standard case. Suppose $\mathcal{S}$ dose not abort the simulation and the forger $\mathcal{F}_B$ outputs a valid signature chain $\sigma^{(I)} = (\sigma, I, m^{(I)}, H^{(I)}, V^{(I)})$ where $\sigma = (m, U, h, V)$ is a root signature, I is an index set, $m^{(I)} = \{m^{(i)} \mid i \in I\}$, $H^{(I)} = \{h^{(i)} \mid i \in I\}$ and $V^{(I)} = \{V^{(i)} \mid i \in I\}$. $\mathcal{S}$ checks the value of the random coin $c^{(i)}$ corresponding to the $\sigma^{(i)}$ for each $i \in I$.

- If $c^{(i)} = 1$ for all $i \in I$ then report Failure and terminate.
- If $c^{(i)} = 0$ is blank for some $i \in I$, $\mathcal{S}$ can compute the secret key D as following. Let $J = \{j \in I \mid c^{(j)} = 1\}$, then $\mathcal{S}$ knows all values $t^{(i)}$ for $i \in I - J$ and can compute $V^{(j)}$s for all $j \in J$, finally gets

$$D = \left(\sum_{i \in I-J} h^{(i)} \right)^{-1} \left(V - \sum_{j \in J} V^{(i)} - \sum_{i \in I-J} t^{(i)} U \right)$$

Now consider the success probability of $\mathcal{F}_B$ to solve the CDHP. Similar to the standard scheme, the success probability of $\mathcal{F}_B$ is

$$\Pr[\mathcal{E}_1 \wedge \mathcal{E}_3] \geq \epsilon \cdot \left\{ 1 - \frac{1}{2^\ell} \cdot q_{RS} \cdot (2q_{H_1} + q_{RS}) \right\} \cdot \delta \cdot (1-\delta)^{q_{SS}+n-1},$$

when $|I| = n$. To maximize this probability, $\mathcal{S}$ chooses $\delta = \frac{1}{(q_{SS}+n)}$. Finally, $\mathcal{S}$ solves the CDHP with probability at least $\epsilon \cdot \frac{1}{e} \left\{ 1 - \frac{1}{2^\ell} \cdot q_{RS} \cdot (2q_{H_1} + q_{RS}) \right\} \cdot \frac{1}{(q_{SS}+n-1)}$.
The running time of $\mathcal{S}$ is longer than that of the forger $\mathcal{F}_B$ since $\mathcal{S}$ performs computations on the group G to respond queries of $\mathcal{F}_B$. In simulation, $\mathcal{S}$ responds at most $(q_{H_1} + q_{RS} + q_{SS})$ H_1 queries, q_S RootSign queries, $(q_{H_2} + q_{SS})$ H_2 queries, q_{SS} SubSign queries to $\mathcal{F}_B$ and n-group operations. So, the total running time of $\mathcal{S}$ is at most $t + c_G(q_{H_1} + q_{H_2} + 2q_{RS} + 3q_{SS} + n + 2)$ where c_G means the time for the group operation in G. □

ISaC, Department of Mathematical Sciences,, Seoul National University, Seoul, 151-747, Korea

E-mail address: {jhcheon,whkim,jin25}@math.snu.ac.kr