Metered Signatures
- How to restrict the Signing Capability -

Woo-Hwan Kim, HyoJin Yoon, and Jung Hee Cheon

Abstract: **We propose a new notion of *metered signatures*. Metered signature is an extension of *k-times signatures* in which a signer can generate only k signatures. However, the restriction of metered signatures can be more elaborate: It can be used k-times every day or to preserve the order of signed messages in some applications. Any violation of this regulation reveals a secret key or the signature on a predetermined message. The applications includes proxy signatures, limited free downloads, and the rating web site. We give two instances of metered signatures: one is based on the computational Diffie-Hellman problem (CDHP) using a bilinear map and the other is based on the RSA problem. In both schemes, the signature and certificate size and the verification cost are constant with respect to k. Further, we show that the proposed metered signatures admit batch verification of many signatures almost at one verification cost with small security loss.**

Index Terms: **digital signature, k-times, proxy signature, authentication, Diffie-Hellman problem, RSA problem**

I. INTRODUCTION

In the real world, we can easily find the situation that only limited number of actions are approved: One can withdraw money from an ATM (Automated Teller Machine) up to predetermined times a day. A book of coupons consists of fixed number of tickets. Memberships at a golf club are highly priced by limited issuing. In digital signatures, 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 in many applications. In proxy signatures, a signer delegates the signing right to the proxy signer, but he may want to restrict the number of proxy signatures. A contents provider may advertise his web site by permitting free downloads only by restrict times. Electronic coupons or admission tickets with limited use are another examples.

A. Revealing a Secret Key or a Signature

We adopt an approach that a secret key or a signature on a predetermined message is revealed if the signer generates too many signatures. Revealing the secret key would be better for more strict restriction. If the secret key is the one that is used for important applications (for example, the secret key for banking account), it can be more effective. For more mild applications, we may take the signature revealing. For example, the signer commits an unsigned check to get a k-times usable free ticket and the check is automatically signed after more than k-times signing without any further protocol.

Woo-Hwan Kim is with the attached institute of ETRI, Korea, email: whkim5@ensec.re.kr.

HyoJin Yoon is with Samsung SDS IT R&D center, Korea, email: hj1230.yoon@samsung.com

Jung Hee Cheon is with the department of mathematical sciences, Seoul National University, Korea, email: jhcheon@snu.ac.kr

B. Our Contributions

We introduce a new notion of *metered signature scheme* consisting of two kinds of signature schemes: *a root signature scheme* and *a subsignature scheme*. The root signature specifies an index set, each element of which is used to generate only one subsignature. Multiple uses of one index reveals some secret information as stated above. The index set can be various including several typical types: 1) $\mathbb{Z}_k$ admits only k signatures; 2) $\mathbb{Z}_k \times Date$ admits k signatures for every day; and 3) $\mathbb{Z}_{\geq 0}$ admits infinite number of signatures, but every signature is ordered by a unique index.

We may choose a proper type of index set for a given application. We note that the regulation should be certified by the third party, for example, by the service provider and is accompanied by each subsignature.

We present two instances of metered signatures: one is based on the computational Diffie-Hellman problem (CDHP) using a bilinear map and the other is based on the RSA problem. Further we show that the first scheme gives an ID-based metered signature. Differently from the previous k-times signature schemes, the size of signature, certificate and the verification cost of our scheme are constant with respect to k. We propose a formal definition and a security model for metered signatures and prove that our schemes are secure against adaptive chosen-[$spec$, index, and message] attack in the random oracle model. Also we realize the batch verifications of our signature schemes with provable security.

C. Related Works

One-time signature [15] 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. Hwang *et al.* [9] suggested a *k-times* signature in which the certificate size increases linearly with respect to k. If we use a Merkle hash tree for authenticating random commitments, we can obtain a k-times signature scheme with a certificate of a constant size [11]. Its signature size and verification cost are proportional to $\log k$. The k-times restriction technique was also developed for electronic cash in which more than k-times authentications reveal user's identity [14], [6]. Recently several k-times anonymous authentication schemes (k-TAA) are proposed [17], [13], [18]. In k-TAA, each user can be authenticated anonymously by the application provider(AP) for a bounded number of times. One may convert a k-TAA scheme to a metered sig-

1229-2370/03/$10.00

nature scheme by use of message hash as a random challenge in the authentication. But the efficiency(the size of public key of AP, computational cost, authentication message size) of proposed k-TAA schemes depends on the number of users and k (linear or log scale) while the efficiency of the schemes in this paper does not.

We extend the authentication scheme by Okamoto and Ohta [14] to the metered signature scheme based on the RSA problem with security proof.

D. Organization of this paper

The rest of this paper is organized as follows. In Section II, we provide the formal model and security model of metered signatures. Two metered signature schemes are instantiated in Section III and Section IV. One is based on the CDHP using a bilinear map and the other is based on the RSA problem. We show that both the scheme are secure in the random oracle model. In Section V, we present secure batch verifications of two metered signatures. In Section VI, we investigate applications of metered signatures. We conclude the paper in Section VII.

II. METERED SIGNATURES

A metered signature scheme consists of two kinds of signature schemes: a root signature scheme and a subsignature scheme. There are three players (a signer, a certifier and a verifier) in a metered signature. The signer chooses some specification ($spec$), which describes the restriction on the signing capability, and obtains a certificate on $spec$ from the certifier.

A. The Formal Model of Metered Signatures

Definition 1: A metered signature consists of 7-tuple of algorithms (KeyGen, RootSign, RootVerify, Cert, SubSign, SubVerify, Reveal) such that:

1. KeyGen is a probabilistic algorithm which takes as input a security parameter ℓ and returns a secret key SK and public parameters $param$ that includes a public key PK.
2. RootSign takes a message m and SK as input and returns a root signature (m, σ).
3. RootVerify takes as input a m, a root signature σ and $param$ and outputs Valid or $\perp$.
4. Cert takes a root signature on $spec$, which includes $param$ and the specification of the usage of SubSign from $spec$ and returns a certificate σ_{CA} on $spec$ if the signature is valid. When the user (signer) generates the $spec$, the user may keep a (secret) parameter t_{spec} associated with $spec$ for SubSign if necessary.
5. SubSign takes σ_{CA}, SK, $spec$ (and t_{spec} if necessary), an index i and a message m_i as input and outputs an i-th metered signature $(spec, \sigma_{CA}, i, m_i, \sigma_i)$.
6. SubVerify takes as input $param$, an i-th metered signature $(spec, \sigma_{CA}, i, m_i, \sigma_i)$ and outputs Valid or $\perp$.
7. Reveal takes as input two metered signatures from $spec$ with the same index and outputs the secret key SK.

The certificate and $spec$ confine the signer to generate subsignatures in a certain way: $spec$ specifies the index set which can be used in subsignatures and each subsignature is accompanied by a unique index. Subsignature scheme is designed so as to reveal the secret key of the root signature scheme if the two subsignatures with the same index are gathered. The index set is not only integers, but also some strings. Some typical examples are as follows:

1. $I_1 = \{i : i$ is an integer from 1 to $k\}$
2. $I_2 = \{(i, j) : i$ is an integer from 1 to k and j is a date between January 1st, 2005 to December 31st, 2005$\}$
3. $I_3 = \{i : i$ is a positive integer$\}$

I_1 allows only k signatures and I_2 does k signatures per each day in 2005. On the other hand, I_3 allows infinite number of signatures, but every signature is ordered with its unique index. The index set can be more complicated including $[1, k_1] \times [1, k_2] \times \cdots \times [1, k_n]$ where $[1, k] = \{i |\ i$ is an integer between 1 and $k\}$. Further each k_i can be infinite and some index set can be replaced by strings.

In Section III and IV, we restrict the signing capability by limiting the randomness used in subsignatures and ID-based signatures such as [5] and [8] are exploited with some modification to realize the metered signatures.

B. Security Model of the Metered Signatures

Let $\mathcal{F}$ be a forger of the metered signatures. $\mathcal{F}$ is allowed to make root signature queries for adaptively chosen messages, and subsignature queries for adaptively chosen indices, root signatures, and messages. $\mathcal{F}$ succeeds if it outputs a *new* root signature or a *new* subsignature. 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.** $(param, SK) \leftarrow \mathsf{KeyGen}(1^\ell)$, where ℓ is the security parameter. The forger $\mathcal{F}$ is given $param$.
- **Queries.** Proceeding adaptively, $\mathcal{F}$ can make RootSign queries, SubSign queries and hash queries.
  - RootSign queries: $\mathcal{F}$ chooses a $spec$ and requests a root signature on $spec$.
  - SubSign queries: $\mathcal{F}$ makes SubSign queries on $(spec, i, m_i)$ and is given the subsignature together with the certificate on the $spec$.
  - Hash queries: If necessary, $\mathcal{F}$ can make hash queries also.
- **Response.** Finally $\mathcal{F}$ wins if $\mathcal{F}$ outputs one of the followings:
  - A valid root signature on m such that $\mathcal{F}$ has not made the RootSign query on m before. This type of forger and its advantage are denoted by $\mathcal{F}_R$ and $\mathsf{Adv}_{\mathcal{F}_\mathsf{R}}$ respectively.
  - A valid subsignature $(spec, \sigma_{CA}, j, m_j, \sigma_j)$ such that $\mathcal{F}$ has not made the SubSign query on $(spec, j, m_j)$ before. This type of forger and its advantage are denoted by $\mathcal{F}_S$ and $\mathsf{Adv}_{\mathcal{F}_S}$ respectively.

Definition 2: $\mathcal{F}_S$ ($\mathcal{F}_R$) is a $(t, q_{RS}, q_{SS}, q_H, \epsilon)$-subsignature (root signature) forger of a metered 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} RootSign queries, q_{SS} SubSign queries, q_H hash queries and $\mathsf{Adv}_{\mathcal{F}_\mathsf{S}}$ ($\mathsf{Adv}_{\mathcal{F}_\mathsf{R}}$) is at least ϵ.

III. A CONSTRUCTION: BASED ON THE CDHP

Let G be a cyclic group of prime order p with a generator P. We recall some problems related to the discrete logarithm.

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

$(\mathrm{mod}\ p)$, (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

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

Definition 3: The CDHP of G is (t, ϵ)-hard if there is no algorithm $\mathcal{A}$ whose advantage $\mathsf{AdvCDHP}_{\mathcal{A}}$ is at least ϵ within running time t.

Definition 4: Let $G = \langle P \rangle$ and G_T be groups with $|G| = |G_T| = p$ for some prime p. A map $\tilde{e} : G \times G \rightarrow G_T$ is called an admissible bilinear map if

1. (Nondegenerate) $\tilde{e}(P, P) \neq 1$.
2. (Bilinear) $\tilde{e}(aP, bP) = \tilde{e}(P, P)^{ab}$ for all $a, b \in \mathbb{Z}_p$.
3. (Efficient) $\tilde{e}$ is efficiently computable.

Remark 1: Joux and Nguyen [10] showed that if G has an admissible bilinear map $\tilde{e}$, the DDHP in G can be solved by

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

A. A Metered Signature based on the CDHP

RootSign and RootVerify are identical to those of the Cha and Cheon [5] ID-based signature scheme except that the P_1 (corresponding to Q_{ID} in their scheme) is not a hash value of an identity *ID* but only public key.

• KeyGen. Given a security parameter ℓ, take two abelian groups $G = \langle P \rangle$ and G_T such that $|G| = |G_T| = p$ for some prime $p \geq 2^{\ell}$ and an admissible bilinear map $\tilde{e} : G \times G \rightarrow G_T$. Randomly choose $P_1 \in G$ and $s \in \mathbb{Z}_{\mathrm{l}}^*$. Compute $P_2 = sP$ and $D = sP_1$. (P, P_1, P_2) is the public key and D is the secret key. Also specifies full domain hash functions $H_1 : \{0,1\}^* \rightarrow \mathbb{Z}_p$ and $H_2 : \{0,1\}^* \rightarrow G$. The $param$ includes $(G, G_T, \tilde{e}, P, P_1, P_2, H_1, H_2)$.

• RootSign. Given the $param$, the secret key D and a message m, pick a random $r \in \mathbb{Z}_p$ and output a root signature (m, U, V) where $U = rP_1$ and $V = (r + H_1(m, U))D$.

• RootVerify. Given the root signature (m, U, V) and $param$, check whether $\tilde{e}(P, V) = \tilde{e}(U + H_1(m, U)P_1, P_2)$.

• Cert. The signer takes a (secret) random $t \in \mathbb{Z}_p^*$ and an index set subsignatures. The message $spec$ of the root signature is the index set, $W = tP$ and $param$. He sends the root signature $(spec, U, V)$ on $spec$. The certifier verifies the root signature and returns a certificate σ_{CA} on $spec$. t means the t_{spec} in definition 1.

• SubSign. Given the secret key D, t in $W = tP$, an index $i \in \mathbb{N}$, $spec$, a message m_i and σ_{CA}, the signer chooses a random string $x_i \in \{0,1\}^{\ell/2}$ and computes $\sigma_i = tH_2(spec, i) + h_i D$ where $h_i = H_1(i, x_i, m_i, spec)$. $(spec, \sigma_{CA}, i, x_i, m_i, \sigma_i)$ is an i-th subsignature from $spec$.

• SubVerify. Given the i-th subsignature

$$(spec, \sigma_{CA}, i, x_i, m_i, \sigma_i),$$

the verifier checks the following:

1. Checks whether i is valid with respect to $spec$.
2. Checks whether σ_{CA} is valid.
3. Computes $h_i = H_1(i, x_i, m_i, spec)$ and checks whether

$$\tilde{e}(P, \sigma_i) = \tilde{e}(W, H_2(spec, i)) \cdot \tilde{e}(h_i P_1, P_2).$$

• Reveal. Suppose two valid i-th signatures $\sigma_i = tH_2(spec, i) + h_i D$ and $\sigma_i' = tH_2(spec, i) + h_i' D$ are given. Then the secret key D can be computed by $D = (h_i - h_i')^{-1}(\sigma_i - \sigma_i')$.

$W = tP$ is included in $spec$ and t is necessary to make subsignatures from $spec$. t in W should be kept secret. Otherwise the secret key D can be computed from a subsignatures from $spec$ which include $W = tP$. One role of W is binding $spec$ and subsignatures. Another is that signer can use an index set more than once by use of different W in $spec$.

The correctness of the signature schemes follows immediately from the observations:

• RootVerify: $\tilde{e}(P, V) = \tilde{e}(P, (r + H_1(m, U))D) = \tilde{e}(U + H_1(m, U)P_1, P_2)$.

• SubVerify: $\tilde{e}(P, \sigma_i) = \tilde{e}(P, tH_2(spec, i) + h_i D) = \tilde{e}(W, H_2(spec, i)) \cdot \tilde{e}(h_i P_1, P_2)$.

H_2 is a full-domain hash function from $\{0,1\}^*$ onto G. When G is a subgroup of an elliptic curve, we can use a MapToGroup in [3].

B. Security Analysis

The security proof is in the random oracle model under the CDH assumption. We consider two signature forgers defined in Section II-B. Since the root signature is the same as Cha-Cheon signature, we only analyze the security against the subsignature forger.

Theorem 1: If there exists a $(t, q_{RS}, q_{SS}, q_H, \epsilon)$ subsignature forger $\mathcal{F}$ of the CDHP based metered signature, then we can construct a simulator $\mathcal{S}$ which solves the CDHP of G with advantage ϵ' within t' where

$$\epsilon' \leq (1 - 1/2^{\ell})\epsilon \text{ and } t' \leq t + c(q_{H_1} + q_{H_2} + q_{RS} + q_{SS})$$

for the security parameter ℓ and some constant c regarding group operations.

Proof: The simulator $\mathcal{S}$ is given (P, aP, bP) and sets $P_1 = aP$ and $P_2 = bP$. $\mathcal{S}$ chooses a random t_j, computes $W_j = t_j P_1$ and give $spec_j$ including the $param$ and W_j to the forger $\mathcal{F}$. The simulator responds to the queries as follows.

• H_1 query: To The H_1 query, $\mathcal{S}$ responds with randomly chosen elements in $\mathbb{Z}_p^*$.

• RootSign query: To the RootSign query on m_i, $\mathcal{S}$ chooses random value y_i and returns (m_i, U_i, V_i, h_i) where $U_i = y_i P - h_i aP$, $V_i = y_i bP$ and $h_i = H_1(m_i, U_i)$.

• H_2 query: To the H_2 query on $(spec_j, j_k)$, $\mathcal{S}$ chooses random s_{jk}, h_{jk} and returns $s_{jk}P - (h_{jk}/t_j)bP$ as $H_2(spec_j, j_k)$.

• SubSign query: To the SubSign query on $(spec_j, j_k, m_{j_k})$, $\mathcal{S}$ first chooses a random value x_{j_k}. Since $\mathcal{F}$ is supposed to make no H_1 query on $(spec_j, j_k, x_{j_k}, m_{j_k})$ previously with overwhelming probability, $\mathcal{S}$ is free to set $H_1(spec_j, j_k, x_{j_k}, m_{j_k}) = h_{j_k}$. $\mathcal{S}$ computes $\sigma_{j_k} = t_j s_{j_k} P_1$ and returns $(spec_j, W_j, \sigma_{CA_j}, j_k, x_{j_k}, m_{j_k}, \sigma_{j_k})$ which is a valid

signature, since

$$\begin{aligned}&\tilde{e}(W_j, H_2(spec_j, j_k)) \cdot \tilde{e}(H_1(spec_j, j_k, x_{j_k}, m_{j_k})P_1, P_2)\\ &= \tilde{e}(t_j P_1, s_{j_k} P - (h_{j_k}/t_j)bP) \cdot \tilde{e}(h_{j_k} aP, bP)\\ &= \tilde{e}(P, t_j s_{j_k} aP) = \tilde{e}(P, \sigma_{j_k}).\end{aligned}$$

Suppose $\mathcal{F}$ produces a subsignature $(spec, W, \sigma_{CA}, \alpha, x_\alpha, m_\alpha, \sigma_\alpha)$ such that $\mathcal{F}$ has not made the SubSign query on $(spec, \alpha, m_\alpha)$. Then $h'_\alpha = H_1(spec, \alpha, x_\alpha, m_\alpha)$ does not equal h_α which is used for $H_2(spec, \alpha)$ with the probability $1 - 1/2^\ell$; in this case, $\mathcal{S}$ can recover abP from the forgery $(h'_\alpha - h_\alpha)^{-1}(\sigma_\alpha - ts_\alpha P_1)$ where t is the value chosen for $W = tP_1$. The success probability is $(1 - 1/2^\ell)\epsilon$. □

Remark 2: The above proof also implies that if the signer uses an index only once, the secret key is not revealed. In fact, the simulator in the proof can respond the SubSign query for given $(spec, i)$ only once, which models the legitimate signer. The same assertion holds for the scheme based on the RSA problem described in the next section.

C. Revealing a Signature

In the above scheme, the secret key is revealed when there are two subsignatures with the same index. A little modification in KeyGen, RootSign and RootVerify gives "revealing a signature" rather than the secret key as follows.

- KeyGen. Generate $param$ similarly to the above scheme. Randomly choose $s \in \mathbb{Z}_p$ and compute $Q = sP$. The secret key is s and $param = (G, G_T, \tilde{e}, P, Q, H_1, H_2)$.
- RootSign. Given the $param$, the secret key s and a message m, outputs a root signature $\sigma = sH_2(m)$ on m, as in [3].
- RootVerify. Given the root signature $\sigma = sH_2(m)$ on m and $param$, check whether $\tilde{e}(P, \sigma) = \tilde{e}(G, H_2(m))$.

Also the revealed signature may play the role of the secret key in ID-based schemes such as [1] and [5] and we consider an ID-based variant in the subsection.

D. An ID-based Variant

We describe an ID-based variant of the metered signature based on the CDHP. We divide KeyGen of the metered signature scheme into two steps, Setup and Extract. A new hash function $H_3 : \{0,1\}^* \to G$ is introduced.

- Setup. Generate $param$ in the same way as III-A except P_1 and P_2. Generate an additional full domain hash function $H_3 : \{0,1\}^* \to G$. There is an entity called the PKG (private key generator) which generates secret keys of users. PKG randomly chooses $s \in \mathbb{Z}_p^*$ and sets $P_{pub} = sP$. s is the master secret and $(G, G_T, \tilde{e}, P, P_{pub}, H_1, H_2, H_3)$ is a system parameter.
- Extract. Given ID, PKG computes $Q_{ID} = H_3(ID)$ and $D_{ID} = sQ_{ID}$. D_{ID} is the secret key corresponding to the ID.
- Cert, RootSign, RootVerify, SubSign, SubVerify, Reveal. As in the metered signature except that P_1 and P_2 are replaced by Q_{ID} and P_{pub}.

The ID-based variant of a metered signature scheme can be easily discourage a malicious signer who generates two subsignatures of the same index from a $spec$. Combining [5, Lemma 1] and the technique of the proof of Theorem 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 CDH assumption.

IV. A CONSTRUCTION: BASED ON THE RSA PROBLEM

In Crypto'89, Okamoto and Ohta [14] introduced the notion of disposable zero knowledge proof for electronic coupon tickets. The payment protocol in the electronic coupon ticket is extended to the metered signature scheme.

The RSA problem is to find a such that $a^e = b \pmod n$ from given (n, e, b) where n is an ℓ-bit RSA modulus for the security parameter ℓ, the exponent e is larger than 1 and $b \in \mathbb{Z}_n^*$ is randomly chosen. The advantage of an algorithm $\mathcal{A}$ on the RSA problem is defined as

$$\mathsf{AdvRSA}_{\mathcal{A}} \stackrel{\text{def}}{=} \Pr\left[\mathcal{A}(n, e, b) = b^{1/e} : b \stackrel{r}{\leftarrow} \mathbb{Z}_n^*\right].$$

Definition 5: The RSA problem is (t, ϵ)-hard if there is no algorithm $\mathcal{A}$ whose advantage $\mathsf{AdvRSA}_{\mathcal{A}}$ is at least ϵ within running time t.

A. A Metered Signature based on the RSA Problem

RootSign and RootVerify are identical to Guillou-Quisquater ID-based signature [8].

- KeyGen. Let ℓ_1 and ℓ_2 be security parameters. Generate an ℓ_1-bit RSA modulus $n = pq$ and an $(\ell_2 + 1)$-bit prime e relatively prime to $\phi(n)$. Compute $1/e \bmod \phi(n)$. Choose a random $a \in \mathbb{Z}_n^*$ and compute $b \equiv a^e \pmod n$. Also specify full domain hash functions $H_1 : \{0,1\} \to \{0,1\}^{\ell_2}$ and $H_2 : \{0,1\}^* \to \mathbb{Z}_n^*$. The secret key is $(a, 1/e \bmod \phi(n))$ and the $param$ includes (n, e, b, H_1, H_2).
- RootSign. Given the public key (n, e, b), the secret key a and a message m, randomly choose $k \in \mathbb{Z}_n^*$ and compute $r \equiv k^e \pmod n$. Compute $s \equiv ka^{H_1(m,r)} \pmod n$. $\sigma = (m, r, s)$ is a root signature on m.
- RootVerify. Given the root signature (m, r, s) and the $param$, check whether $s^e \equiv rb^{H_1(m,r)} \pmod n$.
- Cert. The signer sends a root signature on $spec$, which contains an index set of subsignatures and $param$. The certifier returns a certificate σ_{CA} on them.
- SubSign. Given the secret key a, $spec$, an index $i \in \mathbb{N}$, a message m_i and σ_{CA}, choose a random string $x_i \in \{0,1\}^{\ell_2/2}$ and computes

$$\sigma_i = H_2(spec, i)^{1/e} a^{H_1(spec, i, x_i, m_i)} \pmod n.$$

$(spec, \sigma_{CA}, i, x_i, m_i, \sigma_i)$ is an i-th subsignature from $spec$.

- SubVerify. Given the subsignature $(spec, \sigma_{CA}, i, x_i, m_i, \sigma_i)$, $param$ and the certificate σ_{CA}, the verifier checks the following:
  1. Checks whether i is valid with respect to $spec$.
  2. Checks whether σ_{CA} is valid.
  3. Checks whether $\sigma_i^e = H_2(spec, i) \cdot b^{H_1(spec, i, x_i, m_i)} \pmod n$.
- Reveal. Suppose two valid i-th subsignatures

$$\sigma_i = H_2(spec, i)^{1/e} a^{H_1(spec, i, x_i, m_i)}$$

and

$$\sigma_i' = H_2(spec, i)^{1/e} a^{H_1(spec,i,x_i',m_i')}$$

are given. Let $h = H_1(spec, i, x_i, m_i)$ and $h' = H_1(spec, i, x_i', m_i')$. Since e is (ℓ_2+1)-bit prime and $(h-h')$ is ℓ_2-bit, e and $h-h'$ are mutually prime. Thus there exist integers α, β such that $\alpha(h-h') + \beta e = 1$ and the secret key a can be computed by

$$(\sigma_i/\sigma_i')^\alpha \cdot b^\beta = (a^{h-h'})^\alpha \cdot (a^e)^\beta = a^{\alpha(h-h')+\beta e} = a.$$

B. Security Analysis

The security proof is in the random oracle model. We consider a signature forger defined in section II-B. Since the root signature is the same as Guillou-Quisquater signature, we only analyze the security against the subsignature forger.

Theorem 2: If there exists a $(t, q_{RS}, q_{SS}, q_H, \epsilon)$-subsignature forger $\mathcal{F}$, then we can construct a simulator $\mathcal{S}$ which solves the RSA problem with advantage ϵ' within t' where

$$\epsilon' \leq (1 - 1/2^{\ell_2})\epsilon \text{ and } t' \leq t + c(q_{H_1} + q_{H_2} + q_{RS} + q_{SS})$$

for the security parameter ℓ and some constant c regarding group operations.

Proof: The simulator $\mathcal{S}$ is given (n, e, b) where n is an ℓ_1-bit RSA modulus, $b \in \mathbb{Z}_n^*$ and e is (ℓ_2+1)-bit prime. $\mathcal{S}$ sets (n, e, b) to be the public key and its goal is to use the adversary's forgery to compute $b^{1/e} \pmod n$. The simulator responds to the queries as follows.

- H_1 query: To the H_1 query, $\mathcal{S}$ responds with randomly chosen ℓ_2-bit strings.
- RootSign query: To the RootSigh query on m_i, $\mathcal{S}$ chooses random $s_i \in \mathbb{Z}_n$ and returns (m_i, r_i, s_i, h_i) where $r_i = s_i^e b^{-h_i}$ and $h_i = H_1(m, r_i)$.
- H_2 query: To the H_2 query on $(spec_j, j_k)$, $\mathcal{S}$ chooses a random number $t_{j_k} \in \mathbb{Z}_n$ and a random exponent $c_{j_k} \in \{0,1\}^{\ell_2}$ and sets $r_{j_k} = H_2(spec_j, j_k) = t_{j_k}^e b^{-c_{j_k}} \pmod n$.
- SubSign query: To the Subsign query on $(spec_j, m_{j_k})$, $\mathcal{S}$ first chooses a random value x_{j_k}. Since $\mathcal{F}$ will not have had H_1 query on $(spec_j, j_k, x_{j_k}, m_{j_k})$ previously with overwhelming probability, so $\mathcal{S}$ is free to set $H_1(spec_j, j_k, x_{j_k}, m_{j_k}) = c_{j_k}$. $\mathcal{S}$ can then compute

$$\begin{aligned}\sigma_{j_k} &= (H_2(spec_j, j_k) b^{H_1(spec_j, j_k, x_{j_k}, m_{j_k})})^{1/e} \\ &= (t_{j_k}^e b^{-c_{j_k}} b^{c_{j_k}})^{1/e} = t_{j_k} \pmod n.\end{aligned}$$

Suppose $\mathcal{F}$ produces a subsignature $(spec, \sigma_{CA}, \alpha, x_\alpha, m_\alpha)$ such that $\mathcal{F}$ has not made the SubSign query on $(spec, \sigma_{CA}, \alpha, m_\alpha)$. Then the value of $c_\alpha' = H_1(spec, \alpha, x_\alpha, m_\alpha)$ does not equal c_α which is used for $H_2(spec, \alpha)$ with the probability $1 - 1/2^{\ell_2}$; in this case, $\mathcal{S}$ can recover $b^{1/e}$ from the forgery $(t_\alpha^e b^{c_\alpha' - c_\alpha})^{1/e} \pmod n$. The success probability is $(1 - 1/2^{\ell_2})\epsilon$. □

Remark 3: As the CDHP based metered signature scheme, revealing a signature is possible. If b is replaced by $H_2(M)$ for some message M, $H_2(M)^{1/e}$ which is the RSA signature [16], would be revealed.

Remark 4: While an ID-based variant is possible for the metered signature based on the CDHP, it seems not for the metered signature based on the RSA. Differently from Guillou-Quisquater ID-based signature, the signer should know $1/e \pmod{\phi(n)}$ to generate subsignatures of the metered signature based on the RSA.

V. BATCH VERIFICATION OF METERED SIGNATURES

Batch cryptography was introduced first by Fiat [7] and batch verifications for the DSS and the RSA signature scheme have been studied in [2], [4]. Roughly speaking, the batch verification is to verify a set of signatures at one time. So it can be more efficient than verifying signatures individually. If a set of signatures passes the batch verification algorithm, each signature of them should be valid. On the other hand, a weaker notion of batch verification is "screening" [2]. The difference between the batch verification and the screening is that even though a set of signatures passes the screening, each of them is not necessarily a valid signature of its message. However, the screening guarantees that every message is authenticated by the signer, which is enough for most of applications. In fact, the batch verification in metered signatures is also screening which is enough for our applications.

A. Security Model for the Batch Verification of Metered Signatures

Let $\sigma_I = \{(spec, \sigma_{CA}, i, x_i, m_i, \sigma_i) \mid i \in I\}$ be a set of subsignatures for some index set I. Let BatchVer be a batch verification algorithm which takes $param, \sigma_I$ and outputs Valid or $\perp$. Let $\mathcal{F}_B$ be a forger against the batch verification. The advantage of $\mathcal{F}_B$ is defined as the success probability in the following game.

- **Setup, Queries.** Identical to the game in Section II-B.
- **Response.** Finally, $\mathcal{F}_B$ outputs (I, σ_I) which passes the batch verification and there is at least one index $i \in I$ or one message m_j which has not been made in SubSign query with $spec$.

B. Batch Verification of Metered Signatures based on the CDHP

Suppose there are subsignatures of the metered signature based on the CDHP ,

$$\{(spec, \sigma_{CA}, i, x_i, m_i, \sigma_i) \mid i \in I\}$$

for some index set I. Then the batch verification can be done by checking whether

$$\begin{aligned}&\tilde{e}\left(P, \sum_{i\in I} \sigma_i\right) \\ =&\tilde{e}\left(W, \sum_{i\in I} H_2(spec, i)\right) \cdot \tilde{e}\left(\left(\sum_{i\in I} h_i\right) P_1, P_2\right)\end{aligned}$$

holds where $h_i = H_1(spec, i, x_i, m_i)$. The batch verification requires only three bilinear map computations which are the same as the verification of one signature. We compare the number of cryptographic operations of batch verifications with that of k individual verifications in Table 1. In practice, G is a subgroup of a supersingular elliptic curve over a finite field $\mathbb{F}_q$, G_T is a multiplicative subgroup of an extension field of $\mathbb{F}_q$ and $\tilde{e}$ is a Tate pairing. For more detail, refer to [5]. Ignoring the time for

some light computations such as ordinary hash functions, finite field operations and elliptic curve addition, we need to compute 3 pairings, 1 MapToGroup and 1 scalar multiplication to verify a single metered signature. Thus it takes about 73.91 ms and $73.91k$ ms for the verification of a signature and the individual verification of k signatures, respectively. The batch verification of k signatures takes about $(72.09 + 1.82k)$ ms which is comparable to that of a single signature if k is moderate.

Table 1. Efficiency comparison of the batch verification based on the CDHP

Operations	Individual verification of k signatures	Batch verification of k signatures
	Operations / Time(ms)	Operations / Time(ms)
Bilinear map	$3k$ / $66.54k$	3 / 66.54
Scalar mult. on G	k / $5.55k$	1 / 5.55
MapToGroup hash	k / $1.82k$	k / $1.82k$

To evaluate the performance, optimized MIRACL (using comba method) library v.4.8.3 [12] in Xeon(TM) CPU 2.8GHz with 1 Gbytes memory is used. In MapToGroup and Pairing, a subgroup of order p in a supersingular elliptic curve E over $\mathbb{F}_q$ is used, where q is a 512 bit prime and p is a 160 bit prime.

Theorem 3: The above batch verification is secure.

Proof: Suppose (P, aP, bP) is given. $\mathcal{S}$ puts them in $param$ as P is a generator, $aP = P_1$ and $bP = P_2$. We construct an algorithm $\mathcal{S}$ which outputs abP using the forger $\mathcal{F}_B$. $\mathcal{S}$ executes a simulation as the same way in the proof of Theorem 1. Suppose $\mathcal{S}$ does not abort the simulation and the forger $\mathcal{F}_\mathcal{B}$ outputs $\{(spec, \sigma_{CA}, i, x_i, m_i, \sigma_i) \mid i \in I\}$ which passes the batch verification. Then $\mathcal{S}$ can compute $abP = (\sum h'_i - h_i)^{-1} \sum(\sigma_i - ts_i aP)$. □

C. Batch Verification of Metered Signatures based on the RSA Problem

Suppose there are subsignatures of the metered signature based on the RSA problem,

$$\{(spec, \sigma_{CA}, i, x_i, m_i, \sigma_i) \mid i \in I\}$$

for some index set I. Then the batch verification can be done by checking whether

$$\left(\prod_{i\in I} \sigma_i\right)^e = \left(\prod_{i\in I} H_2(spec, i)\right) \cdot b^{\sum H_1(spec,i,x_i,m_i)}$$

holds.

For individual verifications of k signatures, we need to compute $2k$ modular exponentiations, $2k$ hash functions and k multiplication on $\mathbb{Z}_n^*$. But the batch verification requires only two modular exponentiations, $k-1$ integer addition of H_1 hash values, $2k$ hash computations and $2k-1$ multiplication on $\mathbb{Z}_n^*$. More detailed comparison on modular exponentiations is given in Table 2. Recall that ℓ_1 is the bit-size of the RSA modulus and ℓ_2 is the bit-size of the verifying exponent e. In practice ℓ_1 is 1024 and ℓ_2 is 160. Note that one ℓ_2-bit exponentiation and one $(\ell_2 + \log k)$-bit exponentiation are required in batch verification.

Theorem 4: The above batch verification is secure.

Proof: Suppose (n, e, b) is given. $\mathcal{S}$ puts them the public parameters. We construct an algorithm $\mathcal{S}$ which outputs $b^{1/e}$ using the forger $\mathcal{F}_B$. $\mathcal{S}$ executes a simulation as the same way in the proof of Theorem 1. Recall that to the H_2 query on $(spec, i)$, $\mathcal{S}$ responds with $r_i = H_2(spec, i) = t_i b^{-c_i}$ for randomly chosen c_i and t_i. In H_1 queries, $\mathcal{S}$ chooses a random string and returns. Suppose that $\mathcal{F}_B$ outputs $\{(spec, \sigma_{CA}, i, x_i, m_i, \sigma_i) \mid i \in I\}$. Let $c'_i = H_1(spec, i, x_i, m_i)$ which passes the batch verification and $c = \sum_{i\in I} c'_i - \sum_{i\in I} c_i$. With overwhelming probability, $(c, e) = 1$. $\mathcal{S}$ computes $(\prod_{i\in I} \sigma_i)(\prod_{i\in I} t_i)^{-1} = (b^c)^{1/e}$. From $(b^{1/e})^e$ and $(b^{1/e})^c$ with $(c, e) = 1$, $\mathcal{S}$ computes $b^{1/e}$. □

Table 2. Efficiency comparison of the batch verification based on the RSA problem

Individual verification of k signatures		Batch verification of k signatures	
# of modular exp.	time(ms)	# of modular exp.	time(ms)
$2k$ (ℓ_2-bit)	$4.78k$	1(ℓ_2-bit) + 1($(\log k + \ell_2)$-bit)	≈ 4.78

To evaluate the performance, optimized MIRACL (using comba method) library v.4.8.3 [12] in Xeon(TM) CPU 2.8GHz with 1 Gbytes memory is used.

VI. APPLICATIONS OF METERED SIGNATURES

A. k-times Delegation

Metered signature can be used to restrict the signing capability by k times. In proxy signatures, the original signer, say Alice, can delegate her signing right to a proxy signer, say Bob, only for k times: First Bob generates $U = rP$ and sends it to Alice. Second, Alice makes a certificate on U, k and some warrant information by her private key and sends it to Bob. Bob can generate k metered signatures with his secret key and r, which can be verified by Alice's public key, the Bob's public key, U, and the certificate. In case Bob generates these metered signatures more than k times, the signing key is revealed from two signatures with the same index. This restriction can be more effective if the proxy signer's key is also used for more important purpose at the same time. For example, Alice may enforce Bob to use the secret key of his banking account for proxy signatures in very critical applications. We note that the restriction can be extended to *k-times a day*.

B. Signature Commitment

We can use metered signatures to reveal a signature on some contract, rather than a signing key. If *spec* in the root signature contains an appropriate contract, the contract is automatically signed on any violation of the agreement. For example, suppose that a web site provides free trial downloads by five times a month and demands ten dollars payments for more than five times. A user registers to this site by giving an unsigned check on ten dollars and a root signature and receives a certificate on it. He can issue only five signatures with this certificate in each month to access the trial download. Otherwise, the internet site can obtain the signed check on ten dollars from two signatures with the same index.

C. Integrity of Signature Chains

Another application of metered signatures is to guarantee the order of previously issued signatures without using time stamping authority. It can be used to audit any removal or replacement

of signed messages in a public bulletin board. For example, consider the situation that the owner of a restaurant runs a web site on which customers post their evaluation on it. One problem of this site is that the owner can easily add a good evaluation and remove a bad evaluation. Adding a good evaluation can be partially restricted, for example, by admitting only one posting by one pseudonym which can be issued anonymously from each real identity. Our metered signature provides a solution for preventing removal or replacement of a bad reputation. When a customer wants to give an evaluation on the restaurant, he receives an indexed token (a metered signature) signed by the web site manager and uploads it with his evaluation. In this case, any removal, insertion, or change of order is detected. Further, any replacement of a bad reputation requires issuing two tokens with the same index, which reveals the secret key of the manager and can ruin all of the evaluation values in the web site.

VII. CONCLUSION

We proposed a new notion of metered signatures and two instances are exemplified. It can be used to restrict the signing ability by k-times or more specifically k-times a day. This property is desirable for many applications including proxy signatures, internet polling systems, and electronic coupons. Further, when all the signatures issued by a signer are publicly obtainable as in bulletin board, metered 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.

We have two kinds of penalties against violation in metered signatures: One is to reveal the secret key and the other is to reveal a signature on some contract. Revealing secret key can be used for more strict restrictions. For more mild applications, we may use signature revealing techniques. It would be an interesting problem to add the anonymity such as the unlinkability on metered signatures. Then revealing the identity of the signer rather than a secret key or a signature becomes a countermeasure against the violation.

ACKNOWLEDGEMENT

The authors would like to thank Craig Gentry for fruitful comments and discussions on the design of a metered signature and especially permitting us to include his idea of revealing signatures in this paper. The authors also want to thank Gene Tsudik and Jeong Hyun Yi for helpful discussions. This work was supported by the SRC program of Korea Science and Engineering Foundation (KOSEF) grant funded by the Korea government (MEST) (No. R11-2007-035-01002-0).

REFERENCES

[1] D. Boneh and M. Franklin. Identity-based Encryption from the Weil Pairing. *SIAM J. Comput.*, Vol. 32, No. 3, pp. 586–615, 2003. A preliminary version appeared in *Advesces in Cryptology - Crypto 2001*, LNCS Vol. 2139, pp. 213–229, Springer-Verlag, 2001.

[2] 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.

[3] D. Boneh, B. Lynn, and H. Shacham. Short Signature from the Weil Pairing. *J. of Cryptology*, Vol. 17, No. 4, pp. 297–319, 2004. The extended abstract appeared in *Advances in Cryptology - Asiacrypt 2001*, LNCS Vol. 2248, pp. 514–531, Springer-Verlag, 2001.

[4] 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.

[5] 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.

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

[7] 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.

[8] 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.

[9] 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.

[10] 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.

[11] R. Merkle. A Certified Digital Signature. In *Advences in Cryptology - Crypto'89*, LNCS Vol. 435, pp. 218–238, Springer-Verlag, 1990.

[12] Shamus Software Ltd. Miracl: Multiprecision integer and rational arithmetic c/c++ library. Available at http://indigo.ie/ mscott/.

[13] L. Nguyen and R. Safavi-Naini. Dynamic k-times Anonymous Authentication. *ACNS 2005*,LNCS Vol. 3531, pp. 318-333, Springer-Verlag, 2005.

[14] T. Okamoto and K. Ohta. Disposable Zero-Knowledge Authentication and Their Applications to Untraceable Electronic Cash. In *Advences in Cryptology - Crypto'89*, LNCS Vol. 435, pp.481-496, Springer-Verlag, 1990.

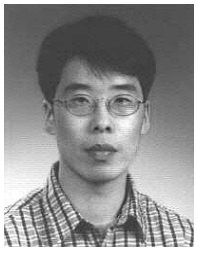
[15] M. Rabin. Digitalized signatures. *Foundations of Secure Computations*, Academic Press, pp. 155–168, 1978.

[16] R. Rivest, A. Shamir, and L. Adleman. A Method for Obtaining Digital Signatures and Public-Key Cryptosystems. In *Commun. ACM 21(2)*, pp. 120–126, 1978.

[17] I. Teranishi, J. Furukawa, and K. Sako. k-times Anonymous Authentication. In *Advances in Cryptology - Asiacrypt 2004*, LNCS Vol. 3329, pp. 308–322, Springer-Verlag, 2004.

[18] I. Teranishi and K. Sako. k-Times Anonymous Authentication With a Constant Proving Cost. *PKC 2006*, LNCS Vol. 3958, pp. 525–542, Springer-Verlag 2006.

[19] H. Yoon, J. Cheon, and Y. Kim. Batch Verifications with ID-based Signatures. In *ICISC 2004*, LNCS Vol. 3506, pp. 233–248, Springer-Verlag, 2004.



Woo-Hwan Kim received the BS, MS and PhD degrees in the department of mathematical sciences from the Seoul National University in 1998, 2000, and 2004, respectively. He joined the attached institute of ETRI in 2004. His research interests include cryptography and information security.

HyoJin Yoon received the BS in the department of mathematical education and MS, PhD degrees in the department of mathematical sciences from the Seoul National University in 1999, 2001, and 2006, respectively. For one year after PhD, she was a post doc at Seoul National University. She joined Samsung SDS IT R&D center in 2007. Her research interests include sensor network, cryptography, information and network security.

Jung Hee Cheon is an associate professor in the department of mathematical sciences of Seoul national university (SNU). He received his B.S. and Ph.D. degrees in mathematics from KAIST in 1991 and 1997, respectively. After working as a senior researcher in ETRI and a visiting scientist in Brown university, he was an assistant professor in Information and Communications University (ICU) for two years. His research interests include computational number theory, cryptography and information security.