ISSN 1342-2812

Research Reports on Mathematical and Computing Sciences

Sanitizable Signature with Secret Information

Manabu Suzuki and Toshiyuki Isshiki and Keisuke Tanaka

December 2005, C–215

Department of
Mathematical and
Computing Sciences
Tokyo Institute of Technology

SERIES C: Computer Science

Sanitizable Signature with Secret Information

Manabu Suzuki Toshiyuki Isshiki Keisuke Tanaka *

Dept. of Mathematical and Computing Sciences
Tokyo Institute of Technology
W8-55, 2-12-1 Ookayama, Meguro-ku, Tokyo 152-8552, Japan
{suzuki1, keisuke}@is.titech.ac.jp

February 20, 2006

Abstract

A sanitizable signature scheme is a signature scheme that allows the sanitizer to sanitize certain portions of the document and to generate the valid signature of the resulting document with no interaction with the signer. There exist many models and schemes for sanitizable signature. In this paper, we precisely formalize the algorithms and the security requirements of sanitizable signature with secret information. We propose a sanitizable signature scheme based on the gap co-Diffie-Hellman groups and prove that our scheme satisfies these security requirements. Furthermore, we discuss various models for sanitizable signature. In particular, we focus on three major properties, such as state controllability for signer, flags for the sanitized messages, and designation of the sanitizer. We also classify the previously proposed schemes and our scheme.

Keywords: sanitizable signature, the gap co-Diffie-Hellman groups, bilinear maps, the random oracle model.

1 Introduction

Digital signature has been an essential tool in current society. The standard digital signature schemes are designed to prevent the alteration of the signed documents. However, in some cases, appropriate alteration of the signed documents should be allowed.

One such situation is the disclosure of the official information. Consider, for example, what happens when a citizen demands the public releases of the official document and its signature. The office then deletes the sensitive data such as personal information or national secrets, and discloses the sanitized version of the document. If this disclosure is done digitally by using the standard digital signature schemes, the citizen cannot verify the disclosed information correctly because the information has been altered to prevent the leakage of sensitive information. The standard digital signature schemes provide the means to achieve the authentication of the document. However, the standard digital signatures do not allow any alteration of the document.

A simple solution is to sign the document every moment when the document is changed, but it is not practical in the case that the document is frequently changed. An alternative solution would be to obtain a valid signature of the sanitized document without any help of the original signer.

There could be many possible reasons for not asking the original signer to re-sign, including; (1) the signer's key has expired, (2) the original signature was securely time-stamped, (3) the signer may not be available, (4) each new signature would cost too much, either in terms of real expense or in terms of computation. Sanitizable signature is introduced in order to address these needs.

*Supported in part by NTT Information Sharing Platform Laboratories and Grant-in-Aid for Scientific Research, Ministry of Education, Culture, Sports, Science, and Technology, 14780190, 16092206.

Sanitizable signatures. A sanitizable signature scheme is a signature scheme that allows the sanitizer to sanitize certain portions of the document and to generate the valid signature of the resulting document with no interaction with the signer. A sanitizable signature is processed by three parties consisting of a signer, a sanitizer, and a verifier. The signer generates the signature assuring the authenticity of the document. The sanitizer receives the document and its signature from the signer. The sanitizer generates the sanitized document and its signature without any help of the signer. The verifier receives the sanitized document of the signature from the sanitizer. The verifier accepts the signature only if he verifies the authenticity of the disclosed document.

Previous works. There are some previously proposed schemes. Miyazaki et al. [9] proposed the schemes called *SUMI-1*, *SUMI-2*, *SUMI-3*, and *SUMI-4*. In the model of the schemes, the sanitizer can sanitize any message he wants. The signer cannot restrict sanitization. Steinfeld, Bull, and Zheng [10] proposed *CES-CV*, *CES-HT*, *CES-RSAP*, and *CES-MERP*. They are provably secure. In the model of the schemes, the signer can assign each message whether it can be sanitized or not. However, the signer cannot change his assignment once the signer generates the signature. Miyazaki et al. [8] proposed *SUMI-5*. In the model of the scheme, the signer can change his assignment even after he generates the signature. Miyazaki et al. [7] also proposed *SUMI-6*. Miyazaki et al. [7] claims that the scheme can hide the number of sanitized messages of the document. Ateniese, Chou, Medeiros, and Tsudik introduced the *sanitizable signatures* [1]. In the model of the scheme, only the designated sanitizer can sanitize the document. Even the signer cannot sanitize the document after he generates the signature. Notice that the meaning of 'sanitize' in this scheme is different from the other protocols. In this scheme, 'sanitize the message' means 'change the message', while in our scheme and the other schemes, 'sanitize the message' means 'hide the message'.

Although there exist many models and schemes for sanitizable signature, most of the schemes are not proved to be secure.

Our contribution. In this paper, we precisely formalize the algorithms and the security requirements of sanitizable signature with secret information. We propose a sanitizable signature scheme based on the gap co-Diffie-Hellman groups and prove that our scheme satisfies these security requirements. In our model, the signer can assign each message whether it can be sanitized or not. The signer can change his assignment even after he generates the signature. Furthermore, we discuss various models for sanitizable signature. In particular, we focus on three major properties, such as state controllability for signer, flags for the sanitized messages, and designation of the sanitizer. We also classify the previously proposed schemes and our schemes.

Organization. The rest of this paper is organized as follows. In Section 2, we provide some mathematical preliminaries. In Section 3, we describe sanitizable signature with secret information. We provide the notation, procedure, and some security definitions. In Section 4, we describe a sanitizable signature scheme based on the gap co-Diffie-Hellman groups and prove that our scheme satisfies the security requirements. In Section 5, we survey previously proposed schemes and classify them. We conclude in Section 6.

2 Preliminaries

We review several concepts related to bilinear maps and gap Diffie-Hellman groups. We also review the co-GDH signature scheme and the aggregate signature.

2.1 Bilinear Maps and Gap Diffie-Hellman Groups

We use the following notation:

1. G_1 and G_2 are two (multiplicative) cyclic groups of prime order p,
2. g_1 is a generator of G_1 and g_2 is a generator of G_2,
3. Let $\psi : G_2 \to G_1$ be an isomorphism that satisfies $\psi(g_2) = g_1$,
4. $e : G_1 \times G_2 \to G_T$ is a computable bilinear map.

With this setup, we obtain natural generalizations of the CDH and DDH problem:

Computational co-Diffie-Hellman problem on (G_1, G_2)**.** Given $g_2, g_2^a \in G_2$ and $h \in G_1$, compute $h^a \in G_1$.

Decision co-Diffie-Hellman problem on (G_1, G_2)**.** Given $g_2, g_2^a \in G_2$ and $h, h^b \in G_1$, output **yes** if $a = b$ and **no** otherwise.

Gap co-Diffie-Hellman groups. We define gap co-Diffie-Hellman groups. We say that a pair (G_1, G_2) is a gap co-Diffie-Hellman group pair if the decision co-Diffie-Hellman problem on (G_1, G_2) is easy but the computational co-Diffie-Hellman problem on (G_1, G_2) is hard.

We define the advantage of an algorithm $\mathcal{A}$ in solving the computational co-Diffie-Hellman problem on (G_1, G_2) as

$$\textbf{Adv co-CDH}_{\mathcal{A}} = \Pr[\mathcal{A}(g_2, g_2^a, h) = h^a; a \leftarrow \mathbb{Z}_p, h \leftarrow G_1]$$

The probability is taken over the choice of a, h and $\mathcal{A}$'s coin tosses. An algorithm $\mathcal{A}$ (t, ϵ)-breaks the computational co-Diffie-Hellman problem on G_1 and G_2 if $\mathcal{A}$ runs in time at most t, and **Adv co-CDH**$_{\mathcal{A}}$ is at least ϵ.

Definition 1 *Two Groups (G_1, G_2) are a (t, ϵ)-gap co-Diffie-Hellman group (co-GDH group) pair if they satisfy the following properties:*

1. *The computation on both G_1 and G_2 and the map ψ from G_2 to G_1 can be computed in one time unit,*
2. *The decision co-Diffie-Hellman problem on (G_1, G_2) can be solved in one time unit,*
3. *No algorithm (t, ϵ)-breaks the computational co-Diffie-Hellman problem on (G_1, G_2).*

Currently, the only examples of gap Diffie-Hellman groups arise from bilinear maps [5].

Bilinear Maps. Let G_1 and G_2 be two groups as above, with an additional group G_T such that $|G_1| = |G_2| = |G_T|$. A bilinear map is a map $e : G_1 \times G_2 \to G_T$ with the following properties:

1. Bilinear: for all $u \in G_1, v \in G_2$ and $a, b \in \mathbb{Z}$, $e(u^a, v^b) = e(u, v)^{ab}$,
2. Non-degenerate: $e(g_1, g_2) \neq 1$.

These properties imply that for any $u_1, u_2 \in G_1$ and $v \in G_2$, $e(u_1 u_2, v) = e(u_1, v) \cdot e(u_2, v)$, and that for any $u, v \in G_2, e(\psi(u), v) = e(\psi(v), u)$.

2.2 The co-GDH Signature Scheme

We review the co-GDH signature scheme [4] proposed by Boneh, Lynn, and Shacham. This scheme works in any gap co-Diffie-Hellman group pair (G_1, G_2). It uses a full-domain hash function $h : \{0,1\}^* \to G_1$, viewed as a random oracle [3].

Key Generation. Pick random $x \leftarrow \mathbb{Z}_p$, and compute $v \leftarrow g_2^x$. The public key is $v \in G_2$. The secret key is $x \in \mathbb{Z}_p$.

Signing. Given a secret key x and a message $M \in \{0,1\}^*$, compute $w \leftarrow h(M)$, where $w \in G_1$, and $\sigma \leftarrow w^x$. The signature is $\sigma \in G_1$.

Verification. Given a public key v, a message M, and a signature σ, compute $w \leftarrow h(M)$ and verify that $e(w, v) = e(\sigma, g_2)$ holds.

This scheme satisfies the existential unforgeability under the chosen message attack [6] in the random oracle model assuming that (G_1, G_2) is a (t, ϵ)-gap co-Diffie-Hellman group pair, where ϵ is negligible.

2.3 Bilinear Aggregate Signatures

We review a bilinear aggregate signature scheme based on the co-GDH signature scheme described above. Individual signatures in the aggregate signature scheme are created and verified precisely as are signatures in the co-GDH signature scheme. Aggregate verification makes use of a bilinear map on G_1 and G_2.

The aggregate signature scheme allows the creation of signatures on arbitrary distinct messages $M_i \in \{0,1\}^k$, where k is the security parameter. An individual signature σ_i is an element of G_1. The base groups G_1 and G_2, their respective generators g_1 and g_2, the computable isomorphism ψ from G_2 to G_1, and the bilinear map $e : G_1 \times G_2 \to G_T$, with target group G_T, are system parameters.

The scheme employs a full-domain hash function $h : \{0,1\}^* \to G_1$, viewed as a random oracle.

Key Generation. For a particular user, pick random $x \stackrel{R}{\leftarrow} \mathbb{Z}_p$, and compute $v \leftarrow g_2^x$. The user's public key is $v \in G_2$. The user's secret key is $x \in \mathbb{Z}_p$.

Signing. For a particular user, given the secret key x and a message $M \in \{0,1\}^k$, compute $w \leftarrow h(M)$, where $w \in G_1$, and $\sigma \leftarrow w^x$. The signature is $\sigma \in G_1$.

Verification. Given user's public key v, a message M, and a signature σ, compute $w \leftarrow h(M)$; accept if $e(\sigma, g_2) = e(w, v)$ holds.

Aggregation. For the set U of users, assign to each user an index i, ranging from 1 to $n = |U|$. Each user $u_i \in U$ provides a signature $\sigma_i \in G_1$ on message $M_i \in \{0,1\}^k$ of his choice. The messages M_i must all be distinct. Compute $\sigma \leftarrow \prod_{i=1}^{n} \sigma_i$. The aggregate signature is $\sigma \in G_1$.

Aggregate Verification. We are given an aggregate signature $\sigma \in G_1$ for U, indexed as before, and are given the original messages $M_i \in \{0,1\}^k$ and public keys $v_i \in G_2$ for all users $u_i \in U$. To verify the aggregate signature σ,

1. ensure that the messages M_i are all distinct, and reject otherwise; and
2. compute $w_i \leftarrow h(M_i)$ for $1 \le i \le n = |U|$, and accept if $e(\sigma, g_2) = \prod_{i=1}^{n} e(w_i, v_i)$ holds.

A bilinear aggregate signature, like a co-GDH signature, is a single element of G_1. Note that aggregation can be done incrementally.

The intuition behind bilinear aggregate signatures is as follows. Each user u_i has a secret key $x_i \in \mathbb{Z}_p$ and a public key $v_i = g_2^{x_i}$. User u_i's signature, if correctly formed, is $\sigma_i = w_i^{x_i}$. Using the properties of the bilinear map, the left-hand side of the verification equation expands:

$$e(\sigma, g_2) = e(\prod_i w_i^{x_i}, g_2) = \prod_i e(w_i, g_2)^{x_i} = \prod_i e(w_i, g_2^{x_i}) = \prod_i e(w_i, v_i),$$

which is the right-hand side, as required.

3 Sanitizable Signature with secret information

In this section, we precisely formalize the algorithms and the security requirements of sanitizable signature with secret information.

A sanitizable signature scheme is a signature scheme that allows the sanitizer to sanitize certain portions of the document and to generate the valid signature of the resulting document with no interaction with the signer. Sanitizable signature is processed by three parties consisting of a signer, a sanitizer, and a verifier. The signer generates the signature assuring the authenticity of the document. The sanitizer receives the document and the signature from the signer. The sanitizer generates the sanitized document and its signature without any help of the signer. The verifier receives the sanitized document of the signature from the sanitizer. The verifier accepts the signature only if he verifies the authenticity of the disclosed document.

3.1 Notation

In this section, we describe the notation for document and message, state, and secret information.

Document and message. We denote that a document $\boldsymbol{M}$ is the list of messages M_i (e.g. $\boldsymbol{M} = (M_1 \parallel \cdots \parallel M_n)$). We denote that $\parallel$ is the concatenation. We use $length(\boldsymbol{M})$ to denote the number of the messages in $\boldsymbol{M}$. For example, if the document is $\boldsymbol{M} = (M_1 \parallel \cdots \parallel M_n)$, then $length(\boldsymbol{M}) = n$. We use ϕ to indicate that the message is sanitized. For example, if the i-th message of the document $\boldsymbol{M}$ is sanitized, then the document $\boldsymbol{M}'$ is represented as $\boldsymbol{M}' = (M_1 \parallel \cdots \parallel M_{i-1} \parallel \phi \parallel M_{i+1} \parallel \cdots \parallel M_n)$. For $\boldsymbol{M}^1$ and $\boldsymbol{M}^2$ such that $length(\boldsymbol{M}^1) = length(\boldsymbol{M}^2)$, we say that $\boldsymbol{M}^1$ is a subdocument of $\boldsymbol{M}^2$ if $M_i^1 = M_i^2$ for all i where M_i^1 are not sanitized. For example, if $\boldsymbol{M}^1 = (a \parallel b \parallel \phi \parallel d \parallel \phi)$ and $\boldsymbol{M}^2 = (a \parallel b \parallel \phi \parallel d \parallel e)$, then $\boldsymbol{M}^1$ is a subdocument of $\boldsymbol{M}^2$.

State. Each message has one of the following three states:

1. Sanitized,
2. Disclosed and sanitizing is allowed,
3. Disclosed and sanitizing is prohibited.

We define the state $st_{\boldsymbol{M}}$ of the document $\boldsymbol{M}$.

Definition 2 (State) *Let $\boldsymbol{M}$ be a document. $st_{\boldsymbol{M}}$ describes the states of M_i. Each M_i is either 'sanitized', 'disclosed and sanitizing is allowed', or 'disclosed and sanitizing is prohibited'. $st_{\boldsymbol{M}}$ is constructed as $st_{\boldsymbol{M}} = (st^S_{\boldsymbol{M}}, st^A_{\boldsymbol{M}}, st^P_{\boldsymbol{M}})$, where $st^S_{\boldsymbol{M}}, st^A_{\boldsymbol{M}}, st^P_{\boldsymbol{M}}$ are sets of indices. $st^S_{\boldsymbol{M}}$ is a set of indices of the messages that are 'sanitized'. $st^A_{\boldsymbol{M}}$ is a set of indices of the messages that are 'disclosed and sanitizing is allowed'. $st^P_{\boldsymbol{M}}$ is set of indices of the messages that are 'disclosed and sanitizing is prohibited'. Note that every index i must belong in only one of $st^S_{\boldsymbol{M}}, st^A_{\boldsymbol{M}}$, and $st^P_{\boldsymbol{M}}$.*

Secret information. In our model, the signer can control the states of the messages whether 'disclosed and sanitizing is allowed' or 'disclosed and sanitizing is prohibited'. The signer can control the states by using the secret information. The signer generates the secret information for each message of the document. The secret information is necessary to generate the signature of the sanitized document. The signer sends the secret information of the message to the sanitizer if he allows the sanitizer to sanitize the message. Otherwise he does not send the secret information of the message.

Definition 3 (Secret Information) *Let $\boldsymbol{M}$ be a document. SI is a set of the secret information of the messages, that is generated by the signer. It is described as $SI = \{A_i | i \in st^A_{\boldsymbol{M}}\}$, where A_i is the secret information to sanitize the message M_i.*

3.2 Procedure

Sanitizable signature schemes with secret information consists of the set of four algorithms.

Key Generation. A probabilistic algorithm **KeyGen**, on input 1^k, outputs public and secret keys (PK, SK), where k is the security parameter:

$$(PK, SK) \leftarrow \mathbf{KeyGen}(1^k).$$

Signing. A probabilistic algorithm **Sign** takes as input a document $\boldsymbol{M}$, a secret key SK, and a state $st_{\boldsymbol{M}}$ of the document. The signing algorithm outputs a document $\boldsymbol{M}$, a signature σ of $\boldsymbol{M}$, a state $st_{\boldsymbol{M}}$ of $\boldsymbol{M}$, and a set of secret information $SI = \{A_i | i \in st^A_{\boldsymbol{M}}\}$. A_i is necessary for the sanitizer to sanitize the message M_i. In this model, the signer can control the state of each message of the document by the secret information A_i. The signer can assign each message of the document whether it is allowed to sanitize or prohibited to sanitize:

$$(\boldsymbol{M}, st_{\boldsymbol{M}}, \sigma, SI) \leftarrow \mathbf{Sign}(\boldsymbol{M}, st_{\boldsymbol{M}}; SK).$$

Sanitizing. A deterministic algorithm **Sanitize** takes as input a document $\boldsymbol{M}$, a state $st_{\boldsymbol{M}}$ of $\boldsymbol{M}$, a signature σ on $\boldsymbol{M}$, secret information $SI = \{A_i | i \in st^A_{\boldsymbol{M}}\}$, and a state of the sanitized document $st_{\boldsymbol{M}'}$. It outputs a sanitized document $\boldsymbol{M}'$, a state $st_{\boldsymbol{M}'}$ of $\boldsymbol{M}'$, a signature σ' of $\boldsymbol{M}'$. It outputs $\perp$ if $i \in st^P_{\boldsymbol{M}}$ satisfies $i \in st^S_{\boldsymbol{M}'}$:

$$((\boldsymbol{M}', st_{\boldsymbol{M}'}, \sigma') \text{ or } \perp) \leftarrow \mathbf{Sanitize}(\boldsymbol{M}, st_{\boldsymbol{M}}, \sigma, SI, st_{\boldsymbol{M}'}; PK).$$

Verification. A deterministic algorithm **Verify** that, on input a sanitized document $\boldsymbol{M}'$, a state $st_{\boldsymbol{M}'}$ of $\boldsymbol{M}'$, a possibly valid signature σ', and a public key PK, outputs '*accept*' or '*reject*':

$$\{\textit{accept, reject}\} \leftarrow \mathbf{Verify}(\boldsymbol{M}', st_{\boldsymbol{M}'}, \sigma'; PK).$$

We require that sanitizable signature schemes satisfy the following correctness condition: for any $st_{\boldsymbol{M}'}$ such that $i \in st^S_{\boldsymbol{M}'}$ is in $st^A_{\boldsymbol{M}}$,

$$\begin{aligned}\Pr[\mathbf{Verify}(\boldsymbol{M}', st_{\boldsymbol{M}'}, \sigma'; PK) = \textit{accept} \quad | \\ (PK, SK) \leftarrow \mathbf{KeyGen}(1^k), \\ (\boldsymbol{M}, st_{\boldsymbol{M}}, \sigma, SI) \leftarrow \mathbf{Sign}(\boldsymbol{M}, st_{\boldsymbol{M}}; SK), \\ (\boldsymbol{M}', st_{\boldsymbol{M}'}, \sigma') \leftarrow \mathbf{Sanitize}(\boldsymbol{M}, st_{\boldsymbol{M}}, \sigma, SI, st_{\boldsymbol{M}'}; PK)] = 1.\end{aligned}$$

3.3 Security Requirements

Sanitizable signature schemes with secret information should satisfy the following criteria, that is, unforgeability and indistinguishability.

Unforgeability. It should be difficult to generate the valid signature on the document without the knowledge of the secret signing key, and also to sanitize specific messages of the document without the secret information.

Game.

Unforgeability is defined using the following game between the challenger $\mathcal{B}$ and the adversary $\mathcal{A}$. $\mathcal{A}$ can query to the signing oracle. Let q_s be the number of the queries to the signing oracle.

Setup. The challenger $\mathcal{B}$ runs the algorithm **KeyGen** to obtain a public key PK and a secret key SK. The adversary $\mathcal{A}$ is given PK.

Queries. Adaptively, $\mathcal{A}$ requests signatures with PK on at most q_s documents $\boldsymbol{M}^1, \ldots, \boldsymbol{M}^{q_s}$. The attacker $\mathcal{A}$ queries $(\boldsymbol{M}^j, st_{\boldsymbol{M}^j}, \mathit{flag})$ to the signing oracle, where $\mathit{flag} \in \{0,1\}$. The attacker $\mathcal{A}$ chooses $\boldsymbol{M}^j = {M_1}^j \parallel {M_2}^j \parallel \cdots \parallel {M_n}^j$ (notice that each message ${M_i}^j$ can be ${M_i}^j = \phi$), and the state $st_{\boldsymbol{M}^j}$ for $\boldsymbol{M}^j$. $\mathcal{A}$ queries $(\boldsymbol{M}^j, st_{\boldsymbol{M}^j}, \mathit{flag})$ to the signing oracle. Remind that $st_{\boldsymbol{M}^j} = (st^S_{\boldsymbol{M}^j}, st^A_{\boldsymbol{M}^j}, st^P_{\boldsymbol{M}^j})$. The signing oracle returns answers according to the $\mathit{flag} \in \{0,1\}$ as follows.

$\mathit{flag} = 0$.

The signing oracle responds to each query with the queried document and its signature $(\boldsymbol{M}^j, \sigma^j)$ and the secret information $SI^j = \{A^j_i\}$ where $i \in st^A_{\boldsymbol{M}^j}$. For $i \in st^S_{\boldsymbol{M}^j}$, the signing oracle picks a random $R^j_i \in \{0,1\}^k$ and sets $M^{j'} = M_1^{j'} || \cdots || M_n^{j'}$, where if $i \in st^S_{\boldsymbol{M}^j}$ then $M_i^{j'} = R_i^j$, else $M_i^{j'} = M_i^j$. The signing oracle generates a signature on $\boldsymbol{M}^{j'}$. Then, the signing oracle sanitizes the document $\boldsymbol{M}^{j'}$ (i.e. for any $i \in st^S_{\boldsymbol{M}^j}$, replaces $M_i^{j'}$ with ${M_i}^j = \phi$) and generates its signature σ^j. Note that the sanitized document is equal to $\boldsymbol{M}^j$. The signing oracle sets $SI^j = \{A^j_i\}$, where $i \in st^A_{\boldsymbol{M}^j}$. The oracle returns $(\boldsymbol{M}^j, \sigma^j, SI^j)$ to $\mathcal{A}$.

$\mathit{flag} = 1$.

The signing oracle responds to each query with the sanitized document and the signature pair $(\boldsymbol{M}^{j''}, \sigma^{j''})$ and the secret information $SI^{j''} = \phi$. For $i \in st^S_{\boldsymbol{M}^j}$, the signing oracle picks a random $R^j_i \in \{0,1\}^k$ and sets $\boldsymbol{M}^{j'} = M_1^{j'} || \cdots || M_n^{j'}$, where if $i \in st^S_{\boldsymbol{M}^j}$ then $M_i^{j'} = R_i^j$, else $M_i^{j'} = M_i^j$. The signing oracle generates a signature on $\boldsymbol{M}^{j'}$. Then, the signing oracle sanitizes the document $\boldsymbol{M}^{j'}$ (i.e. for any $i \in st^S_{\boldsymbol{M}^j} \cup st^A_{\boldsymbol{M}^j}$, replaces $M_i^{j'}$ with ${M_i}^j = \phi$) and generates its signature. The oracle returns $(\boldsymbol{M}^{j''}, \sigma^{j''}, SI^{j''})$ to $\mathcal{A}$, where $\boldsymbol{M}^{j''}$ is sanitized document, $\sigma^{j''}$ is its signature, and $SI^{j''} = \phi$.

Output. Eventually, $\mathcal{A}$ outputs a triple $(\boldsymbol{M}^B, st_{\boldsymbol{M}^B}, \sigma^B)$ and wins the game if

$\mathbf{Verify}(\boldsymbol{M}^B, st_{\boldsymbol{M}^B}, \sigma^B; PK) = \mathit{accept}$ and either 1, 2, or 3 holds.

1. $\boldsymbol{M}^B$ is not a subdocument of any $\boldsymbol{M}^j (i = 1, \ldots, q_s)$.
2. (a) and (b) are satisfied.
   (a) $\boldsymbol{M}^B$ is a subdocument of some $\boldsymbol{M}^j (i = 1, \ldots, q_s)$.
   (b) Some messages $M^B_i (i \in st^P_{\boldsymbol{M}^j})$ are sanitized.

3. (a) and (b) are satisfied.
   (a) $\boldsymbol{M}^B$ is a subdocument of some $\boldsymbol{M}^j (i = 1, \ldots, q_s)$.
   (b) There exists some i such that $i \in st^S_{\boldsymbol{M}^j} \wedge i \notin st^S_{\boldsymbol{M}^B}$.

The statement 1 corresponds to the standard forging of signatures. The statement 2 implies that the attacker sanitizes a message which is prohibited to sanitize. The statement 3 implies that the attacker discloses a message which is sanitized.

We now define $AdvSig_{\mathcal{A}}$ to be the probability that $\mathcal{A}$ wins the game.

Definition 4 *A forger $\mathcal{A}$ (t, q_s, q_h, ϵ)-wins the game if $\mathcal{A}$ runs in time at most t, $\mathcal{A}$ makes at most q_s signature queries and at most q_h queries to the hash function, and $AdvSig_{\mathcal{A}}$ is at least ϵ. A sanitizable signature scheme is (t, q_s, q_h, ϵ)-unforgeable if no forger (t, q_s, q_h, ϵ)-wins it.*

Indistinguishability. It should be difficult to obtain any information on the sanitized messages. Informally, it should be infeasible for the attacker, having access to the signing oracle, to distinguish between two signatures σ^0 and σ^1 of his choice, where σ^0 (σ^1) is the signature on the sanitized document $\boldsymbol{M}^{0'}$ ($\boldsymbol{M}^{1'}$) of the original document $\boldsymbol{M}^0$ ($\boldsymbol{M}^1$).

For example, let $\boldsymbol{M}^0 = (l \parallel i \parallel k \parallel e)$ and $\boldsymbol{M}^1 = (l \parallel o \parallel v \parallel e)$. Let $\boldsymbol{M}^{0'} = (l \parallel \phi \parallel \phi \parallel e)$ and $\boldsymbol{M}^{1'} = (l \parallel \phi \parallel \phi \parallel e)$. Let σ_0 (σ_1) is a signature of the sanitized document $\boldsymbol{M}^{0'}$ ($\boldsymbol{M}^{1'}$). The attacker cannot distinguish the signatures σ_0 and σ_1 of two sanitized documents.

Game.

Formally, indistinguishability is defined using the following game between the challenger $\mathcal{B}$ and the adversary $\mathcal{A} = (\mathcal{A}_{find}, \mathcal{A}_{guess})$. $\mathcal{A}$ is constructed by two algorithms $\mathcal{A}_{find}$ and $\mathcal{A}_{guess}$. Let q_s be the number of the queries to the signing oracle.

Setup. The challenger $\mathcal{B}$ runs the algorithm **KeyGen** to obtain a public key PK and a secret key SK. The adversary $\mathcal{A}$ is given PK.

Queries. Adaptively, $\mathcal{A} = (\mathcal{A}_{find}, \mathcal{A}_{guess})$ requests signatures with PK on at most q_s documents $\boldsymbol{M}^1, \ldots, \boldsymbol{M}^{q_s}$. Each Query is done in the same way as in the game for the unforgeability.

$\mathcal{A}_{find}$. $\mathcal{A}_{find}$ outputs two original documents and the corresponding states $(\boldsymbol{M}^0, st_{\boldsymbol{M}^{0'}})$ and $(\boldsymbol{M}^1, st_{\boldsymbol{M}^{1'}})$. They must satisfy the following properties.

1. $st_{\boldsymbol{M}^{0'}} = st_{\boldsymbol{M}^{1'}}$.
2. $M^0_i = M^1_i$ for all $i \notin st^S_{\boldsymbol{M}^{0'}}$ $(i \notin st^S_{\boldsymbol{M}^{1'}})$.
3. $M^0_i \neq M^1_i$ for some $i \in st^S_{\boldsymbol{M}^{0'}}$ $(i \in st^S_{\boldsymbol{M}^{1'}})$.

$\mathcal{A}_{find}$ sends $(\boldsymbol{M}^0, st_{\boldsymbol{M}^{0'}})$ and $(\boldsymbol{M}^1, st_{\boldsymbol{M}^{1'}})$ to the challenger $\mathcal{B}$.

Responds of the challenger $\mathcal{B}$. The challenger $\mathcal{B}$ receives $(\boldsymbol{M}^0, st_{\boldsymbol{M}^{0'}})$ and $(\boldsymbol{M}^1, st_{\boldsymbol{M}^{1'}})$ from $\mathcal{A}_{find}$.

The challenger $\mathcal{B}$ first checks that these documents and states satisfy above statements. Then the challenger $\mathcal{B}$ randomly chooses $b \leftarrow \{0, 1\}$ and sends $(\boldsymbol{M}^{b'}, st_{\boldsymbol{M}^{b'}}, \sigma^{b'})$ to $\mathcal{A}_{guess}$, where $\sigma^{b'}$ is the signature of the sanitized document.

$\mathcal{A}_{guess}$. $\mathcal{A}_{guess}$ outputs b' and wins if $b = b'$.

We now define $AdvInd_{\mathcal{A}}$ as $AdvInd_{\mathcal{A}} = |\Pr[b = b'] - 1/2|$. We say that the sanitizable signature scheme satisfies indistinguishability if there is no adversary that wins the above game with non-negligible probability.

Definition 5 *A forger $\mathcal{A}$ (t, q_s, q_h, ϵ)-wins the game if $\mathcal{A}$ runs in time at most t, $\mathcal{A}$ makes at most q_s signature queries and at most q_h queries to the hash function, and $AdvInd_{\mathcal{A}}$ is at least ϵ. A sanitizable signature scheme is (t, q_s, q_h, ϵ)-indistinguishable if no forger (t, q_s, q_h, ϵ)-wins it.*

Note that the above definition on the indistinguishability is computational. As defined in [1], the indistinguishability can be strengthened to be information theoretic. In fact, as done in the proof on the indistinguishability of our scheme is information theoretic.

4 Our Scheme

4.1 Description

We describe our scheme in this section. Our scheme is processed by three parties consisting of a signer, a sanitizer, and a verifier.

The key generation algorithm **KeyGen**, on input 1^k, to obtain (x, g_2, g_2^x) and output (PK, SK), where $PK = (g_2, g_2^x, h : \{0,1\}^* \to G_1, e : G_1 \times G_2 \to G_T)$ and $SK = (x)$.

We now describe how to sign the document. The inputs to the algorithm **Sign** are a document $\boldsymbol{M}$, a state $st_{\boldsymbol{M}}$ of $\boldsymbol{M}$, and a secret key SK. The outputs are the document $\boldsymbol{M}$, the state $st_{\boldsymbol{M}}$ of $\boldsymbol{M}$, the signature σ of $\boldsymbol{M}$, and the secret information $SI = \{A_i | i \in st^A_{\boldsymbol{M}}\}$.

Sign$(\boldsymbol{M}, \boldsymbol{st_M}, SK)$

1. Choose random strings $r_i \in \{0,1\}^k$ for $i = 1, \ldots, n+1$. Note that k is the security parameter.
2. Compute $w_i = h(M_i \parallel r_i)$ for $i = 1, \ldots, n$.
3. Compute $A_i = (w_i)^x$ for $i = 1, \ldots, n$.
4. Compute $A = \prod_i^n A_i$.
5. Compute $w_{n+1} = h(w_1 \parallel \cdots \parallel w_n \parallel r_{n+1})$.
6. Compute $S = (w_{n+1})^x$.
7. Compute $D = A \cdot S$.
8. Return $(\boldsymbol{M}, st_{\boldsymbol{M}}, \sigma, SI)$, where $\sigma = D \parallel r_1 \parallel \cdots \parallel r_{n+1}$ and $SI = \{A_i | i \in st^A_{\boldsymbol{M}}\}$.

A_i is necessary to sanitize message M_i. If the signer wants to prohibit the sanitization of M_i, then he does not send A_i to the sanitizer. The signer sends $(\boldsymbol{M}, st_{\boldsymbol{M}}, \sigma, SI = \{A_i | i \in st^A_{\boldsymbol{M}}\})$ to the sanitizer.

The sanitizer receives $(\boldsymbol{M}, st_{\boldsymbol{M}}, \sigma, SI = \{A_i | i \in st^A_{\boldsymbol{M}}\})$ from the signer. We now describe how to sanitize the document $\boldsymbol{M} = M_1 \parallel M_2, \parallel \cdots \parallel M_n$. The input is $(\boldsymbol{M}, st_{\boldsymbol{M}}, \sigma, SI, st_{\boldsymbol{M}'})$. It outputs a sanitized document $\boldsymbol{M}'$, a state $st_{\boldsymbol{M}'}$ of the document, a signature σ' of $\boldsymbol{M}'$.

Sanitize $(\boldsymbol{M}, \boldsymbol{st_M}, \sigma, SI, \boldsymbol{st_{M'}})$

1. Check that each index $i \in st^S_{\boldsymbol{M}'}$ is in $st^A_{\boldsymbol{M}}$. If not, then report failure and terminate. Otherwise continue.
2. Check that A_i satisfies $e(w_i, g_2^x) = e(A_i, g_2)$ for all $i \in st^A_{\boldsymbol{M}}$. If not, then report failure and terminate. Otherwise continue.
3. Compute $w_i = h(M_i \parallel r_i)$ for all $i \in st^A_{\boldsymbol{M}}$ and $i \in st^P_{\boldsymbol{M}}$.

4. Compute $D^{'} = D / \prod_{i \in st^S_{M'}} A_i$.

5. Return $(\boldsymbol{M}^{'}, st_{\boldsymbol{M}'}, \sigma^{'})$, where $\sigma^{'} = D^{'} \parallel r_i (i \notin st^S_{\boldsymbol{M}'}) \parallel w_i (i \in st^S_{\boldsymbol{M}'})$.

The sanitizer sends $(\boldsymbol{M}^{'}, st_{\boldsymbol{M}'}, \sigma^{'})$ to the verifier, where $\boldsymbol{M}^{'}$ is the sanitized document of $\boldsymbol{M}$ by replacing each M_i $(i \in st^S_{\boldsymbol{M}'})$ with ϕ.

Verify$(\boldsymbol{M}^{'}, \boldsymbol{st}_{\boldsymbol{M}'}, \sigma^{'}, PK)$

1. Check that $M_i^{'} = \phi$ for all $i \in st^S_{\boldsymbol{M}'}$ and $M_i^{'} \neq \phi$ for all $i \notin st^S_{\boldsymbol{M}'}$. If not, then report failure and terminate. Otherwise continue.

2. Compute each $w_i = h(M_i \parallel r_i)$ for all $i \notin st^S_{\boldsymbol{M}'}$.

3. Compute $w_{n+1} = h(w_1 \parallel \cdots \parallel w_n \parallel r_{n+1})$

4. Compute $w = (\prod_{i \notin st^S_{M'}} w_i) \cdot w_{n+1}$.

5. Return '*accept*' if $e(D^{'}, g_2) = e(w, g_2^x)$ and return '*reject*' otherwise.

4.2 Security

Correctness. It is clear that for any for any $st_{\boldsymbol{M}'}$ such that $i \in st^S_{\boldsymbol{M}'}$ is in $st^A_{\boldsymbol{M}}$, a signature of the sanitized document generated correctly by the algorithms **KeyGen**, **Sign**, and **Sanitize** is accepted by the algorithm **Verify**.

4.2.1 Unforgeability

The following theorem implies that our scheme is unforgeable. The proof of the theorem is in Appendix A.

Theorem 1 *Let (G_1, G_2) be a $(t^{'}, \epsilon^{'})$-co-GDH group pair of order p. Then our scheme on (G_1, G_2) is (t, q_s, q_h, ϵ)-unforgeable in the random oracle model for all t and ϵ satisfying*

$$\epsilon \geq e(q_s + 1) \cdot \epsilon^{'},$$

$$t \leq t^{'} - c_{G_1}(q_h + (n+2) \cdot q_s),$$

where c_{G_1} is constant that depends on G_1 and e is the base of the natural logarithm.

Hence, security of the signature scheme follows from the hardness of the co-CDH problem on (G_1, G_2).

4.2.2 Indistinguishability

The following theorem implies that our scheme satisfies indistinguishability. The proof of the theorem is in Appendix B.

Theorem 2 *If h is a random oracle, $\mathcal{A}$ can distinguish the signature with the sanitized document of $(\boldsymbol{M}^0, st_{\boldsymbol{M}^0})$ and that of $(\boldsymbol{M}^1, st_{\boldsymbol{M}^1})$ with probability at most $1/2^{k-1}$.*

5 Discussion on the Models for Sanitizable Signature

There exist many models and schemes for sanitizable signature. In this section, we first discuss various models of sanitizable signature. We then classify the previously proposed schemes and our scheme.

5.1 Various Models

In this section, we discuss various models for sanitizable signature. We here focus on three major properties such as state controllability for the signer, flags for the sanitized messages, and designation of the sanitizer.

State controllability for the signer. We can consider three types of the state controllability for the signer.

1. The signer cannot control each state of the message. It means that the sanitizer can sanitize any message he wants and the signer cannot restrict sanitization.
2. The signer can assign one of the states 'disclosed and sanitizing is allowed' or 'disclosed and sanitizing is prohibited' to the message. However, one cannot change each state of the message without the signer's secret key after the signer generates the signature.
3. The signer can assign one of the conditions 'disclosed and sanitizing is allowed' or 'disclosed and sanitizing is prohibited'. One can control each state of the message without the signer's secret key even after the signer generates the signature.

We cannot say that which one is superior to another. It depends on the application.

Flags for the sanitized messages. We can consider three types of the flags for the sanitized messages.

1. The positions of the sanitized message are flagged. In this case, anyone can notice which messages are sanitized by the sanitizer.
2. The positions of the sanitized messages are not flagged. In this case, anyone except for the signer and the sanitizer cannot notice the sanitization of the messages.
3. The sanitizer can select whether the positions of the sanitized message are flagged or not.

We cannot say that which one is superior to another. It depends on the application.

Designation of the sanitizer. We can consider three types of the designation of the sanitizer.

1. Anyone who receives the document and its signature can generate the signature of the sanitized document.
2. The signer has to designate the specific sanitizer. Only the designated sanitizer can sanitize the document. Even the signer cannot sanitize the document after he generates the signature.
3. The signer can select whether he designates the specific sanitizer or not.

We cannot say that which one is superior to another. It depends on the application.

5.2 Previous Works

In this section, we review the previously proposed schemes. We also classify these schemes and ours in terms of the categories discussed above.

Digitally signed document sanitizing schemes (*SUMI-1*, *SUMI-2*, *SUMI-3*, *SUMI-4*). Miyazaki et al. [9] proposed simple schemes *SUMI-1*, *SUMI-2*, *SUMI-3*, and *SUMI-4*. For example, we describe *SUMI-4* as follows. The signer generates random numbers for all messages of the document. Then the signer calculates hash values with corresponding random numbers for all messages of the document and generates the signature for the concatenation of the hash values. The sanitizer can sanitize any messages he wants. If he wants to sanitize the specific messages, he just calculates the hash values and substitutes the hash values for the messages. The model of the scheme is the type 1 of the state controllability for the signer. The signer cannot prohibit sanitization. This model is the type 1 of the flags for the sanitized messages. The positions of the sanitized messages are flagged. If the message is replaced with its hash value, one can notice that the message is sanitized. This model is the type 1 of the designation of the sanitizer. Anyone who receives the documents, corresponding signatures, and the secret information can generate the sanitized signatures.

Content extraction signatures (*CES-CV*, *CES-HT*, *CES-RSAP*, *CES-MERP*). Steinfeld, Bull, and Zheng proposed [10] *CES-CV*, *CES-HT*, *CES-RSAP*, and *CES-MERP*). They are provably secure. *CES-CV* uses the standard digital signature scheme and the message commitment scheme. The security is based on unforgeability of the standard digital signature, hiding and binding of the commitment scheme. *CES-RSAP* is the modification of an RSA batch screening verifier, proposed by Bellare et al [2]. *CES-HT* is the variant of *CES-CV*, and *CES-MERP* is the variant of *CES-RSAP*.

In the model of the scheme, the signer controls the states of the messages by Content Extraction Access Structure (CEAS). The signer uses CEAS to specify which messages the signer allows sanitization. CEAS is an encoding of positions of the messages that are allowed to sanitize.

This model is the type 2 of the state controllability for the signer. The signer can assign to the message, one of the conditions 'disclosed and sanitizing is allowed' or 'disclosed and sanitizing is prohibited' by CEAS. However, the signer cannot control each state of the message of once the signer generates the signature of the document. The signer signs to the concatenation of the document and CEAS. This model is the type 1 of the flags for the sanitized messages. The positions of the sanitized messages are flagged. One can notice which messages are sanitized by the sanitizer. This model is the type 1 of the designation of the sanitizer. Anyone who receives the documents, corresponding signatures can generate the sanitized signatures.

Digitally signed document sanitizing scheme with disclosure condition control (*SUMI-5*). Miyazaki et al. [8] proposed *SUMI-5*. This scheme uses the standard digital signature scheme and the message commitment scheme. It is provably secure. The security is based on unforgeability of the standard digital signature, hiding and binding of the commitment scheme.

The model of the scheme is the type 3 of the state controllability for the signer. The signer can assign one of the conditions 'disclosed and sanitizing is allowed' or 'disclosed and sanitizing is prohibited'. In addition to it, the signer can control each state of the message even after he generates the signature. The signer can control the states by some auxiliary information. This model is the type 1 of the flags for the sanitized messages. The positions of the sanitized messages are flagged. One can notice which messages are sanitized by the sanitizer. This model is the type 1 of the designation of the sanitizer. Anyone who receives the documents, corresponding signatures, and some auxiliary information can generate the sanitized signatures.

Digitally signed document sanitizing scheme from bilinear maps (*SUMI-6*). Miyazaki et al. [7] also proposed *SUMI-6*. This scheme is based on the aggregate signature from bilinear maps.

scheme	state controllability	flag	designation
SUMI-4 [9]	1	1	1
CES-CV [10]	2	1	1
SUMI-5 [8]	3	1	1
SUMI-6 [7]	3	2	1
Sanitizable signatures [1]	2	-	2
Our scheme	3	1	1

Figure 1: The types of models

The model of the scheme is the type 3 of the state controllability for the signer. In this model, the signer can control the state of the document even after he generates the signature. This model is the type 2 of the flags for the sanitized messages. Miyazaki et al. claims that the scheme can hide the number of sanitized messages of the document. It means that the positions of the sanitized messages are not flagged. . This model is the type 1 of the designation of the sanitizer. Anyone who receives the document, the signature, and the secret information can sanitize.

However, there are some defects in this scheme. At first, it has to put the message ID and the address of each message. It is undesirable to put the message ID and the address. Second, they did not provide the security proof. The definition of the security is not clear and not enough. This scheme has the attack that anyone can sanitize the message which is prohibited by the signer to sanitize.

Sanitizable signatures. Ateniese, Chou, Medeiros, and Tsudik proposed schemes called *sanitizable signatures* [1]. This model of the scheme allows the sanitizer (authorized semi-trusted censors) to modify parts of the signed message without interacting with the signer. Notice that the meaning of 'sanitize' in this scheme is different from other protocols. In this scheme, 'sanitize the message' means 'change the message', while in our scheme and the other schemes, 'sanitize the message' means 'hide the message'.

We describe the scheme. There exist two pairs of keys. The signer and the sanitizer (semi-trusted censors who can modify parts of a signed message) have their own public keys and secret keys. The signer signs the document with his own secret key and the sanitizer's public key. The sanitizer (semi-trusted censors who receives the signed message) uses his secret key to generate the new document of the signature. The verifier verifies the signature with the signer's public key and sanitizer's public key.

The model of the scheme is the type 2 of the state controllability for the signer. The signer can assigns to the message, one of the conditions 'disclosed and sanitizing is allowed' or 'disclosed and sanitizing is prohibited'. However, once the signer generates the signature of the document, the signer cannot control each state of the message of the document. In other words, the signer cannot change each state of the message once the signer generates the signature. In this model, only the designated sanitizer can sanitize the document. Even the signer cannot sanitize the document after he generates the signature of it. However, the meaning of 'sanitize' in this scheme is different from other protocols.

5.3 Our Scheme

In this section, we classify our scheme and the previously proposed schemes in terms of the categories discussed above. We discuss our model and then list the models with the types in Fig 1.

The model of our scheme is the type 3 of the state controllability for the signer. The signer can assign one of the conditions 'disclosed and sanitizing is allowed' or 'disclosed and sanitizing is prohibited'. In addition to it, the signer can control each state of the message even after he generates the signature. The signer can control the states by the secret information. He sends the

secret information of the message to the sanitizer if he allows the sanitizer to sanitize the message. Otherwise he does not sends the secret information of the message. This model is the type 1 of the flags for the sanitized messages. The positions of the sanitized message are flagged. One can notice which messages are sanitized by the sanitizer. In our scheme, ϕ indicates that the message is sanitized. This model is the type 1 of the designation of the sanitizer. Anyone who receives the documents, corresponding signatures, and the secret information can generate the sanitized signatures.

The model of our scheme is same as *SUMI-5*. In *SUMI-5*, they use a standard digital signature scheme and a commitment scheme. The security of their scheme is based on the unforgeability of the standard digital signature and the perfect concealing and computational binding properties of the commitment scheme, while our scheme satisfies the security requirement by assuming the gap co-Diffie-Hellman groups in the random oracle model.

6 Concluding Remarks

In this paper, we have precisely formalized the algorithms and the security requirements of sanitizable signature with secret information. We have proposed a sanitizable signature scheme based on the gap co-Diffie-Hellman groups and proved that our scheme satisfies these security requirements.

We have also discussed various models of sanitizable signature. In particular, we have focused on three major properties, such as state controllability for the signer, flags for the sanitized messages, and designation of the sanitizer. We have also classified the previously proposed schemes and our scheme.

The proof for the indistinguishability of our scheme heavily depends on the random oracle model. It is interesting to consider the schemes without the random oracles and their security proofs.

References

[1] Ateniese, G., Chou, D., de Medeiros, B., and Tsudik, G. Sanitizable signatures. In *ESORICS 2005* (Milan, Italy, 2005), vol. 3679 of *Lecture Notes in Computer Science*, Springer-Verlag, pp. 159–177.

[2] Bellare, M., Garay, J., and Rabin, T. Fast batch verification for modular exonentiation and digital signatures. In *Advances in Cryptology – EUROCRYPT '98* (Espoo, Finland, May 1998), K. Nyberg, Ed., vol. 1403 of *Lecture Notes in Computer Science*, Springer-Verlag, pp. 236–250.

[3] Bellare, M., and Rogaway, P. The exact security of digital signatures - how to sign with rsa and rabin. In *Advances in Cryptology – EUROCRYPT '96* (Saragossa, Spain, May 1996), U. Maurer, Ed., vol. 1070 of *Lecture Notes in Computer Science*, Springer-Verlag, pp. 399–416.

[4] Boneh, D., Lynn, B., and Shacham, H. Short signatures from the weil pairing. In *Advances in Cryptology – ASIACRYPT 2001* (Gold Coast, Australia, December 2001), C. Boyd, Ed., vol. 2248 of *Lecture Notes in Computer Science*, Springer-Verlag, pp. 514–532.

[5] Joux, A., and Nguyen, K. Separating decision diffie-hellman from diffie-hellman in cryptographic groups. Cryptology ePrint Archive, Report 2001/003, `http://eprint.iacr.org/`, 2001.

[6] Micali, S., and Rivest, R. A digital signature scheme secure against adaptive chosen-message attacks. *SIAM J. Computing 17*, 2 (1988), 281–308.

[7] Miyazaki, K., Hanaoka, G., and Imai, H. Digitally signed document sanitizing scheme from bilinear maps. In *The 2005 Symposium on Cryptography and Information Security (SCIS2005)* (Maiko Kobe, Japan, 2005), pp. 1471–1476.

[8] Miyazaki, K., Iwamura, M., Matsumoto, T., Sasaki, R., Yoshiura, H., and Imai, H. Digitally signed document sanitizing scheme with disclosure condition control. *IEICE Trans. Fundamentals E88-A*, 1 (2005), 239–247.

[9] Miyazaki, K., Susaki, S., Iwamura, M., Matsumoto, T., Sasaki, R., and Yoshiura, H. Digital documents sanitizing problem. Tech. Rep. ISEC2003-20, IEICE, 2003.

[10] Steinfeld, R., Bull, L., and Zheng, Y. Content extraction signatures. In *Information Security and Cryptology–ICISC'01* (Seoul, South Korea, 2002), vol. 2288 of *Lecture Notes in Computer Science*, Springer-Verlag, pp. 285–304.

A Unforgeability

The following theorem shows that our scheme is secure.

Theorem 1 *Let (G_1, G_2) be a (t', ϵ')-co-GDH group pair of order p. Then our scheme on (G_1, G_2) is (t, q_s, q_h, ϵ)-unforgeable in the random oracle model for all t and ϵ satisfying*

$$\epsilon \geq e(q_s + 1) \cdot \epsilon',$$

$$t \leq t' - c_{G_1}(q_h + (n+2) \cdot q_s),$$

where c_{G_1} is constant that depends on G_1 and e is the base of the natural logarithm.

Hence, security of the signature scheme follows from the hardness of the co-CDH problem on (G_1, G_2).

Proof of Theorem 1. Suppose $\mathcal{A}$ is a forger algorithm that (t, q_s, q_h, ϵ)-breaks our scheme. We show how to construct a t'-time algorithm $\mathcal{B}$ that solves the co-CDH problem in (G_1, G_2) with probability at least ϵ'. This will contradict the fact that (G_1, G_2) are (t', ϵ')-co-GDH group pair.

Let g_2 be a generator of G_2. Algorithm $\mathcal{B}$ is given $g_2, g_2^a \in G_2$ and $t \in G_1$. Its goal is to output $t^a \in G_1$. Algorithm $\mathcal{B}$ simulates the challenger and interacts with forger $\mathcal{A}$ as follows.

Setup. Algorithm $\mathcal{B}$ starts by giving $\mathcal{A}$ the generator g_2 and the public key $g_2^a \cdot g_2^r$, where r is random in $\mathbb{Z}_p$.

h-queries. At any time algorithm $\mathcal{A}$ can query to the random oracle h. To respond to these queries algorithm $\mathcal{B}$ maintains a list (h-list) of tuples $(M_i, r_i, w_i, b_i, c_i)$ as explained below. The list is initially empty. When $\mathcal{A}$ queries $M_i \parallel r_i$ ($M_i \in \{0,1\}^k$ and $r_i \in \{0,1\}^k$) to the oracle h, algorithm $\mathcal{B}$ responds as follows.

1. If the query $M_i \parallel r_i$ already appears on the h-list in a tuple $(M_i, r_i, w_i, b_i, c_i)$, then algorithm $\mathcal{B}$ responds with $h(M_i \parallel r_i) = w_i \in G_1$.
2. Otherwise, $\mathcal{B}$ generates a random coin $c_i \in \{0,1\}$ such that $Pr[c_i = 0] = 1/(q_s + 1)$.
3. Algorithm $\mathcal{B}$ picks a random $b_i \in \mathbb{Z}_p$. If $c_i = 0$, $\mathcal{B}$ computes $w_i \leftarrow t \cdot \psi(g_2)^{b_i} \in G_1$. If $c_i = 1$, $\mathcal{B}$ computes $w_i \leftarrow \psi(g_2)^{b_i} \in G_1$.
4. Algorithm $\mathcal{B}$ adds the tuple $(M_i, r_i, w_i, b_i, c_i)$ to the h-list and responds w_i.

Note that w_i is uniform in G_1 and is independent of $\mathcal{A}$'s current view.

Signature queries. Let $(\boldsymbol{M}^j, st_{\boldsymbol{M}^j}, flag)$ be a signature query issued by $\mathcal{A}$, where $\boldsymbol{M}^j = M_1{}^j \parallel M_2{}^j, \parallel \cdots \parallel M_n{}^j$. Algorithm $\mathcal{B}$ responds to this query as follows:

$flag = 0$.

1. Algorithm $\mathcal{B}$ obtains $M_i^j (i = 1, \ldots, n)$ from $\boldsymbol{M}^j$. If $M_i^j = \phi$, then $\mathcal{B}$ picks a random $M_i^j \in \{0,1\}^k$.
2. Algorithm $\mathcal{B}$ picks a random $r_i^j \in \{0,1\}^k$ and runs above algorithm for responding to h-queries to obtain a $w_i^j \in G_1$ as $h(M_i^j \parallel r_i^j)$ for all $i = 1, \ldots, n$. Let $(M_i^j, r_i^j, w_i^j, b_i^j, c_i^j)$ be the corresponding tuple on the h-list. If $c_i^j = 0$ for some $i = 1, \ldots, n$, $\mathcal{B}$ repeats running above algorithm until it satisfies $c_i^j = 1$. It follows that $c_i^j = 1$ for all $i = 1, \ldots, n$.
3. Let $M_{n+1}^j = w_1^j \parallel \cdots \parallel w_n^j$. Algorithm $\mathcal{B}$ picks a random $r_{n+1}^j \in \{0,1\}^k$ and runs above algorithm for responding to h-queries to obtain a $w_{n+1}^j \in G_1$ as $h(M_{n+1}^j \parallel r_{n+1}^j)$. Let $(M_{n+1}^j, r_{n+1}^j, w_{n+1}^j, b_{n+1}^j, c_{n+1}^j)$ be the corresponding tuple on the h-list. If $c_{n+1}^j = 0$, then $\mathcal{B}$ reports failure and terminates. Otherwise continues.
4. Algorithm $\mathcal{B}$ computes $A_i^j = (w_i^j)^{a+r} = \psi(g_2^a)^{b_i^j} \cdot \psi(g_2)^{r b_i^j}$. (Note that $w_i^j = \psi(g_2)^{b_i^j}$ since $c_i^j = 1$ for all $i = 1, \ldots, n$.)
5. Algorithm $\mathcal{B}$ computes $S^j = (w_{n+1}^j)^{a+r} = \psi(g_2^a)^{b_{n+1}^j} \cdot \psi(g_2)^{r b_{n+1}^j}$.
6. Algorithm $\mathcal{B}$ computes $D^j = S^j \cdot (\prod_{i=1}^{n} A_i^j)$ and $\sigma^j = D^j \parallel r_i^j \parallel \cdots \parallel r_{n+1}^j$. Note that Algorithm $\mathcal{B}$ sanitizes M_i^j for $i \in st_{\boldsymbol{M}^j}^S$.
7. Algorithm $\mathcal{B}$ returns $\sigma^j = D^j \parallel r_i^j \parallel \cdots \parallel r_{n+1}^j$ and A_i^j $(i \in st_{\boldsymbol{M}^j}^A)$.

$flag = 1$.

1. Algorithm $\mathcal{B}$ obtains $M_i^j (i = 1, \ldots, n)$ from $\boldsymbol{M}^j$. If $M_i^j = \phi$, then $\mathcal{B}$ picks a random $M_i^j \in \{0,1\}^k$.
2. Algorithm $\mathcal{B}$ picks a random $r_i^j \in \{0,1\}^k$ and runs above algorithm for responding to h-queries to obtain a $w_i^j \in G_1$ such that $h(M_i^j \parallel r_i^j)$ for all $i = 1, \ldots, n$. Let $(M_i^j, r_i^j, w_i^j, b_i^j, c_i^j)$ be the corresponding tuple on the h-list. If $c_i^j = 0$ for some $i = 1, \ldots, n$, $\mathcal{B}$ repeats running above algorithm until it satisfies $c_i^j = 1$. It follows that $c_i^j = 1$ for all $i = 1, \ldots, n$.
3. Let $M_{n+1}^j = w_1^j \parallel \cdots \parallel w_n^j$. Algorithm $\mathcal{B}$ picks a random $r_{n+1}^j \in \{0,1\}^k$ and runs above algorithm for responding to h-queries to obtain a $w_{n+1}^j \in G_1$ such that $h(M_{n+1}^j \parallel r_{n+1}^j)$. Let $(M_{n+1}^j, r_{n+1}^j, w_{n+1}^j, b_{n+1}^j, c_{n+1}^j)$ be the corresponding tuple on the h-list. If $c_{n+1}^j = 0$, then $\mathcal{B}$ reports failure and terminates. Otherwise continues.
4. Algorithm $\mathcal{B}$ computes $A_i^j = (w_i^j)^{a+r} = \psi(g_2^a)^{b_i^j} \cdot \psi(g_2)^{r b_i^j}$. (Note that $w_i^j = \psi(g_2)^{b_i^j}$ since $c_i^j = 1$ for all $i = 1, \ldots, n$.)
5. Algorithm $\mathcal{B}$ computes $S^j = (w_{n+1}^j)^{a+r} = \psi(g_2^a)^{b_{n+1}^j} \cdot \psi(g_2)^{r b_{n+1}^j}$.
6. Algorithm $\mathcal{B}$ computes $D^j = S^j \cdot (\prod_{i=1}^{n} A_i^j)$ and $\sigma^j = D^j \parallel r_i^j \parallel \cdots \parallel r_{n+1}^j$. Note that Algorithm $\mathcal{B}$ sanitizes M_i^j for $i \in st_{\boldsymbol{M}^j}^S$.
7. Algorithm $\mathcal{B}$ computes $D^{j'} = D^j / \prod_{i \in st_{\boldsymbol{M}^j}^A} A_i^j$.
8. Algorithm $\mathcal{B}$ returns $\sigma^{j'} = D^{j'} \parallel r_i^j \parallel \cdots \parallel r_{n+1}^j$. Note that algorithm $\mathcal{B}$ does not return any A_i^j $(i \in st_{\boldsymbol{M}^j}^A)$.

Outputs. Eventually algorithm $\mathcal{A}$ generates a document-signature pair $(\boldsymbol{M}^B, \sigma^B)$. By the definition of the unforgeability, $(\boldsymbol{M}^B, \sigma^B)$ satisfies one of these cases.

1. $\boldsymbol{M}^B$ is not a subdocument of any $\boldsymbol{M}^j (i = 1, \ldots, q_s)$.
2. (a) and (b) are satisfied.
   (a) $\boldsymbol{M}^B$ is a subdocument of some $\boldsymbol{M}^j (i = 1, \ldots, q_s)$.
   (b) Some messages $M_i^B (i \in st_{\boldsymbol{M}^j}^P)$ are sanitized.
3. (a) and (b) are satisfied.
   (a) $\boldsymbol{M}^B$ is a subdocument of some $\boldsymbol{M}^j (i = 1, \ldots, q_s)$.
   (b) There exists some i such that $i \in st_{\boldsymbol{M}^j}^S \wedge i \notin st_{\boldsymbol{M}^B}^S$.

We show that algorithm $\mathcal{B}$ can output $t^a \in G_1$ in Case 1, Case 2, and Case 3.

Case 1

If a document-signature pair $(\boldsymbol{M}^B, \sigma^B)$ satisfies Case 1, then algorithm $\mathcal{A}$ generates a document-signature pair $(\boldsymbol{M}^B, \sigma^B)$ such that no signature query was issued for $\boldsymbol{M}^B$.

1. Algorithm $\mathcal{B}$ obtains $M_i^B (i = 1, \ldots, n)$ from $\boldsymbol{M}^B$.
2. Algorithm $\mathcal{B}$ does nothing to the messages $M_i^B (i \in st_{\boldsymbol{M}}^S)$.
3. Algorithm $\mathcal{B}$ runs h-queries algorithm to obtain $(M_i^B, r_i^B, w_i^B, b_i^B, c_i^B)$ if there is no tuple on the h-list.
4. Algorithm $\mathcal{B}$ obtains all $(M_i^B, r_i^B, w_i^B, b_i^B, c_i^B)$ for $i = 1, \ldots, n+1$. Notice that $M_{n+1}^B = w_1^B \parallel \cdots \parallel w_n^B$.
5. If $c_i^B = 0$ for only one i and $c_i^B = 1$ for n other i, then $\mathcal{B}$ computes $D^B / (t^r \cdot \prod_{i=1}^{n+1} (\psi(g_2^a) \cdot \psi(g_2)^{b_i r}))$.

This completes the description of algorithm $\mathcal{B}$ if $\mathcal{A}$'s output satisfies Case 1. It remains to show that $\mathcal{B}$ solves the given instance of the co-CDH problem in (G_1, G_2) with probability at least ϵ'. To do so, we analyze the three events needed for $\mathcal{B}$ to succeed:

ε_1: Algorithm $\mathcal{B}$ does not abort as a result of any of $\mathcal{A}$'s signature queries.

ε_2: $\mathcal{A}$ generates a valid document-signature forgery $(\boldsymbol{M}^B, \sigma^B)$.

ε_3: Event ε_2 and $c_{n+1}^B = 0$.

Algorithm $\mathcal{B}$ succeeds if all of these events happen. The probability $\Pr[\varepsilon_1 \wedge \varepsilon_3]$ decomposes as

$$\Pr[\varepsilon_1 \wedge \varepsilon_3] = \Pr[\varepsilon_1] \cdot \Pr[\varepsilon_2 | \varepsilon_1] \cdot \Pr[\varepsilon_3 | \varepsilon_1 \wedge \varepsilon_2]$$

The following claims give a lower bound for each of these terms.

Lemma 1 *The probability that algorithm $\mathcal{B}$ does not abort as a result of $\mathcal{A}$'s signature queries is at least $1/e$. Hence, $\Pr[\varepsilon_1] \geq 1/e$.*

Proof. Without loss of generality, we assume that $\mathcal{A}$ does not ask for the signature of the same document twice. We prove by induction that after $\mathcal{A}$ makes i signature queries, the probability that $\mathcal{B}$ does not abort is at least $(1 - 1/(q_s + 1))^i$. The lemma is trivially true for $i = 0$. Let $\boldsymbol{M}_i$ be $\mathcal{A}$'s signature query and let $(M_i, r_i, w_i, b_i, c_i)$ be the corresponding tuple on the h-list. The only value that could be given to $\mathcal{A}$ that depends on c_i is $H(M_i \parallel r_i)$, but the distribution on $H(M_i \parallel r_i)$ is the same whether $c_i = 0$ or $c_i = 1$. Therefore, the probability that this query causes $\mathcal{B}$ to abort is at most $1/(q_s + 1)$. That is because $\mathcal{B}$ generates a random coin $c_i \in \{0, 1\}$ only when the query is $M_{n+1}^j = w_1^j \parallel \cdots \parallel w_n^j \parallel r_{n+1}$. Using the inductive hypothesis and the independence of c_i, the probability that $\mathcal{B}$ does not abort after this query is at least $(1 - 1/(q_s + 1))^i$. This proves the inductive lemma. Since $\mathcal{A}$ makes at most q_s signature queries the probability that $\mathcal{B}$ does not abort as a result of all signature queries is at least $(1 - 1/(q_s + 1))^{q_s} \geq 1/e$.

Lemma 2 *If algorithm $\mathcal{B}$ does not abort as a result of $\mathcal{A}$'s signature queries, then algorithm $\mathcal{A}$'s view is identical to its view in the real attack. Hence,* $\Pr[\varepsilon_2|\varepsilon_1] \geq \epsilon$.

Proof. The public key given to $\mathcal{A}$ is from the same distribution as a public key generated by algorithm **KeyGen**. Responses to h-queries are as in the real attack since each response is uniformly and independently distributed in G_1. All responses to signature queries are valid. Therefore, $\mathcal{A}$ will generate a valid document-signature pair with probability at least ϵ. Hence, $\Pr[\varepsilon_2|\varepsilon_1] \geq \epsilon$.

Lemma 3 *The probability that algorithm $\mathcal{B}$ does not abort after $\mathcal{A}$ outputs a valid forgery that satisfies Case 1 is at least* $1/(q_s + 1)$. *Hence,* $\Pr[\varepsilon_3|\varepsilon_1 \wedge \varepsilon_2] \geq 1/q_s + 1$.

Proof. Given that event ε_1 and ε_2 happened, algorithm $\mathcal{B}$ will abort only if $\mathcal{A}$ generates a forgery $(\boldsymbol{M}^B, \sigma^B)$ for which the tuple $(M_{n+1}^B, r_{n+1}^B, w_{n+1}^B, b_{n+1}^B, c_{n+1}^B)$ on the h-list has $c_{n+1}^B = 1$. The distribution on $h(M_{n+1}^B \parallel r_{n+1}^B)$ is the same whether $c_{n+1}^B = 0$ or $c_{n+1}^B = 1$. Since $\mathcal{A}$ could not have issued a signature query for $(M_{n+1}^B \parallel r_{n+1}^B)$. We know that c is independent of $\mathcal{A}$'s view. Therefore, $\Pr[c = 0|\varepsilon_1 \wedge \varepsilon_2] \geq 1/q_s + 1$ as required.

Using the bounds from the lemma above and equation $\Pr[\varepsilon_1 \wedge \varepsilon_3] = \Pr[\varepsilon_1] \cdot \Pr[\varepsilon_2|\varepsilon_1] \cdot \Pr[\varepsilon_3|\varepsilon_1 \wedge \varepsilon_2]$, we can say that $\mathcal{B}$ generates the correct answer with probability at least $\epsilon/e(q_s + 1) \geq \epsilon^{'}$. Algorithm $\mathcal{B}$'s running time is the same as $\mathcal{A}$'s running time plus the time to respond to $(q_h + (n + 1) \cdot q_s)$ hash queries and q_s signature queries. Each query requires an exponentiation in G_1. We assume that it takes time c_{G_1}. Hence, the total running time is at most $t + c_{G_1}(q_h + (n + 2) \cdot q_s) \leq t^{'}$.

Case 2

If a document-signature pair $(\boldsymbol{M}^B, \sigma^B)$ satisfies Case 2, then $\mathcal{A}$ outputs some A_i that is not asked to the signing oracle before.

In this case, we can mention that algorithm $\mathcal{B}$ can output correct answer by the following proposition.

Proposition 1 *Let (G_1, G_2) be a $(t^{'}, \epsilon^{'})$-co-GDH group pair of order p. Then the co-GDH signature scheme on (G_1, G_2) is (t, q_s, q_h, ϵ)-secure against existential forgery under an adaptive chosen-message attack in the random oracle model for all t and ϵ satisfying*

$$\epsilon \geq e(q_s + 1) \cdot \epsilon^{'},$$

$$t \leq t^{'} - c_{G_1}(q_h + 2q_s),$$

where c_{G_1} is constant that depends on G_1 and e is the base of the natural logarithm.

We described the co-GDH signature scheme in section 2. The proof of this proposition shows directly how algorithm $\mathcal{B}$ outputs $t^a \in G_1$. This proof is done by Boneh, Lynn, Shacham in [4].

Case 3

In this case, it satisfies the following three statement.

1. $\boldsymbol{M}^B$ is a subdocument of some $\boldsymbol{M}^j (i = 1, \ldots, q_s)$.
2. There exists some i such that $i \in st^S_{\boldsymbol{M}^j} \wedge i \notin st^S_{\boldsymbol{M}^B}$.

If a document-signature pair $(\boldsymbol{M}^B, \sigma^B)$ satisfies Case 3, $\mathcal{B}$ can output $t^a \in G_1$ as follows.

1. Algorithm $\mathcal{B}$ obtains $M^B_i (i = 1, \ldots, n)$ from $\boldsymbol{M}^B$.
2. Algorithm $\mathcal{B}$ does nothing to the messages $M^B_i (i \in st^S_{\boldsymbol{M}})$.
3. Algorithm $\mathcal{B}$ runs h-queries algorithm to obtain $(M^B_i, r^B_i, w^B_i, b^B_i, c^B_i)$ if there is no tuple on the h-list.
4. Algorithm $\mathcal{B}$ obtains all $(M^B_i, r^B_i, w^B_i, b^B_i, c^B_i)$ for $i = 1, \ldots, n+1$. Notice that $M^B_{n+1} = w^B_1 \parallel \cdots \parallel w^B_n$.
5. Algorithm $\mathcal{B}$ checks the signature list and obtain all the signatures of $\boldsymbol{M}^j$ that are the subdocument of $\boldsymbol{M}^B$.
6. If $c^B_i = 0$ for only one i and $c^B_i = 1$ for n other i, then $\mathcal{B}$ computes D^j / D^B.

Lemma 4 *The probability that algorithm $\mathcal{B}$ does not abort after $\mathcal{A}$ outputs a valid forgery that satisfies Case 3 is at least $1/(q_s + 1)$.*

Proof. The probability that algorithm $\mathcal{B}$ outputs a valid forgery is equal to the probability that $c^B_i = 0$ for only one i and $c^B_i = 1$ for n other i.

Summary of Case 1, Case 2, and Case 3

We showed that given $g_2, g_2^a \in G_2$ and $t \in G_1$, algorithm $\mathcal{B}$ can output $t^a \in G_1$ in Case 1, Case 2, and Case 3.

If the output of $\mathcal{A}$ satisfies Case 1, $\mathcal{B}$ generates the correct answer with probability at least $\epsilon / e(q_s + 1) \geq \epsilon^{'}$. The running time is estimated as $t + c_{G_1}(q_h + (n+2) \cdot q_s) \leq t^{'}$.

If the output of $\mathcal{A}$ satisfies Case 2, $\mathcal{B}$ generates the correct answer with probability at least $\epsilon / e(q_s + 1) \geq \epsilon^{'}$. The running time is $t \leq t^{'} - c_{G_1}(q_h + 2q_s)$.

If the output of $\mathcal{A}$ satisfies Case 3, $\mathcal{B}$ generates the correct answer with probability at least $\epsilon / e(q_s + 1) \geq \epsilon^{'}$. The running time is estimated as $t + c_{G_1}(q_h + (n+2) \cdot q_s) \leq t^{'}$.

Namely, given $g_2, g_2^a \in G_2$ and $t \in G_1$, algorithm $\mathcal{B}$ can output $t^a \in G_1$ with probability at least $\epsilon / e(q_s + 1) \geq \epsilon^{'}$. The running time is estimated as $t + c_{G_1}(q_h + (n+2) \cdot q_s) \leq t^{'}$. It does not depends on the probability of the event Case 1, Case 2, and Case 3. We proved the theorem.

B Indistinguishability

The following theorem shows that our scheme satisfies indistinguishability.

Theorem 2 *If h is a random oracle, $\mathcal{A}$ can distinguish the signature with the sanitized document of $(\boldsymbol{M}^0, st_{\boldsymbol{M}^0})$ and that of $(\boldsymbol{M}^1, st_{\boldsymbol{M}^1})$ with probability at most $1/2^{k-1}$.*

Proof of Theorem 2. Suppose $\mathcal{A}_{find}$ outputs $(\boldsymbol{M}^0, st_{\boldsymbol{M}^{0\prime}})$ and $(\boldsymbol{M}^1, st_{\boldsymbol{M}^{1\prime}})$.

We now compare two signatures with the sanitized document of $(\boldsymbol{M}^0, st_{\boldsymbol{M}^0})$ and that of $(\boldsymbol{M}^1, st_{\boldsymbol{M}^1})$. We show that $\mathcal{A}_{guess}$ cannot distinguish them.

Let $\sigma^0 = D'^0 \parallel r_i^0(i \notin st^S_{\boldsymbol{M}'0}) \parallel h(M_i^0 \parallel r_i^0)(i \in st^S_{\boldsymbol{M}'0})$ and $\sigma^1 = D'^1 \parallel r_i^1(i \notin st^S_{\boldsymbol{M}'1}) \parallel h(M_i^1 \parallel r_i^1)(i \in st^S_{\boldsymbol{M}'1})$. Note that $D'^b = D^b / \prod_{i \in st^S_{\boldsymbol{M}'b}} A_i^b$, and $A_i^b = h(M_i^b \parallel r_i^b)$.

If $\mathcal{A}_{guess}$ distinguishes some of their elements, $\mathcal{A}_{guess}$ can distinguish σ^b from the challenger $\mathcal{B}$.

By the definition of the **Sign**, each r_i^b is chosen randomly so that $\mathcal{A}_{guess}$ can distinguish them with the probability $1/2^k$. By the definition of the random oracle, $\mathcal{A}_{guess}$ can distinguish $h(M_i^b \parallel r_i^b)$ with probability at most $1/2^k$. We can also say that $\mathcal{A}_{guess}$ can distinguish each A_i with probability at most $1/2^k$. We can also say that $\mathcal{A}_{guess}$ can distinguish D' with probability at most $1/2^k$.

Therefore, $\mathcal{A}$ can distinguish the signature with the sanitized document of $(\boldsymbol{M}^0, st_{\boldsymbol{M}^0})$ and that of $(\boldsymbol{M}^1, st_{\boldsymbol{M}^1}$ with probability at most $1/2^{k-1}$.

Remarks. The above proof heavily depends on the random oracle model. It is interesting to consider the schemes without the random oracles and their security proofs.