IEICE TRANS. FUNDAMENTALS, VOL.E94–A, NO.1 JANUARY 2011

PAPER

Sequential Bitwise Sanitizable Signature Schemes

Goichiro HANAOKA[†], **Shoichi HIROSE**[††], **Atsuko MIYAJI**[†††], **Kunihiko MIYAZAKI**[††††], **Bagus SANTOSO**[†a)], ***Members***, ***and*** **Peng YANG**[†††††], ***Nonmember***

SUMMARY A sanitizable signature scheme is a signature scheme which, after the signer generates a valid signature of a message, allows a specific entity (sanitizer) to modify the message for hiding several parts. Existing sanitizable signature schemes require the message to be divided into pre-defined blocks before signing so that each block can be sanitized independently. However, there are cases where the parts of the message which are needed to be sanitized can not be determined in the time of signing. Thus, it is difficult to decide the partition of the blocks in such cases. Since the length of the signature is usually proportional to the number of blocks, signing every bit independently will make the signature too long. In this paper, we propose a solution by introducing a new concept called *sequential bitwise sanitizable signature schemes*, where *any sequence of bits* of the signed document can be made sanitizable without pre-defining them, and without increasing the length of signature. We also show that a one-way permutation suffices to get a secure construction, which is theoretically interesting in its own right, since all the other existing schemes are constructed using stronger assumptions.

key words: *sanitizable signature, bitwise control, one-way permutation, pseudorandom generator*

1. Introduction

1.1 Background

1.1.1 Sanitizable Signatures

The digital signatures are widely employed to provide authentication and integrity of the associated messages, and such messages could be considered to range from daily emails sent between friends to highly classified documents confidentially stored by the government. Research around digital signatures has never stopped and has become an important fundamental part of cryptology. Especially, digital signature schemes with various high functionalities are always attracting much attention of both society and scientists.

Sanitizable signature schemes are introduced to solve the following problem: we want a document to be properly signed by an authorized signer, but afterward we need certain parts of the signed document to be hidden or masked (sanitized) to protect sensitive information without harming the validity of the signed document. The most attractive feature of sanitizable signature schemes is that the sanitizing process can be done without requiring the signer to sign again the sanitized document in order to guarantee its validity. This feature is very essential in many cases where the signer is not always available.

Here, we give a typical scenario in which sanitizable signatures are effectively used. Suppose a situation where the president of a country signs an official document and the government office keeps this signed document. When a citizen requests to disclose it and this request is *partially* approved by a trial, the judge orders the government office to *partially* disclose the document. If the utilized signature is a standard one, the citizen cannot verify validity of the disclosed part of the document (or the government office has to disclose the whole document). By using a sanitizable signature, this problem can be easily solved. Namely, the government office can sanitize a part of the document which is not required to disclose, and the citizen can verify the disclosed part without revealing the remaining part. In this scenario, the president, the government office, and the citizen play the roles of the signer, the sanitizer, and the verifier, respectively.

Other interesting examples of applications of sanitizable signature schemes are also introduced in [1], [2].

1.1.2 Necessity for Bitwise Control

Sanitizable signature schemes proposed in previous works [1], [2], [6]–[11] usually require a message to be divided into fixed blocks of different sensitive information and require the signer to sign each block separately so that each block can be sanitized later independently. If the message is a "pre-formatted" document such as driver licenses or birth certificate, the partition of blocks are clear, since the locations of sensitive information to hide and be made sanitizable are clear. However, there are cases where the parts of the message which are needed to be sanitized can not be determined in the time of signing. A typical scenario is if the disclosed part is determined by a third person (e.g. the judge) who is independent of the signer and the sanitizer.

Manuscript received April 22, 2010.
Manuscript revised August 13, 2010.

[†]The authors are with the Research Center for Information Security, National Institute of Advanced Industrial Science and Technology, Tokyo, 101-0021 Japan.
[††]The author is with the Graduate School of Engineering, University of Fukui, Fukui-shi, 910-8507 Japan.
[†††]The author is with Japan Advanced Institute of Science and Technology, Ishikawa-ken, 923-1292 Japan.
[††††]The author is with The Systems Development Laboratory, Hitachi Ltd., Yokohama-shi, 244-0817 Japan.
[†††††]The author is with the Institute of Industrial Science, The University of Tokyo, Tokyo, 153-8505 Japan.
a) E-mail: santoso.bagus@aist.go.jp
DOI: 10.1587/transfun.E94.A.392

As a trivial solution, the signer can divide the message into very small blocks, e.g., one bit as one block, so that any bit can be sanitized independently. But, this trivial approach will certainly make the size of signature too large, since it usually grows in proportional to the number of blocks.

To give a more clear picture on how our proposed scheme can be useful in practice, we show the following examples.

1.1.3 Examples of Practical Application

First, consider the following situation. In a trial, a prosecutor presents a document as an evidence which is signed by a witness, say "Michael Jackson". Because he does not want to reveal the whole name of the witness to protect the privacy of the witness, he masked the document using a sanitizable signature scheme. In order to confirm that the prosecutor really knows the signer of the document, the judge may ask the prosecutor to mention an arbitrary character of the name of the witness, for example, the third character of the last name ("c"). Then, the judge ask the prosecutor to disclose the third character of the last name to verify that the document is truly signed by someone whose last name has "c" as its third character. This is a problem if the prosecutor is using a standard sanitizable signature scheme since he does not know which character is asked by the judge (unless he sign each character independently), but our proposed scheme can solve this problem easily and efficiently.

Another example is as follows. Suppose that in a public investigation toward a government of a country, the government has to reveal an official signed document which contains all names of its secret intelligence agents. Also suppose that the public investigator only needs to know whether an agent A is related to the document but the investigator also does not want the government to know which agent is before hand. Thus, the government can sign and mask the document using our proposed scheme and then reveal the part which contains the name of agent A without revealing other part of the documents.

1.2 Our Contribution

In this paper, we propose a new sanitizable signature scheme which enables the sanitizer to disclose an arbitrary *bit sequence* in the document without losing verifiability of the (fixed) given signature. Our scheme yields fairly short signatures. Namely, message overhead (i.e. length of signed document minus length of the plain document) of our scheme is only $O(\lambda)$ where λ is the security parameter. Surprisingly, *our proposed scheme can be generically obtained from any one-way permutation* while previous schemes with similar functionality, i.e. [3], [4], require the (significantly stronger) RSA function. Furthermore, [3] also requires random oracles (which do not exist in the real world [12]). However, we should also mention that the schemes in [3] and [4] allow more flexible control of disclosure: an arbitrary set of bits (not only a bit sequence) can be disclosed.

Our main idea is to combine a standard signature scheme and a special type of pseudorandom generator which, can be constructed from any one-way permutation. Below, we give an overview of a more basic form of sequential bitwise signature scheme, where any *j last bits* of a sanitized document can be disclosed.

1.2.1 Idea of Construction

Consider a signature scheme S, and a pseudorandom generator PG which has the following properties: for any positive integer i, (1) the i-th random bit c_i can be generated using the i-th seed s_i, and (2) the i-th seed s_i can be efficiently generated using the $(i-1)$-th seed s_{i-1}. The signer holds the secret signing key and the first seed s_1. The signing process of an n bits message $\mathbf{m} = m_1 m_2 \dots m_n$ ($m_i \in \{0,1\}$), is as follows. First, the signer uses PG with s_1 to obtain n more seeds $s_2, \dots, s_n, s_{n+1}$, and then generates a random sequence $\mathbf{c} = c_1 c_2 \dots c_n$, $c_i \in \{0,1\}$. Then, it signs $\langle \widetilde{\mathbf{m}} = \mathbf{m} \oplus \mathbf{c}, s_{n+1} \rangle$ using S. Finally, the signer submits $(\sigma, \widetilde{\mathbf{m}}, s_1)$ to the sanitizer, where σ is the valid signature of $\langle \widetilde{\mathbf{m}}, s_{n+1} \rangle$. When the sanitizer gets a request to disclose j last bits of $\widetilde{\mathbf{m}}$ with signature σ, it sends a response $(\sigma, \widetilde{\mathbf{m}}, s_{n-j+1})$ which is generated from $(\sigma, \widetilde{\mathbf{m}}, s_1)$. Note that s_{n-j+1} can be generated from s_1. The validity of the response $(\sigma, \widetilde{\mathbf{m}}, s_{n-j+1})$ is guaranteed by the fact that σ is a valid signature of $\langle \widetilde{\mathbf{m}}, s_{n+1} \rangle$, and that s_{n+1} can be generated from s_{n-j+1}. To disclose j last bits of $\widetilde{\mathbf{m}}$, one can use PG with s_{n-j+1} to get the last j bits of $\mathbf{c}$, i.e., $c_{n-j+1} \dots c_n$ (property (1) of PG) and then unmask the last j bits of $\widetilde{\mathbf{m}}$.

The basic form shown above can be extended easily into a full-fledge sequential bitwise signature scheme such that, instead of only any *j last bits* of the sanitized document can be disclosed, any sequence of bits from j_1-th bit to j_2-th bit can be disclosed. The main trick is that instead of using one pseudorandom generator, we employ two pseudorandom generators and then apply the generated two sequences of random bits in opposite directions, i.e., one is from the starting bit to the ending bit of the message, and the other one is from the ending bit to the starting bit of the message. See the detailed construction in Sect. 6.

It turns out that an ordinary signature scheme S which is existentially unforgeable against chosen message attack and a Blum-Micali(-Yao) pseudorandom generator [13], [14] are sufficient to instantiate the construction above. Since such a signature scheme and the pseudorandom generator can be constructed by only using a one-way permutation, one-way permutation is sufficient to instantiate our scheme.

1.3 Comparison with Other Proposed Bitwise Signature Schemes

Johnson et al. [3] and Nojima et al. [4] have also proposed sanitizable signature schemes which allow efficient bitwise sanitization, i.e., the message overhead is only $O(\lambda)$, where λ is the security parameters. Both schemes allow the san-

Table 1 Comparison between bitwise sanitizable signature schemes.

	Sanitizing ability	Security assumption	Signature overhead(*)
Johnson et al. [3]	any set of bits	RSA with random oracle	1024 bits
Nojima et al. [4]	any set of bits	RSA without random oracle	2048 bits
Our scheme	any sequence of bits	**any one way permutation**	**320** bits (**)

(*)Here we assume the standard 80 bits security.†

(**) Using one way permutation construction on elliptic curve proposed by Kaliski[5].††

itizer to sanitize any *set* of bits in the document, while our scheme only allows the sanitizer to sanitize any *sequence* of bits in the document. In spite of this fact, our scheme still has advantages in terms of security and the signature length. The comparison is illustrated in Table 1.

1.4 Related Works

The first technique to solve the digital document sanitizing problem is introduced by Steinfeld et al. [15] in 2001 in the name of content extraction signature. Similarly, Miyazaki et al. [1] also proposed SUMI-4, where the signer generates random numbers for all subdocuments, and then calculates hash values for all subdocuments with corresponding random numbers and generates a signature for the concatenation of those hash values.

On other research aspects, Miyazaki et al. [6] and Izu et al. [7] addressed sanitizable signature schemes with disclosure condition control. And based on bilinear maps, Miyazaki et al. [8] proposed a sanitizable signature scheme with invisibility. Izu et al. [9] improved this work against stronger attacker. Another scheme with aggregation was proposed in [10].

By replacing a conventional hash function with a strongly unforgeable chameleon hash function, Ateniese et al. [2] achieved a sanitizable signature scheme which allows semi-trusted sanitizers to erase or even modify parts of a signed document without interacting with the original signer. This work was extended by Klonowski et al. [16].

By employing an identity based chameleon hash function [17], Canard et al. [11] expanded the flexibility of designating power of sanitizing in such a way that the power can be designated to plural parties even after the original document is signed.

In the above schemes, it is not easy to flexibly determine the disclosed part since the partition which divides the disclosed and the sanitized parts has to be pre-defined at the signing phase. For this issue, Johnson et al. [3] proposed the first sanitizable signature scheme which allows efficient bitwise sanitizing (i.e. its message overhead is only $O(\lambda)$ for the security parameter λ) under the RSA assumption in the random oracle model. Nojima et al. [4] presented another scheme with message overhead $O(\lambda)$ under the RSA assumption without using random oracles. It should be noticed that as mentioned in [4], the scheme of Nojima et al. is in fact based on the preliminary versions of our proposed scheme [18], [19] which have been announced before [4] was published. Both [3] and [4] yields *full* bitwise sanitizing where the sanitizer can sanitize any set of bits in the document. However, in many practical applications, the disclosed part is usually a sequence of bits in the document, and sequential bitwise sanitizing (which our proposed scheme provides) is sufficient.

1.5 Roadmap

The rest of the paper is constructed as follows. In Sect. 2, we give an overview of several important concepts. In Sect. 3, the notation and several important definitions are described. In Sect. 4, we formalize the definition of sequential bitwise sanitizable signature, and define the security notions. In Sect. 5, we propose our basic construction. In Sect. 6, we show the extension of our basic scheme into a full-fledge sequential bitwise signature scheme and a concrete instantiation. Finally, we draw a concluding remark. The detailed proofs of lemmas and theorems are put in the appendix.

2. Intuition of the Model and Security Definition

Before going into rigorous definitions of the model and security requirements, here we give intuitions on them (since these are significantly different from those of standard digital signature schemes).

2.1 Model

The model of a sequential bitwise sanitizable signature scheme is formed by three parties: (1) the signer who has a signing key and is only able to generate valid signatures, (2) the sanitizer who keeps clear signed messages of the signer and sanitizes them according to the request, and (3) the verifier(s) who verifies validity of sanitized signed messages. In our model, neither the sanitizer nor the verifier(s) have secret keys. Namely, for a given (sanitized) signed message, everyone can (further) sanitize and publicly verify it by using only public data. When applying the typical scenario in the previous section, the signer, the sanitizer, and the verifier(s) correspond to the president, the government office, and the citizen(s), respectively (as mentioned above).

For simplicity, we only consider the following setting: the signer publishes a signed message $(\sigma, \mathbf{m})$ where

†All schemes in Table 1 are constructed based on a traditional signature scheme. The signature overhead here is defined as the difference between the size of the sanitizable signature and the size of the underlying traditional signature.

††See Appendix E for the detailed construction.

$\mathbf{m} \in \{0,1\}^n$ is the message to be signed and σ is the valid signature of $\mathbf{m}$ by the signer. For this signed message, the verifier can ask the sanitizer to sanitize only the i first successive bits of $\mathbf{m}$ where the verifier can choose any i such that $0 \le i \le n$. According to this request, the sanitizer modifies $(\sigma, \mathbf{m})$ to another signed message $(\tilde{\sigma}, \tilde{\mathbf{m}})$ such that $\tilde{\mathbf{m}}$ is the $n-i$ last bits of $\mathbf{m}$ and $\tilde{\sigma}$ is the valid signature of $\tilde{\mathbf{m}}$ of the signer.

We stress that this simplified setting can be easily extended to the general one where the verifier can ask to disclose any sequence of bits in $\mathbf{m}$. See Sect. 6 for this extension. For the rest of the paper, unless noted otherwise, the term "sequential bitwise signature scheme" refers to one in the above simplified setting rather than full-fledged one.

2.1.1 Security Requirements

In the above model, the adversary would try (1) forgery of a signed message which is accepted as the signer's, or (2) recovery of a sanitized part of a signed message which the sanitizer does not disclose.

Specifically, the forgery is considered successful if the adversary generates a valid signature σ' of a message $\mathbf{m}'$ where

- $(\sigma', \mathbf{m}')$ is never been publicized by the sanitizer, and
- $(\sigma', \mathbf{m}')$ cannot be obtained by trivially sanitizing a signed message which has been publicized by the sanitizer.

It should be noticed that forgery of $(\sigma', \mathbf{m}')$ such that $(\sigma', \mathbf{m}')$ is actually generated by the signer is considered successful if $(\sigma', \mathbf{m}')$ is never publicized by the sanitizer (since the adversary indeed succeeds in generating a valid signed message which has never publicized). If for any adversary, the probability of succeeding in the above forgery is negligible, we say the scheme satisfies *unforgeability*.

On the other hand, the recovery of a sanitized part is considered successful if the adversary wins the following game. The adversary chooses a pair of messages $(\mathbf{m}^{(0)}, \mathbf{m}^{(1)})$ and i $(0 \le i \le n)$ such that the i last bits of $\mathbf{m}^{(0)}$ and $\mathbf{m}^{(1)}$ are identical, and submits them to the signer. The signer flips a random coin $r \in \{0,1\}$ and signs $\mathbf{m}^{(b)}$ with his signing key, and gives the signed message $(\sigma, \mathbf{m}^{(b)}, i)$ to the sanitizer where σ is the signature for $\mathbf{m}^{(b)}$. The sanitizer sanitizes $\mathbf{m}^{(b)}$ except for the i last bits, and gives $(\sigma^\star, \mathbf{m}^\star)$ to the adversary where $(\sigma^\star, \mathbf{m}^\star)$ is the sanitized signature derived from $(\sigma, \mathbf{m}^{(b)})$ (i.e. $\mathbf{m}^\star$ is the i last bits of $\mathbf{m}^{(b)}$). The adversary wins the game if it correctly outputs b. If for any adversary, the probability of winning the above game is negligibly close to 1/2, we say the scheme satisfies *data confidentiality*.

3. Preliminaries

Notations. Let $[\mathbf{x}]^y$ denote the y-bit prefix of a string $\mathbf{x}$ and $[\mathbf{x}]_z$ denote the z bit suffix of a string $\mathbf{x}$. Both $[\mathbf{x}]^0$ and $[\mathbf{x}]_0$ represent an empty string. $[\mathbf{x}]_z^y$ denotes the $z-y+1$ bits of a string $\mathbf{x}$ from the y-th bit to the z-th bit, where $z \ge y$. The length of string $\mathbf{x}$ in bits is denoted by $|\mathbf{x}|$. If f is a function or an algorithm, let $x \leftarrow f^{(0)}(x)$ and let $f^{(i)}(f(x)) \leftarrow f^{(i+1)}(x)$, where $i \ge 0$. Unless noted otherwise, any algorithm in this paper is a probabilistic polynomial-time Turing machine.

Definition 1 (One-way Permutation): We say a bijective mapping $f : \{0,1\}^\lambda \to \{0,1\}^\lambda$ is a *one-way permutation* if f is efficiently computable, but for any algorithm A, $\Pr[x \leftarrow_R \{0,1\}^\lambda; y \leftarrow f(x) : A(y) = x] \le \epsilon$, where ϵ is negligible in λ.

Definition 2 (Hard-core Predicate): We say a function $h : \{0,1\}^\lambda \to \{0,1\}$ is a hard-core predicate of another function $f : \{0,1\}^\lambda \to \{0,1\}^\lambda$ if h is efficiently computable, and for any predicator P, $|\Pr[x \leftarrow_R \{0,1\}^\lambda; y \leftarrow f(x) : P(y) = h(x)] - 1/2| \le \epsilon$, where ϵ is negligible in λ. It is known that any one-way function has a hard-core predicate [20].

Definition 3 (Blum-Micali-Yao PRNG): It is known that a pseudorandom number generator (PRNG) can be generically derived from any one-way permutation [13], [14]. Let $f : \{0,1\}^\lambda \to \{0,1\}^\lambda$ and $h : \{0,1\}^\lambda \to \{0,1\}$ be a one-way permutation and its hard-core predicate, respectively. Let $G(s,n)$ be $(h(f^{(0)}(s))\|h(f^{(1)}(s))\| \cdots \|h(f^{(n-1)}(s)))$. Then, $(G(s,n)\|f^{(n)}(s)))$ is indistinguishable from a random $(n+\lambda)$-bit string. Formally, we have the following lemma:

Lemma 1: For any distinguisher D, $|\Pr[s \leftarrow_R \{0,1\}^\lambda; D(G(s,n)\|f^{(n)}(s)) = 1] - \Pr[z \leftarrow_R \{0,1\}^{n+\lambda} : D(z) = 1]| \le \epsilon$, where ϵ is negligible in λ.

The proof of the above lemma is given in [14]. We call $G(s,n)$ "Blum-Micali-Yao pseudorandom number generator" (BMY-PRNG)[†].

Definition 4 (Digital Signature): A (conventional) digital signature scheme $\mathcal{S} = (\mathsf{gen}, \mathsf{sig}, \mathsf{ver})$ is specified by the following three polynomial-time algorithms: (1) The key generation algorithm gen takes a security parameter λ and outputs a verification key vk and the corresponding signing key sk. We denote this by $(vk, sk) \leftarrow \mathsf{gen}(1^\lambda)$. (2) The signing algorithm sig takes sk and a message $\mathbf{m}$ from some message space, and outputs a signature σ. We denote this by $\sigma \leftarrow \mathsf{sig}(sk, \mathbf{m})$. (3) The verification algorithm ver which takes vk, σ and $\mathbf{m}$, and outputs 0 or 1. We denote this by $0/1 \leftarrow \mathsf{ver}(vk, \sigma, \mathbf{m})$. Here "0" and "1" indicate that σ is rejected and accepted, respectively, on $\mathbf{m}$.

The standard security notion for (conventional) digital signature schemes is *existential unforgeability against chosen message attack* (EUF-CMA). The scheme is said to satisfy EUF-CMA, if any algorithm (forger) F wins the following game with only negligible probability: F interacts with a signing oracle O to obtain signatures of any message as many times as F wants, and finally outputs a valid signature σ' on a message $\mathbf{m}'$ that was never explicitly queried to O.

[†]Lemma 1 means that whole bits in $(G(s,n)\|f^{(n)}(s))$ are pseudorandom rather than only $G(s,n)$. However, dividing "$G(s,n)$" part and "$f^{(n)}(s)$" part significantly simplifies description of our schemes.

4. Formal Security Definitions

In this section, we discuss the formal security definitions of sequential bitwise sanitizable signature (SBSS) schemes. We first define the algorithms in a sequential sanitizable signature scheme, then the necessary building blocks to construct formal security definitions, and finally the formal security notions which capture the goal and attack models of unforgeability and data-confidentiality, as introduced in Sect. 2.

Conventions. As shown in the 'idea of construction' of Sect. 1.2, for simplicity, we consider the case where messages are sanitized sequentially in a left-to-right order. For any n-bit message $\mathbf{m} = m_1 \dots m_n$, $m_i \in \{0,1\}$, $\mathbf{m}_j$ denotes *j-bits-sanitized* $\mathbf{m}$, i.e., $\mathbf{m}$ with its first j bits being sanitized ($j \in [0,n]$). Accordingly, $\mathbf{m}_0$ means $\mathbf{m}$ *without sanitizing* or an *open message* $\mathbf{m}$ ($\mathbf{m}_0 = \mathbf{m}$), and $\mathbf{m}_n$ means a fully sanitized $\mathbf{m}$. $\mathbf{m}_{j'}$ is said to be *more sanitized* (resp. *less* sanitized) than $\mathbf{m}_j$ if $j' > j$ (resp. $j' < j$). Particularly, $\mathbf{m}_{j'}$ is said to be *one-bit-more* sanitized than $\mathbf{m}_j$ if $j' = j+1$, and *one-bit-less* sanitized than $\mathbf{m}_j$ if $j' = j-1$. Obtaining $\mathbf{m}_{j-1}$ from $\mathbf{m}_j$ is called *rewinding* $\mathbf{m}_j$. For simplicity, unless noted otherwise, the length of any message is n bits.

4.1 Algorithms of Sequential Bitwise Sanitizable Signature Schemes

Formally, a SBSS scheme $\mathcal{SBSS} = (\mathsf{GEN}, \mathsf{SIG}, \mathsf{SAN}, \mathsf{VER})$ is specified by the following four algorithms.

- The key generation algorithm GEN takes a security parameter and randomly outputs a signing key and the corresponding verification key. We denote this by $(vk, sk) \leftarrow \mathsf{GEN}(1^\lambda)$.
- The signing algorithm SIG takes the secret signing key and a message from some message space, and outputs a signature. We denote this by $\sigma_0 \leftarrow \mathsf{SIG}(sk, \mathbf{m}_0)$, where $\mathbf{m}_0 \leftarrow \{0,1\}^n$ and $|\mathbf{m}_0| = n$.
- The sanitizing algorithm SAN takes a signature-message pair $(\sigma_{k-1}, \mathbf{m}_{k-1})$ and outputs another signature-message pair $(\sigma_k, \mathbf{m}_k)$, such that the output message is *one-bit-more sanitized* than the input message. We denote this by $(\sigma_k, \mathbf{m}_k) \leftarrow \mathsf{SAN}(\sigma_{k-1}, \mathbf{m}_{k-1})$, where $1 \le k \le n$[†].
- The verification algorithm VER takes the verification key, a signature and a message, and output 1 or 0. We denote this by $\{0,1\} \leftarrow \mathsf{VER}(vk, \sigma_k, \mathbf{m}_k)$.

In order to formalize the goal and attack models, next, we define the oracles to which an adversary can access according to the corresponding goal and attack model.

4.2 Signing-Sanitizing-Oracle (O_1)

The oracle O_1 is to provide an interface for "chosen message attack", analogous to the one in ordinary signature schemes. By accessing this oracle (O_1), the adversary can specify an open message or the disclosed part of a partially sanitized message and obtain the corresponding signature. The oracle generates randomly the prefix of the message to get a full n bits message if the length of the specified disclosed part is less than n bits. Formally, the execution of O_1 is written as $(\sigma_k, \mathbf{m}_k) \leftarrow O_1(\mathbf{x})$, where $[\mathbf{m}_k]_{|\mathbf{x}|} = \mathbf{x}$ holds. If $\mathbf{x}$ is an empty string, O_1 generates a random message and its corresponding signature.

Remark on O_1.

In the framework of SBSS described in the previous subsection, one can see that $O_1(\mathbf{x}, k)$ can be realized by a single access to SIG (when $k = 0$) or simultaneous accesses to SIG and SAN (when $k > 0$) combined with a random bit generator. Note that here the adversary can not control or specify any bit of the sanitized part, since the sanitized part is generated randomly by O_1. One can verify that instead of weakening the level of security notions described in the next subsections, this "restriction" actually gives more freedom to the adversary to achieve its adversarial goal compared to the previous works [3], [4]. See Appendix F for the detailed discussion.

4.3 Rewinding-Oracle (O_2)

By accessing this oracle, the adversary can obtain a one-bit-less version of a *valid* sanitized message (rewind a *valid* sanitized message) and also obtain the corresponding signature, with the condition that a signature of a less sanitized version of it has ever been retrieved from the signing-sanitizing oracle (O_1). Formally, the execution of O_2 is written as $(\sigma_{k-1}, \mathbf{m}_{k-1}) \leftarrow O_2(\sigma_k, \mathbf{m}_k)$, with the condition that at least one $(\sigma_{k'}, \mathbf{m}'_{k'})$ such that $k' < k$ and $[\mathbf{m}'_{k'}]_{n-k} = [\mathbf{m}_k]_{n-k}$ hold has been output by O_1. We can simply say that O_2 responds to the query only if the outputs of O_1 are queried.

4.4 Security Notions of Sequential Bitwise Sanitizable Signature Schemes

Here we give the formal definition of security notions of SBSS in the sense of unforgeability and data-confidentiality. We will formalize the goal and attack model of the adversary by utilizing oracles described in previous subsections.

4.4.1 Formal Definition of Unforgeability (UF)

Intuitively, this security notion is to guarantee that no adversary can perform forgery described in Sect. 2. In this attack model, the adversary can specify an open message or the disclosed part of a partially sanitized message and obtain the corresponding signature. If the adversary specifies nothing,

[†]Although instead of accepting the request of sanitizing the last sequence of bits of a sanitized message with any length we only let the sanitizer to sanitize one more bit of the input message, this representation does not decrease the functionality of the sanitizer at all, since one, who wants a signature of $\mathbf{m}_k$ ($k > 0$) given a signature of $\mathbf{m}_0$, can query SAN k times to get it.

i.e., an empty string, then he will be answered with a signature on a randomly chosen message. Here, the adversary is allowed to access O_1 and O_2 as described above.

Definition 5: Let $\mathcal{SBSS} = (\mathsf{GEN}, \mathsf{SIG}, \mathsf{SAN}, \mathsf{VER})$ be a sequential bitwise sanitizable signature scheme, $O = \{O_1, O_2\}$, and A be an adversary. For $\lambda \in \mathbb{N}$, let

$$\mathbf{Adv}^{\mathsf{UF}}_{\mathcal{SBSS},A}(\lambda) = \Pr[\mathbf{Exp}^{\mathsf{UF}}_{\mathcal{SBSS},A}(\lambda) = 1], \quad (1)$$

where, for $n = |\mathbf{m}^*_k|$ and $0 \le k \le n$,

$$\mathbf{Exp}^{\mathsf{UF}}_{\mathcal{SBSS},A}(\lambda) \stackrel{def}{=} (vk, sk) \leftarrow \mathsf{GEN}(1^\lambda); (\sigma^*_k, \mathbf{m}^*_k) \leftarrow A^{O}(vk); \texttt{return } \mathsf{VER}(vk, \sigma^*_k, \mathbf{m}^*_k).$$

We say that $\mathcal{SBSS}$ is secure in the sense of unforgeability (UF), if $\mathbf{Adv}^{\mathsf{UF}}_{\mathcal{SBSS},F}(\lambda)$ is negligible for any adversary A.

At the end of the UF game, A outputs a signature σ^*_k along with an (open or partially) sanitized message $\mathbf{m}^*_k$, where $0 \le k \le |\mathbf{m}^*_k|$. The restriction is that this pair $\langle \sigma^*_k, \mathbf{m}^*_k \rangle$ should not appear in the answers of O_1, and is not straightforwardly computable from any answer of O_1, i.e., $(\sigma^*_k, \mathbf{m}^*_k) \notin \{\mathsf{SAN}^{(i)}(\sigma_j, \mathbf{m}_j)\}_{1 \le i \le n-k}$, where $(\sigma_j, \mathbf{m}_j)$ is one of the outputs of O_1. We say that the adversary wins UF game if $(\sigma^*_k, \mathbf{m}^*_k)$ is a valid pair with the above restriction.

4.4.2 Formal Definition of Data Confidentiality (DC)

Intuitively, this security notion captures the sense of protecting the SBSS scheme from sanitization leakage. The adversary cannot learn any information about the sanitized part from any sanitized signature-message pair.

Let A denote the adversary. A interacts with the challenger in a two-stage DC game. At the first stage, after given vk, A can query O_1 and O_2 to obtain polynomial numbers of signature-message pairs.

At the end of the first stage, A outputs $(\mathbf{m}^0_0, \mathbf{m}^1_0, k, st)$, such that $\mathbf{m}^0_0, \mathbf{m}^1_0$ are two different messages at the same length n, k is an integer, and st is state information (possibly including vk). Here the $(n-k)$ right-most bits of the two messages should be identical.

During the challenge phase, the challenger randomly picks one message from $(\mathbf{m}^0_0, \mathbf{m}^1_0)$ beyond A's view, signs this message $\mathbf{m}^b_0$ with the secret signing key sk, and executes sanitization until the k-th bit of $\mathbf{m}^b_0$. The output pair $\langle \sigma^*_k, \mathbf{m}^*_k \rangle$ is sent to A as the challenge.

At the second stage, A tries to distinguish which message has been signed and sanitized, i.e., he tries to compute b which was randomly flipped by the challenger. Besides O_1-queries[†], he can query to O_2 with the restrictions that O_2 rejects the query $(\sigma^*_i, \mathbf{m}^*_i)$, such that $[\mathbf{m}^0_0]^{i-1}_{i-1} \ne [\mathbf{m}^1_0]^{i-1}_{i-1}$. At the end of the second stage, the adversary outputs a bit b'. We say the adversary wins DC game if b' is identical with b.

Definition 6: Let $\mathcal{SBSS} = (\mathsf{GEN}, \mathsf{SIG}, \mathsf{SAN}, \mathsf{VER})$ be a sequential bitwise sanitizable signature scheme and let $A = (A_1, A_2)$ be an adversary. For $\lambda \in \mathbb{N}$, let

$$\mathbf{Adv}^{\mathsf{DC}}_{\mathcal{SBSS},A}(\lambda) = \Pr[\mathbf{Exp}^{\mathsf{DC\text{-}1}}_{\mathcal{SBSS},A}(\lambda) = 1] - \Pr[\mathbf{Exp}^{\mathsf{DC\text{-}0}}_{\mathcal{SBSS},A}(\lambda) = 1] \quad (2)$$

where for $\mathbf{m}^0_0 \ne \mathbf{m}^1_0$, $0 \le k \le n$, and $[\mathbf{m}^0_0]_{n-k} = [\mathbf{m}^1_0]_{n-k}$,

$$\mathbf{Exp}^{\mathsf{DC\text{-}b}}_{\mathcal{SBSS},A}(\lambda) \stackrel{\text{def}}{=} (vk, sk) \leftarrow \mathsf{GEN}(1^\lambda); (\mathbf{m}^0_0, \mathbf{m}^1_0, k, st) \leftarrow A^{O}_1(vk); \sigma^*_0 \leftarrow \mathsf{SIG}(sk, \mathbf{m}^b_0); \mathbf{m}^*_0 \leftarrow \mathbf{m}^b_0; (\sigma^*_k, \mathbf{m}^*_k) \leftarrow \mathsf{SAN}^{(k)}(\sigma^*_0, \mathbf{m}^*_0); b' \leftarrow A^{O}_2(st, \sigma^*_k, \mathbf{m}^*_k); \texttt{return } b'$$

We say that $\mathcal{SBSS}$ is secure in the sense of data confidentiality (DC), if $\mathbf{Adv}^{\mathsf{DC}}_{\mathcal{SBSS},A}(\lambda)$ is negligible for any A.

5. Our Construction and Its Security

Here, we show the construction of our (basic) sanitizable signature scheme and its security proofs.

5.1 Construction

Our sequential bitwise sanitizable signature scheme is denoted by $\mathsf{BASIC} = (\mathsf{GEN}, \mathsf{SIG}, \mathsf{SAN}, \mathsf{VER})$. Let λ be the security parameter, and $\{0,1\}^n$ be the message space. Let G be the Blum-Micali-Yao pseudorandom number generator (BMY-PRNG; see Definition 3) where $f : \{0,1\}^\lambda \to \{0,1\}^\lambda$ is the underlying one-way permutation. Let $\mathcal{S} = (\mathsf{gen}, \mathsf{sig}, \mathsf{ver})$ be the underlying (conventional) digital signature secure in the sense of EUF-CMA. The four algorithms of BASIC are described below.

- $\mathsf{GEN}(1^\lambda)$: Given a security parameter λ, the algorithm calls gen (the key generation algorithm of the underlying conventional signature scheme) by passing λ to gen, obtains (vk, sk), and outputs them as a key pair.
- $\mathsf{SIG}(sk, \mathbf{m}_0)$: Given the signing key sk and a message $\mathbf{m}_0$ with length n, the algorithm works as follows: (1) picks a random $s_0 \leftarrow \{0,1\}^\lambda$, (2) computes $\mathbf{m}_n \leftarrow \mathbf{m}_0 \oplus G(s_0, n)$, (3) calls sig (the signing algorithm of the underlying conventional signature scheme) to generate a signature σ on $\mathbf{m}_n \| f^{(n)}(s_0)$, (4) composes the signature σ_0 with $\sigma \| s_0$, (5) outputs σ_0.
- $\mathsf{SAN}(\sigma_{k-1}, \mathbf{m}_{k-1})$: Given a signature-message pair, the algorithm sequentially sanitizes the input with one bit as follows: (1) parses σ_{k-1} as $\langle \sigma, s_{k-1} \rangle$, (2) computes $c_k \leftarrow [\mathbf{m}_{k-1}]^k_k \oplus G(s_{k-1}, 1)$ and assigns $\mathbf{m}_k \leftarrow [\mathbf{m}_{k-1}]^{k-1} \| c_k \| [\mathbf{m}_{k-1}]_{n-k}$, (3) computes the one-way permutation $s_k \leftarrow f(s_{k-1})$, (4) sets $\sigma_k = (\sigma \| s_k)$, (5) outputs $(\sigma_k, \mathbf{m}_k)$.
- $\mathsf{VER}(vk, \sigma_k, \mathbf{m}_k)$: Given a signature-message pair, the algorithm verifies the validity as follows: (1) parses σ_k as $\langle \sigma, s_k \rangle$, (2) computes $\mathbf{m}_n \leftarrow [\mathbf{m}_k]^k \| ([\mathbf{m}_k]_{n-k} \oplus G(s_k, n-k))$, (3) calls ver (the verifying algorithm of the underlying conventional signature scheme) in the way that $b \leftarrow \mathsf{ver}(vk, \sigma, \mathbf{m}_n \| f^{(n-k)}(s_k))$.

[†]We note here that by querying $\mathbf{m}^0_0$ or $\mathbf{m}^1_0$ to O_1, the adversary can not increase his success probability because the signing algorithm is a probabilistic one.

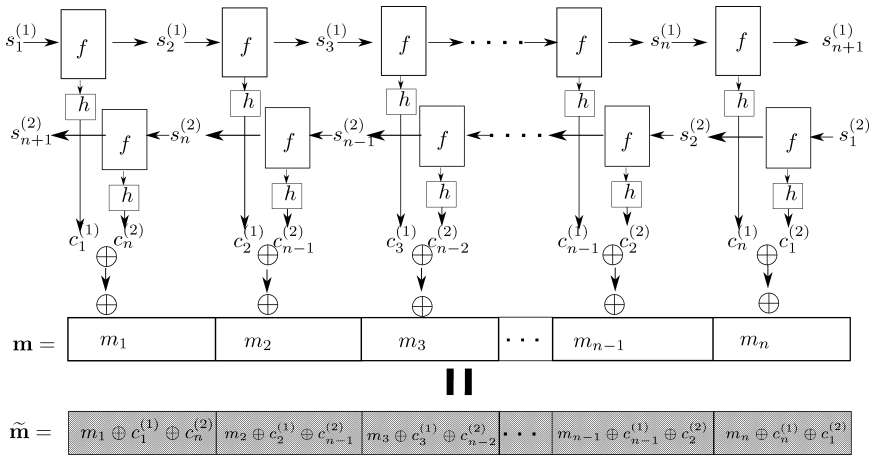


Fig. 1 The signing in proposed full-fledge sequential bitwise sanitizable signature scheme.

5.2 Unforgeability

Theorem 2: *BASIC* is secure in the sense of *UF* if $\mathcal{S}$ is an *EUF-CMA* secure signature scheme and f is a one-way permutation.

Proof An adversary A may achieve several types of forgery as follows.

Type 1. A outputs a signature on a message which is different from any queried messages.

Type 2. A outputs a signature on a message $\mathbf{m}^*_{k-1}$ which is a one bit less sanitized version of a retrieved sanitized message $\mathbf{m}_k$.

Type 3. A outputs a signature on a message $\mathbf{m}^*_k$ such that $\mathbf{m}^*_0 \neq \mathbf{m}_0$ but $\mathbf{m}^*_n = \mathbf{m}_n$, where $\mathbf{m}_n$ can be easily computed from a queried message.

We denote the A's advantages with corresponding types by $\mathbf{Adv}^{\mathsf{UF\text{-}1}}_{\mathcal{SBSS},A}(\lambda)$, $\mathbf{Adv}^{\mathsf{UF\text{-}2}}_{\mathcal{SBSS},A}(\lambda)$ and $\mathbf{Adv}^{\mathsf{UF\text{-}3}}_{\mathcal{SBSS},A}(\lambda)$, respectively. We then evaluate them individually. The detailed proof is in Appendix A. □

5.3 Data Confidentiality

Theorem 3: *BASIC* is secure in the sense of *DC* if f is a one-way permutation.

Proof Toward contradiction, we first assume that there exists an adversary A who can break BASIC in the DC game, then we construct a distinguisher D who can distinguish the output of BMY-PRNG from uniform string by using A as a subroutine, which contradicts Lemma 1. The detailed proof is in Appendix D. □

6. Extension of Basic Sequential Bitwise Signature Scheme

In this section, we show the extension of BASIC to a full-fledge sequential bitwise signature scheme. We stress here that the security of the extension can also be proven on the weak assumption that a one-way permutation exists.

6.1 A Full-Fledge Sequential Sanitizable Signature Scheme

Now, instead of only any j *last bits* of the sanitized document, any sequence of bits from j_1-th bit to j_2-th bit of the sanitized document can be disclosed, where j_1 and j_2 are specified in the request to the sanitizer. We will explain our extension in the manner of 'idea of construction' described in Sect. 1.2. The main trick is that instead of one pseudorandom generator, here we employ two pseudorandom generators and then apply the resulting two sequences of random bits in opposite directions, i.e., one is from the starting bit to the ending bit of the message, and the other one is from the ending bit to the starting bit of the message. Similar to the description in 'idea of construction', let S be a signature scheme S, and PG_1, PG_2 be pseudorandom generators with the same features as PG or Blum-Micali-Yao pseudorandom generators as defined in Definition 3.

The signer holds the secret signing key and two first seeds: $s_1^{(1)}$ of PG_1 and $s_1^{(2)}$ of PG_2. The signing process of an n-bit message $\mathbf{m} = m_1 m_2 \dots m_n$ ($m_i \in \{0,1\}$), is as follows. First, for each $\ell = \{1,2\}$, the signer uses PG_ℓ with $s_1^{(\ell)}$ to obtain n more seeds $s_2^{(\ell)}, \dots, s_n^{(\ell)}, s_{n+1}^{(\ell)}$, and then generates a random sequence $c_1^{(\ell)} c_2^{(\ell)} \dots c_n^{(\ell)}$, $c_i^{(\ell)} \in \{0,1\}$. Remember here that each $c_i^{(\ell)}$ is generated from s_i and each s_i is generated from s_{i-1}. Let $\mathbf{c}^{(1)}$ denote $c_1^{(1)} c_2^{(1)} \dots c_n^{(1)}$, and $\mathbf{c}^{(2)}$ denote $c_n^{(2)} c_{n-1}^{(2)} \dots c_1^{(2)}$. Then, it signs $\langle \widetilde{\mathbf{m}} = \mathbf{m} \oplus \mathbf{c}^{(1)} \oplus \mathbf{c}^{(2)}, s_{n+1}^{(1)}, s_{n+1}^{(2)} \rangle$ using S. Finally, the signer submits $(\sigma, \widetilde{\mathbf{m}}, s_1^{(1)}, s_1^{(2)})$ to the sanitizer, where σ is the valid signature of $\langle \widetilde{\mathbf{m}}, s_{n+1}^{(1)}, s_{n+1}^{(2)} \rangle$. For illustration, see Fig. 1.

When the sanitizer gets a request to disclose j_1-th bit to j_2-th bits of $\widetilde{\mathbf{m}}$ with signature σ, it sends a response $(\sigma, \widetilde{\mathbf{m}}, s_{j_1}^{(1)}, s_{n-j_2+1}^{(2)})$ which can be generated easily from $(\sigma, \widetilde{\mathbf{m}}, s_1^{(1)}, s_1^{(2)})$. Note that for each $\ell \in \{1,2\}$, any $s_j^{(\ell)}$ with $j > 1$ is generatable from $s_1^{(\ell)}$. The validity of the response $(\sigma, \widetilde{\mathbf{m}}, s_{j_1}^{(1)}, s_{n-j_2+1}^{(2)})$ is guaranteed by the fact that σ is a valid sig-

nature of $\langle \widetilde{\mathbf{m}}, s_{n+1}^{(1)}, s_{n+1}^{(2)} \rangle$, while $s_{n+1}^{(1)}$ can be generated easily from $s_{j_1}^{(1)}$ and $s_{n+1}^{(2)}$ can be generated easily from $s_{n-j_2+1}^{(2)}$.

To disclose the j_1-th bit to j_2-th bit of $\widetilde{\mathbf{m}}$, one can use PG_1 with $s_{j_1}^{(1)}$ to get all the bits of $\mathbf{c}^{(1)}$ from j_1-th bit, i.e., $c_{j_1}^{(1)} \dots c_n^{(1)}$, and use PG_2 with $s_{n-j_2+1}^{(2)}$ to get the first j_2 bits of $\mathbf{c}^{(2)}$, i.e., $c_n^{(2)} \dots c_{n-j_2+1}^{(2)}$, and then finally unmask the j_1-th bit to j_2-th bit of $\widetilde{\mathbf{m}}$. Remember that $\widetilde{\mathbf{m}} = m_1 \oplus c_1^{(1)} \oplus c_n^{(2)} \| m_2 \oplus c_2^{(1)} \oplus c_{n-1}^{(2)} \| \dots \| m_n \oplus c_n^{(1)} \oplus c_1^{(2)}$. Using $c_{j_1}^{(1)} \dots c_n^{(1)}$ and $c_n^{(2)} \dots c_{n-j_2+1}^{(2)}$, we can obtain $m_1 \oplus c_1^{(1)} \| \dots \| m_{j_1-1} \oplus c_{j_1-1}^{(1)} \| m_{j_1} \dots m_{j_2} \| m_{j_2+1} \oplus c_{n-j_2}^{(2)} \| \dots \| m_n \oplus c_1^{(2)}$.

6.2 A Concrete Instantiation of Sequential Bitwise Signature Scheme

Here we show a concrete instantiation of a full-fledge sequential bitwise signature scheme based on elliptic curves, by combining a one way permutation proposed by Kaliski [5] and an EUF-CMA secure signature scheme proposed by Waters [21]. Let p be a sufficiently large prime[†]. For simplicity, we assume that the length of any message is n bits.

Setup Parameters for Pseudorandom Generators Let $f : [0, 2p+1] \to [0, 2p+1]$ be a one-way permutation constructed from two twist elliptic curves using Kaliski's method (see Appendix E for the detailed construction). Let $h : [0, 2p+1] \to \{0, 1\}$ be defined as the hard-core of f[††]. Then by using the one-way permutation f and its hard-core predicate h, two PRNGs $PG_1^{(k)}$ and $PG_2^{(k)}$ from $[0, 2p+1]$ to $\{0, 1\}^k$ are defined as follows:

$$\begin{aligned} PG_1^{(k)}(s) &= h(f^{(0)}(s)) \,\|\, h(f^{(1)}(s)) \,\|\, \cdots \\ &\quad \|\, h(f^{(k-2)}(s)) \,\|\, h(f^{(k-1)}(s)), \\ PG_2^{(k)}(s) &= h(f^{(k-1)}(s)) \,\|\, h(f^{(k-2)}(s)) \,\|\, \cdots \\ &\quad h(f^{(1)}(s)) \,\|\, h(f^{(0)}(s)). \end{aligned}$$

Setup Parameters for the Signature Scheme Let $\mathbb{G}$, $\mathbb{G}_1$ be cyclic groups of order p generated from group of points on an elliptic curve[†††] E and let $e : \mathbb{G} \times \mathbb{G} \to \mathbb{G}_1$ be a non-degenerate and computable bilinear map, that is, e satisfies the followings: (1) for all a, b we have $e(g^a, g^b) = e(g, g)^{ab}$, (2) $e(g, g) \neq \mathbf{1}$, where g is a generator of $\mathbb{G}$. For the detailed concrete construction of such groups, see [22]. Finally let $H : \{0, 1\}^* \to \{0, 1\}^n$ denote a collision-resistant hash function.

Key Generation Choose randomly a secret signing key $\alpha \in \mathbb{Z}_p$ and set $g_1 = g^\alpha$. Also choose randomly $g_2, u' \in \mathbb{G}$. Also generate n-length vector $U = (u_i)$, where all u_i's are chosen randomly from $\mathbb{G}$. The public verification key is (g, g_1, g_2, u', U).

Signing Upon receiving an input n-bit open message $\mathbf{m}$, the signer chooses randomly $s_1^{(1)}, s_1^{(2)} \in [0, 2p+1]$, and computes $\widetilde{\mathbf{m}} = \mathbf{m} \oplus PG_1^{(n)}(s_1^{(1)}) \oplus PG_2^{(n)}(s_1^{(2)})$. Let $M = H(\widetilde{\mathbf{m}} \| s_{n+1}^{(1)} \| s_{n+1}^{(2)})$, where $s_{n+1}^{(1)} = f^{(n)}(s_1^{(1)})$ and $s_{n+1}^{(2)} = f^{(n)}(s_1^{(2)})$. Let $\mathcal{M} \subseteq \{1, \dots, n\}$ be the set of all i such that $M_i = 1$, where M_i is the i-th bit of M. Then the signer chooses randomly $r \in \mathbb{Z}_p$ and constructs the signature σ_M as follows.

$$\sigma_M = \left(g_2^\alpha \left(u' \prod_{i \in \mathcal{M}} u_i \right)^r, g^r \right)$$

The signer submits $\left(\sigma_M, \widetilde{\mathbf{m}}, s_1^{(1)}, s_1^{(2)}\right)$ to the sanitizer.

Sanitizing To produce a sanitized signature of $\mathbf{m}$ corresponding to $\left(\sigma_M, \widetilde{\mathbf{m}}, s_1^{(1)}, s_1^{(2)}\right)$ such that the first $(j_1 - 1)$ bits and the last $(n - j_2)$ bits of $\mathbf{m}$ are sanitized, the sanitizer computes $s_{j_1}^{(1)} = f^{(j_1-1)}(s_1^{(1)})$ and $s_{n-j_2+1}^{(2)} = f^{(n-j_2)}(s_1^{(2)})$, where $j_1, j_2 \in [1, n]$. The sanitizer outputs $\left(\sigma_M, \widetilde{\mathbf{m}}, s_{j_1}^{(1)}, s_{n-j_2+1}^{(2)}, j_1, j_2\right)$.

Verification To verify whether a sanitized signature $\left(\sigma_M, \widetilde{\mathbf{m}}, s_{j_1}^{(1)}, s_{n-j_2+1}^{(2)}, j_1, j_2\right)$ is valid, first, the verifier computes $s_{n+1}^{(1)} = f^{(n-j_1+1)}(s_{j_1}^{(1)})$ and $s_{n+1}^{(2)} = f^{(j_2)}(s_{n-j_2+1}^{(2)})$, then it parses $\sigma_M = (\sigma_1, \sigma_2)$ into σ_1,σ_2 and verifies whether the following holds.

$$\frac{e(\sigma_1, g)}{e(\sigma_2, u' \prod_{i \in \mathcal{M}} u_i)} = e(g_1, g_2),$$

where $\mathcal{M}$ is defined corresponding to $M = H(\widetilde{\mathbf{m}} \| s_{n+1}^{(1)} \| s_{n+1}^{(2)})$ as in the **Signing** step. To disclose the j_1-th bit until j_2-th bit of $\widetilde{\mathbf{m}}$, the verifier computes the following.

$$\widetilde{\mathbf{m}}' = \widetilde{\mathbf{m}} \oplus \underbrace{0 \cdots 0}_{(j_1-1)\text{bits}} \| PG_1^{(n-j_1+1)}(s_{j_1}^{(1)}) \oplus PG_2^{(j_2)}(s_{n-j_2+1}^{(2)}) \| \underbrace{0 \cdots 0}_{(n-j_2)\text{bits}}.$$

7. Conclusions

In this paper, we have proposed a new concept of sequential bitwise sanitizable signature schemes and its security model. We have shown a concrete construction, and the scheme has been proved secure based on a weak security assumption that a one-way permutation exists. We have shown that our scheme increases the flexibility of sanitization scope control without increasing the length of signature. Currently, one-way permutation is the weakest cryptographic assumption on which sanitizable signature schemes rely.

[†]To provide the standard 80 bit security, one needs to select $|p| = 160$ bits.

[††]A generic construction of a hard-core predicate for any one-way function has been shown in [20].

[†††]The notation p used in the set up for pseudorandom generators corresponds to the definition field $\mathbb{F}_p$ of 2 twist elliptic curves. On the other hand, the notation p used in the set up for signature scheme corresponds to the order of an elliptic curve, denoted by $\mathbb{G}$, whose definition field is not equal to $\mathbb{F}_p$.

References

[1] K. Miyazaki, S. Susaki, M. Iwamura, T. Matsumoto, R. Sasaki, and H. Yoshiura, "Digital document sanitizing problem," IEICE Technical Report, ISEC, vol.103, no.195, pp.61–67, July 2003.
[2] G. Ateniese, D.H. Chou, B. de Medeiros, and G. Tsudik, "Sanitizable signatures," ESORICS, pp.159–177, 2005.
[3] R. Johnson, D. Molnar, D.X. Song, and D. Wagner, "Homomorphic signature schemes," CT-RSA, pp.244–262, 2002.
[4] R. Nojima, J. Tamura, Y. Kadobayashi, and H. Kikuchi, "A storage efficient redactable signature in the standard model," ISC, Lect. Notes Comput. Sci., vol.5735, pp.326–337, Springer, 2009.
[5] B.S. Kaliski, Jr., "One-way permutations on elliptic curves," J. Cryptol., vol.3, no.3, pp.187–199, 1991.
[6] K. Miyazaki, M. Iwamura, T. Matsumoto, R. Sasaki, H. Yoshiura, S. Tezuka, and H. Imai, "Digitally signed document sanitizing scheme with disclosure condition control," IEICE Trans. Fundamentals, vol.E88-A, no.1, pp.239–246, Jan. 2005.
[7] T. Izu, N. Kanaya, M. Takenaka, and T. Yoshioka, "Piats: A partially sanitizable signature scheme," ICICS, ed. S. Qing, W. Mao, J. Lopez, and G. Wang, Lect. Notes Comput. Sci., vol.3783, pp.72–83, Springer, 2005.
[8] K. Miyazaki, G. Hanaoka, and H. Imai, "Digitally signed document sanitizing scheme based on bilinear maps," ASIACCS, pp.343–354, 2006.
[9] T. Izu, N. Kunihiro, K. Ohta, and M. Takenaka, "On the security of the sanitizable signature schemes from bilinear maps," Trans. IPSJ, vol.47, no.8, pp.2409–2416, 2006.
[10] T. Izu, N. Kunihiro, K. Ohta, M. Takenaka, and T. Yoshioka, "A sanitizable signature scheme with aggregation," ISPEC, ed. E. Dawson and D.S. Wong, Lect. Notes Comput. Sci., vol.4464, pp.51–64, Springer, 2007.
[11] S. Canard, F. Laguillaumie, and M. Milhau, "Trapdoor sanitizable signatures and their application to content protection," ACNS, ed. S.M. Bellovin, R. Gennaro, A.D. Keromytis, and M. Yung, Lect. Notes Comput. Sci., vol.5037, pp.258–276, 2008.
[12] R. Canetti, O. Goldreich, and S. Halevi, "The random oracle methodology, revisited (preliminary version)," STOC, pp.209–218, 1998.
[13] M. Blum and S. Micali, "How to generate cryptographically strong sequences of pseudo-random bits," SIAM J. Comput., vol.13, no.4, pp.850–864, 1984.
[14] A.C. Yao, "Theory and application of trapdoor functions," SFCS'82: Proc. 23rd Annual Symposium on Foundations of Computer Science, pp.80–91, Washington DC, USA, 1982.
[15] R. Steinfeld, L. Bull, and Y. Zheng, "Content extraction signatures," ICISC, pp.285–304, 2001.
[16] M. Klonowski and A. Lauks, "Extended sanitizable signatures," ICISC, pp.343–355, 2006.
[17] G. Ateniese and B. de Medeiros, "Identity-based chameleon hash and applications," Financial Cryptography, pp.164–180, 2004.
[18] K. Miyazaki, G. Hanaoka, and H. Imai, "Bit-by-bit sequence sanitizable digitally signed document sanitizing scheme," Symposium on Cryptography and Information Security — SCIS, 2006.
[19] P. Yang, K. Miyazaki, G. Hanaoka, K. Matsuura, and H. Imai, "Security notions and proof of a bitwise sanitizable signature scheme from any one-way permutation," Symposium on Cryptography and Information Security — SCIS, 2008.
[20] O. Goldreich and L.A. Levin, "A hard-core predicate for all one-way functions," STOC, pp.25–32, 1989.
[21] B. Waters, "Efficient identity-based encryption without random oracles," EUROCRYPT, Lect. Notes Comput. Sci., vol.3494, pp.114–127, 2005.
[22] D. Boneh and M.K. Franklin, "Identity-based encryption from the weil pairing," SIAM J. Comput., vol.32, no.3, pp.586–615, 2003.
[23] J.H. Silverman, The Arithmetic of Elliptic Curves, Springer-Verlag, 1986.

Appendix A: Proof of Theorem 2

A.1 *Proof of Type 1*

For Type 1, the adversary is explicitly given the public verification key vk. He can query O_1 and O_2 to gain information. After the query phase, the adversary outputs a signature σ_n^* along with a completely sanitized message $\mathbf{m}_n^*$, where $n = |\mathbf{m}_n^*|$, with restriction that $\mathbf{m}_n^* \notin \{\mathbf{m}_n^i\}$, where elements of $\{\mathbf{m}_n^i\}$ are completely sanitized messages whose corresponding open or partially sanitized messages have been answered to the adversary by O_1. We say the adversary wins UF-1 game if σ_n^* is a valid signature for $\mathbf{m}_n^*$.

The reason why the output message is completely sanitized is that, if the adversary has the ability to output a valid signature σ_k for a partially sanitized message $\mathbf{m}_k$, then by executing the sanitizing algorithms for $n - k$ times, the adversary can also obtain valid σ_n for $\mathbf{m}_n$. Therefore, in UF-1 game, only focusing on completely sanitized message and its signature is sufficient to capture security.

Lemma 4: $\mathbf{Adv}^{\mathsf{UF\text{-}1}}_{\mathcal{SBSS},A}(\lambda)$ is negligible if $\mathcal{S}$ is an *EUF-CMA* secure signature scheme.

Proof The detailed proof is in Appendix B. □

A.2 *Proof of Type 2*

In this scenario, the adversary A is given vk and has access to O_1 and O_2 as same as in strategy 1. After the query phase, A focuses on one signature-message pair $(\sigma_k^*, \mathbf{m}_k^*)$ and tries to rewind one bit of the sanitized message. A outputs $(\sigma_{k-1}^*, \mathbf{m}_{k-1}^*)$ and wins the game if this is a valid pair. The restriction is that $(\sigma_k^*, \mathbf{m}_k^*)$ cannot be straightforwardly computed from any query result.

Lemma 5: $\mathbf{Adv}^{\mathsf{UF\text{-}2}}_{\mathcal{SBSS},F}(\lambda)$ is negligible if f is a secure one-way permutation.

Proof The detailed proof is in Appendix C. □

A.3 *Proof of Type 3*

In this scenario, A observes one message-signature pair $(\sigma_k, \mathbf{m}_k)$ that can be straightforwardly computed from a query. A tries to find a message $\mathbf{m}_k^*$ such that for some bit $[\mathbf{m}_0^*]_i^i \neq [\mathbf{m}_0]_i^i$ but $\mathbf{m}_n^* = \mathbf{m}_n$. Intuitively, A hopes the corresponding hard-core predicate $h_k^* = 1 - h_k$, so that σ_k is available for both $\mathbf{m}_k^*$ and $\mathbf{m}_k$.

Lemma 6: $\mathbf{Adv}^{\mathsf{UF\text{-}3}}_{\mathcal{SBSS},F}(\lambda)$ is zero if f is a one-way permutation.

Proof. It is obvious that since f is a one-way permutation (1-1 length-preserving), no such hard-core predicate of f exists. We have $\mathbf{Adv}^{\mathsf{UF\text{-}3}}_{\mathcal{SBSS},F}(\lambda) = 0$. This ends the proof of Lemma 6. □

Summing up Lemma 4, 5 and 6, we claim that if $\mathcal{S}$ is an EUF-CMA secure signature scheme and f is a one-way permutation, then BASIC scheme is secure in the sense of UF. This ends the proof of Theorem 2. □

Appendix B: Proof of Lemma 4

Towards contradiction, we first assume there exists an adversary A who can successfully break BASIC with non-negligible advantage, then we construct an adversary F who can successfully break $\mathcal{S}$ with non-negligible advantage, by letting F act as an oracle to answer A's all queries.

Setup. A challenger runs the key generation algorithm gen on security parameter λ, passes the verification key vk to F, and keeps the signing key sk to himself. After receiving vk, F passes it to A.
Query. Since A is an adversary against UF, A will issue two sorts of queries. F maintains a list γ-list of tuple $\langle \mathbf{m}^i, s_0^i \rangle$, and this γ-list is initially empty. F answers the queries as follows,

O_1**-queries.** A issues (t_k, k) to oracle O_1. To respond these queries, F will

1. randomly pick a message $\mathbf{m}_0^i$ from the message space, such that $[\mathbf{m}_0^i]_{n-k} = t_k$,
2. pick a random seed $s_0^i \in \{0,1\}^*$, and compute $c^i \leftarrow \mathbf{m}_0^i \oplus G(s_0^i, n) \| f^{(n)}(s_0^i)$,
3. query the signing oracle assigned to himself to obtain signature sign on ciphertext c^i, and compose σ_0^i,
4. sanitize $\langle \sigma_0^i, \mathbf{m}_0^i \rangle$ as introduced in previous subsection, for k times,
5. add $\langle \mathbf{m}_0^i, s_0^i \rangle$ to γ-list and return $(\sigma_k^i, \mathbf{m}_k^i)$ to A.

O_2**-queries.** A issues $(\sigma_k, \mathbf{m}_k)$ to oracle O_2. As described previously, to respond these queries, F will investigate γ-list. If γ-list reveals the queried message has never been signed, O_2 will clarify the queried pair is invalid and reject the query. Or else F can easily use the information in γ-list to rewind one bit of $\mathbf{m}_k$ and send $(\sigma_{k-1}, \mathbf{m}_{k-1})$ to A.

Forgery. A outputs a forgery $(\sigma_n^*, \mathbf{m}_n^*)$ and wins the UF-1 game if the forgery is valid. F parses σ_n^* as $\sigma \| s_n$, and outputs $(\sigma, \mathbf{m}_n^*)$ as his own output.

B.1 Reduction Evaluation

Obviously, $\mathbf{Adv}^{\text{EUF-CMA}}_{\mathcal{S},F}(\lambda) \geq \mathbf{Adv}^{\text{UF-1}}_{\mathcal{SBSS},A}(\lambda)$. Assuming A's running time is t_A, it is simple to evaluate F's running time t_1 as $t_1 \leq t_A + q_1 \cdot n \cdot t_f + q_2 \cdot n \cdot t_f$, where q_1 (respectively q_2) is the number of queries to O_1 (respectively O_2) and t_f is the running time to compute the one-way permutation f. This ends the proof of Lemma 4. □

Appendix C: Proof of Lemma 5

To successfully rewind $(\sigma_k^*, \mathbf{m}_k^*)$, A has to compute the preimage s_{k-1} of s_k to construct σ_{k-1}^*, and then use the hard-core predicate h_k of f on input s_{k-1} to construct $\mathbf{m}_{k-1}^*$. This means A needs the ability to break the one-wayness of f. Thus, we have $\mathbf{Adv}^{\text{UF-2}}_{\mathcal{SBSS},F}(\lambda) \geq \epsilon_{\text{ow}}$, and $t_2 \leq t_{\text{ow}}$, where ϵ_{ow} and t_{ow} denote the probability and time to break one-wayness. This ends the proof of Lemma 5. □

Appendix D: Proof of Theorem 3

Toward contradiction, we first assume that there exists an adversary A who can break BASIC in the DC game, then we construct a distinguisher D who can distinguish the output of BMY-PRNG from a uniform string by using A as a subroutine, which contradicts Lemma 1.

Setup. The challenger runs BMY-PRNG on input s_0^* and passes the output sequence $(h_1 \| h_2 \| \cdots \| h_n)$, the seed s_n^*, and a uniform sequence in $\{0,1\}^n$ to D, where h_i is a hard-core predicate of f on input s_{i-1}^*, and $s_n^* \leftarrow f^{(n)}(s_0^*)$. D tries to tell the pseudorandom sequence from the truly random sequence. Thus, the advantage of D is evaluated as,

$$\begin{aligned}\mathbf{Adv}(D) &= P_1 - P_2 \\ &= \Pr[z \leftarrow (h_1 \| h_2 \| \cdots \| h_n) : D(z, s_n^*) = 1] \\ &\quad - \Pr[z \leftarrow \{0,1\}^n : D(z, s_n^*) = 1]\end{aligned}$$

Here D outputs 1 when he guesses the input sequence is pseudorandom. Notice D will use s_n^* in the challenge phase.

We then construct such D by using A as follows. D runs the key generation algorithm gen on security parameter λ, passes the verification key vk to A, and keeps the signing key sk to himself.
Query phase 1. Since A is an adversary against DC, A will issue two sorts of queries. D maintains a list γ-list of tuple $\langle \mathbf{m}^i, s_0^i \rangle$, and this γ-list is initially empty. D answers the queries as follows,

O_1**-queries.** A issues (t_k, k) to oracle O_1. To respond these queries, D will

1. randomly pick a message $\mathbf{m}_0^i$ from the message space, such that $[\mathbf{m}_0^i]_{n-k} = t_k$,
2. pick a random seed $s_0^i \in \{0,1\}^*$, and compute $c^i \leftarrow \mathbf{m}_0^i \oplus G(s_0^i, n) \| f^{(n)}(s_0^i)$,
3. use sk to sign on ciphertext c^i, and compose σ_0^i,
4. sanitize $\langle \sigma_0^i, \mathbf{m}_0^i \rangle$ as introduced in previous subsection, for k times,
5. add $\langle \mathbf{m}_0^i, s_0^i \rangle$ to γ-list and return $(\sigma_k^i, \mathbf{m}_k^i)$ to A.

O_2**-queries.** A issues $(\sigma_k, \mathbf{m}_k)$ to oracle O_2. As described previously, to respond these queries, D will investigate γ-list. If γ-list reveals the queried message has never been signed, O_2 will clarify the queried pair is invalid

and reject the query. Or else D can easily use the information in γ-list to rewind one bit of $\mathbf{m}_k$ and send $(\sigma_{k-1}, \mathbf{m}_{k-1})$ to A.

Challenge. At the challenge phase, A outputs $(\mathbf{m}_0^0, \mathbf{m}_0^1, k, st)$. Without loss of generality, we assume the k-th bits of $\mathbf{m}_0^0$ and $\mathbf{m}_0^1$ are different, which means A is forbidden to issue rewinding queries on the challenge constructed by D below.

1. D picks a random bit $b \leftarrow \{0,1\}$, and assigns $\mathbf{m}_0^* \leftarrow \mathbf{m}_0^b$.
2. For every bit of the left-most k bits in $\mathbf{m}_0^*$, D computes the ciphertext, such that $[c^*]_i^i \leftarrow [\mathbf{m}_0^*]_i^i \oplus [z]_{n-k+1}^{n-k+1}$, and sets $\mathbf{m}_k^* \leftarrow [c^*]^k \| [\mathbf{m}_0^*]_{n-k}$, where $0 < i \leq k$, and z is either a pseudorandom sequence or a truly random sequence.
3. D runs f for $n-k$ times on s_n^*. The final seed s_{2n-k}^* will be used as commitment. D uses the $n-k$ output hard-core predicate $\{h_{n+1}, \cdots, h_{2n-k}\}$ to encrypt the right-most $n-k$ bits in $\mathbf{m}_0^*$, such that $[c^*]_i^i \leftarrow [\mathbf{m}_0^*]_i^i \oplus h_{n-k+i}$, where $k < i \leq n$.
4. D uses the signing key sk to sign the message, such that $\sigma_k^* \leftarrow \mathsf{sig}(sk, c^* \| s_{2n-k}^*) \| s_n^*$. D passes $(\sigma_k^*, \mathbf{m}_k^*)$ as the challenge to the adversary A.

Query phase 2. A issues O_1-queries and O_2-queries as in query phase 1, with the restriction that he cannot issue $(\sigma_k^*, \mathbf{m}_k^*)$ to O_2. D answers these queries with the same essence as in phase 1.

Guess. A tries to guess which message of $\mathbf{m}_0^0$ and $\mathbf{m}_0^1$ has been signed and sanitized, i.e., A outputs a bit b' and wins the DC game if $b' = b$. If $b' = b$, then D outputs 1, which means the sequence z is pseudorandom; otherwise D outputs 0, which means z is truly random.

D.1 Reduction Evaluation

Since $\mathbf{Adv}(D) = P_1 - P_2$, we evaluate probabilities individually. To evaluate P_1, we notice that in this case the left-most k bits of sanitized message $[\mathbf{m}_k^*]^k$ is encrypted by a part of BMY-PRNG's output, and D tells z is pseudorandom only when A guesses correctly. Thus

$$\begin{aligned} P_1 &\geq \frac{1}{2}\Pr[\mathbf{Exp}^{\mathsf{DC\text{-}1}}_{\mathcal{SBSS},A}(\lambda) = 1] \\ &\quad + \frac{1}{2}\Pr[\mathbf{Exp}^{\mathsf{DC\text{-}0}}_{\mathcal{SBSS},A}(\lambda) = 0] \\ &= \frac{1}{2} + \frac{1}{2}(\Pr[\mathbf{Exp}^{\mathsf{DC\text{-}1}}_{\mathcal{SBSS},A}(\lambda) = 1] \\ &\quad - \Pr[\mathbf{Exp}^{\mathsf{DC\text{-}0}}_{\mathcal{SBSS},A}(\lambda) = 1]) \\ &= \frac{1}{2} + \frac{1}{2} \cdot \mathbf{Adv}^{\mathsf{DC}}_{\mathcal{SBSS},A}(\lambda) \end{aligned}$$

where ϵ is the advantage of A. To evaluate P_2, we notice that in this case $[\mathbf{m}_k^*]^k$ is encrypted by truly random bit, and D outputs 1 only when A looses DC game. Because A can gain no knowledge by observing $[\mathbf{m}_k^*]^k$, A looses game at the probability of 1/2 constantly. Thus, we have $P_2 = 1/2$.

Summing up, D's advantage is $\mathbf{Adv}(D) = P_1 - P_2 \geq 1/2 \cdot \mathbf{Adv}^{\mathsf{DC}}_{\mathcal{SBSS},A}(\lambda)$. Assuming A's running time is t_A, it is simple to evaluate D's running time t_D as $t_D \leq t_A + q_1 \cdot (t_{\mathsf{sig}} + n \cdot t_f) + q_2 \cdot n \cdot t_f$, where q_1 (respectively q_2) is the number of queries to O_1 (respectively O_2), t_{sig} is the running time of the key generation algorithm and t_f is the running time to compute the one-way permutation f. This ends the proof of Theorem 3. □

Appendix E: The Construction of One-way Permutation on Elliptic Curve [5]

Here we explain the construction of a one-way permutation using two twist elliptic curves [5], which is used in Sect. 4. Set two elliptic curves E_1 and E_2 over a finite field $\mathbb{F}_p$ (a prime $p > 5$) with j-invariant $\neq 0$ and 1728 to twists of each other. Two twists are given as follows.

$$\begin{aligned} E_1 &: y^2 = x^3 + ax + b \\ E_2 &: y^2 = x^3 + au^2x + bu^3, \end{aligned}$$

where $a, b, u \in \mathbb{F}_p$ with $4a^3 + 27b^2 \neq 0$ and u is a quadratic nonresidue in $\mathbb{F}_p$. In order to make a secure one way permutation, we assume that both $\#E_1(\mathbb{F}_p)$ and $\#E_2(\mathbb{F}_p)$ are prime, which are set to n_1 and n_2. Then, $n_1 + n_2 = 2p + 2$ [23] holds from the relation between two twist elliptic curves. Let $G_1 \in E_1(\mathbb{F}_p)$ be an element with order n_1 and $G_2 \in E_2(\mathbb{F}_p)$ be an element with order n_2. The i-th multiple of an element $G = (x_G, y_G)$ is denoted by iG, where x_G denotes the x-coordinate of G and y_G denotes the y-coordinate of G.

Based on the above elliptic curve parameters, the one-way permutation $f : [0, 2p+1] \rightarrow [0, 2p+1]$ is constructed by using two maps $l_1 : E_1(\mathbb{F}_p) \rightarrow [0, 2p+1]$ and $l_2 : E_2(\mathbb{F}_p) \rightarrow [0, 2p+1]$ as follows.

$$f(i) = \begin{cases} l_1(iG_1) & \text{if } i \in [0, n_1 - 1] \\ l_2(iG_2) & \text{if } i \in [n_1, 2p+1] \end{cases}$$

$$l_1(G) = \begin{cases} ux_G \bmod p & \text{if } 0 \leq y_G \leq \frac{p-1}{2}; \\ (ux_G \bmod p) + p + 1 & \text{if } \frac{p+1}{2} \leq y_G \leq p-1; \\ p & \text{if } G = O\,. \end{cases}$$

$$l_2(G) = \begin{cases} x_G \bmod p & \text{if } 0 \leq y_G \leq \frac{p-1}{2}; \\ (x_G \bmod p) + p + 1 & \text{if } \frac{p+1}{2} \leq y_G \leq p-1; \\ 2p+1 & \text{if } G = O. \end{cases}$$

The one-wayness of f is based on the intractability of solving ECDLP on E_1 based on G_1 or E_2 based on G_2.

Appendix F: Additional Note on Unforgeability

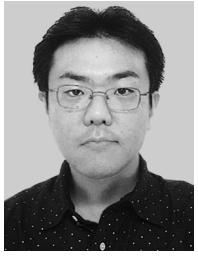
Compared to the previous bitwise signature schemes [3], [4], as shown in Sect. 4, the unforgeability in this paper includes a wider scope. In the previous works, the adversary (forger) has to specify all bits of the message sent to the signer or the sanitizer before hand. In this paper, the next additional scenario is also included. First, the adversary asks the signer to sign a message containing an *unspecified bit sequence*. The signer chooses random bits to substitute the unspecified bit

sequence, then sanitizes the bit sequence, before signing the entire message. The point is that we allow the signer not to reveal the random bits it has chosen by letting it sanitize the bits. Thus, the adversary never sees the random bits chosen by the signer. In fact, by this additional scenario, the adversary (forger) considered in this paper has additional freedom on producing the valid forgery compared to the one in previous works. For an illustration, consider the following example.

Illustration

Let there be only one query allowed to the signing oracle and w.l.o.g., let the length of any valid message be 4 bits. First let us consider the case with the additional scenario. Let the adversary ask the signer to sign "*#101*". Here the signer is free to specify a bit '*0*' or '*1*' for '*#*' and then sanitized it. The adversary then will receive the signature of "$\star$*101*", where '$\star$' denotes a sanitized bit. Notice that the adversary retrieves no information about the bit chosen by the signer which has been sanitized into '$\star$'. Therefore, it becomes natural for us to allow the adversary to output the signature of "*0101*" or "*1101*" as the valid forgery. On the other hand, in the case where all bits of the message have to be specified, when the adversary asks the signer to sign "*1101*" and sanitize the first bit, only the forgery of "*0101*" is the valid forgery.

Of course, if in this paper we put a requirement that the signer always shows publicly the random bits which it chooses to substitute the unspecified bit sequence before it sanitize them, then the additional freedom in forgery mentioned above is disappeared. Note that in this paper we do not put such a requirement.



Goichiro Hanaoka received his bachelors degree in Electronic engineering from the University of Tokyo in 1997, and received his masters and Ph.D. degree in Information and communication engineering from the University of Tokyo in 1999 and 2002, respectively. From 2002 to 2005 he was a Research Fellow of Japan Society for the Promotion of Science (JSPS). Since 2005 he has been with the National Institute of Advanced Industrial Science and Technology, Japan. He received the Best Paper Award of IEICE Trans. in 2008, the Wilkes Award from the British Computer Society in 2007, SCIS Paper Prize from IEICE in 2006, TELECOM System Technology Award in 2004, The 20th Anniversary Prize of ISEC SCIS in 2003, SITA Paper Prize from SITA in 2000.

Shoichi Hirose received the B.E., M.E. and D.E. degrees in information science from Kyoto University, Kyoto, Japan, in 1988, 1990 and 1995, respectively. From 1990 to 1998, he was a research associate at Faculty of Engineering, Kyoto University. From 1998 to 2005, he was a lecturer at Graduate School of Informatics, Kyoto University. From 2005, he is an associate professor at Faculty of Engineering, University of Fukui. His current interests include cryptography and information security. He received Young Engineer Award from IEICE in 1997. He is a member of IACR, ACM, IEEE and IPSJ.

Atsuko Miyaji received the B.Sc., the M.Sc., and the Dr.Sci. degrees in mathematics from Osaka University, Osaka, Japan in 1988, 1990, and 1997 respectively. She joined Panasonic Co., LTD from 1990 to 1998 and engaged in research and development for secure communication. She was an associate professor at the Japan Advanced Institute of Science and Technology (JAIST) in 1998. She has joined the computer science department of the University of California, Davis since 2002. She has been a professor at the Japan Advanced Institute of Science and Technology (JAIST) since 2007 and the director of Library of JAIST since 2008. Her research interests include the application of number theory into cryptography and information security. She received the IPSJ Sakai Special Researcher Award in 2002, the Standardization Contribution Award in 2003, Engineering Sciences Society: Certificate of Appreciation in 2005, the AWARD for the contribution to CULTURE of SECURITY in 2007, IPSJ/ITSCJ Project Editor Award in 2007, 2008, 2009, and 2010, the Director-General of Industrial Science and Technology Policy and Environment Bureau Award in 2007, Editorial Committee of Engineering Sciences Society: Certificate of Appreciation in 2007, and DoCoMo Mobile Science Awards in 2008. She is a member of the International Association for Cryptologic Research, the Information Processing Society of Japan, and the Mathematical Society of Japan.

Kunihiko Miyazaki received B.S. and M.S. degrees in Mathematical Sciences and Ph.D. degree in Information Sciences from the University of Tokyo in 1996, 1998 and 2006, respectively. He has been engaged in research on information security and cryptography at Systems Development Laboratory, Hitachi Ltd. since 1998. He is a member of IPSJ.

Bagus Santoso received his B.E., M.E., and Dr.E. degrees in information and communication engineering from the University of Electro-Communications in 2003, 2005, and 2009 respectively. He is currently with the National Institute of Advanced Industrial Science and Technology, Japan. His current research involves the areas of cryptography and computational number theory. He received the SCIS Paper Prize from IEICE in 2007. He is a member of the International Association for Cryptologic Research.

Peng Yang received his B.E. degree in 2003 from Beihang University (a.k.a., Beijing University of Aeronautics and Astronautics), and received his M.E., and Ph.D. degrees in information science and technology from the University of Tokyo in 2006 and 2009. He was a grantee of the Japanese Government (Monbukagakusho) Scholarship. From 2009, he is a researcher at the University of Tokyo. He is a member of IPSJ.