Short Redactable Signatures Using Random Trees[⋆]

Ee-Chien Chang Chee Liang Lim Jia Xu

School of Computing
National University of Singapore

Abstract. A redactable signature scheme for a string of objects supports verification even if multiple substrings are removed from the original string. It is important that the redacted string and its signature do not reveal anything about the content of the removed substrings. Existing schemes completely or partially leak a piece of information: the lengths of the removed substrings. Such length information could be crucial in many applications, especially when the removed substring has low entropy. We propose a scheme that can hide the length. Our scheme consists of two components. The first component $\mathcal{H}$, which is a "collision resistant" hash, maps a string to an unordered set, whereby existing schemes on unordered sets can then be applied. However, a sequence of random numbers has to be explicitly stored and thus it produces a large signature of size at least (mk)-bits where m is the number of objects and k is the size of a key sufficiently large for cryptographic operations. The second component uses RGGM tree, a variant of GGM tree, to generate the pseudo random numbers from a short seed, expected to be of size $O(k + tk \log m)$ where t is the number of removed substrings. Unlike GGM tree, the structure of the proposed RGGM tree is random. By an intriguing statistical property of the random tree, the redacted tree does not reveal the lengths of the substrings removed. The hash function $\mathcal{H}$ and the RGGM tree can be of independent interests.

Key words: Redactable Signature Scheme, Random Tree, Privacy

1 Introduction

We are interested in a signature scheme for strings of objects whereby their authenticity can be verified even if some substrings have been removed, that is, the strings are redacted. Let $\mathbf{x} = x_1 x_2 \dots x_m$ be a string, for example a text document where each object can be a character or a word, or an audio file where each object is a sample. The string $\mathbf{x}$ is signed by the authority and both $\mathbf{x}$ and its signature $\mathbf{s}$ are passed to another party, say Alice. Alice wants to show Bob $\mathbf{x}$ but Bob is not authorized to view certain parts of the string, say $x_2 x_3 x_4$ and x_7. Thus, Alice shows Bob $\widetilde{\mathbf{x}} = x_1 \diamond x_5 x_6 \diamond x_8 \dots x_m$ where each $\diamond$ indicates the

[⋆] Extended version [5] of this paper is available in Cryptology ePrint Archive.

location of a removed substring. On the other hand, Bob may want to verify the authenticity of $\widetilde{\mathbf{x}}$. A redactable signature scheme allows Alice to produce a valid signature $\widetilde{\mathbf{s}}$ for the redacted string $\widetilde{\mathbf{x}}$, even if Alice does not have the authority's secret key. From the new signature $\widetilde{\mathbf{s}}$, Bob can then verify that $\widetilde{\mathbf{x}}$ is indeed a redacted version of a string signed by the authority.

Unlike the usual signature schemes, redactable signature scheme has additional requirement on privacy: information of the removed strings should be hidden. In this paper, we consider the stringent requirement that, Bob could not obtain any information of any removed substring, except the fact that a non-empty substring has been removed at each location $\diamond$. This simple requirement turns out to be difficult to achieve. Existing schemes are unable to completely hide a piece of information: the length of each removed substring. Note that information on length could be crucial if the substring has low entropy. For example, if the substring is either "Approved" or "Not Approved", then its length reveals everything. The redactable signature scheme proposed by Johnson et al. [9] employs a Merkle tree [11] and a GGM tree [7] to generate a short signature. However, it is easy to derive the length from the structures of the redacted Merkle and GGM trees. A straightforward modification by introducing randomness into the tree structure also does not hide the length completely. Schemes by Johnson et al. [9] (set-homomorphic signatures) and Miyazaki et al. [13] are designed for unordered sets and are not applicable for a string. A way to extend their schemes to strings is by assigning a sequence of increasing random numbers to the objects [13]. However, this leads to large signatures since the random numbers have to be explicitly stored, and more importantly, it is insecure since the gaps in the sequence reveal some information about the number of removed objects.

Note that the type of information to be removed varies for different applications. There are applications where the lengths of the removed strings should not be hidden. As noted by Johnson et al. [9], semantic attack could be possible in some scenarios if the length information is hidden. On the other hand, there are also applications where not only the substrings have to be completely purged, the fact that a string has been redacted must be hidden. Our scheme can be modified to cater for the above two scenarios.

In this paper, we propose a scheme that can hide the lengths of the removed substrings. Our scheme incorporates two components: a hash, and a random tree with a hiding property. We first give a scheme $\mathcal{RSS}$ using the first component, and then another scheme $\mathcal{SRSS}$ with both components. The first component hashes a string of objects to an unordered set. For the unordered set, existing redactable schemes [13, 9] on unordered sets can be applied. The scheme $\mathcal{RSS}$ satisfies the requirements on unforgeability and privacy preserving under reasonable cryptographic assumptions. However, it produces a large signature. Essentially, the main portion of the signature is a sequence of random numbers $\langle r_1, r_2, \ldots, r_m \rangle$, where each r_i is associated with the i-th object in the string.

The goal of the second component is to reduce the signature size by generating the r_i's from a small seed t. If a substring is removed, the corresponding random numbers have to be removed accordingly. Thus, a straightforward method of generating the random numbers iteratively starting from the seed violates privacy, since the seed t reveals all the random numbers.

We employ a variant of GGM binary tree to generate the r_i's in a top-down manner, where the r_i's are at the leaves, and the seed t is at the root. Unlike the GGM tree which is balanced, we use a random binary tree where the *structure* of the binary tree is random. After a substring is removed, the associated leaves and all their ancestors are to be removed, resulting in a collection of subtrees (Figure 1). The roots of the subtrees collectively form the new seed $\widetilde{t}$ for the redacted r_i's. Note that from the structures of the subtrees, an adversary might still derive some information of the length of a removed substring. Our main observation is that, by choosing an appropriate tree generation algorithm, the structure of the subtrees reveals nothing about the size of the original tree. Consider a game between- Alice and Bob. Suppose Alice randomly picks a binary tree and it is equal likely that the tree contains 1000 leaves or 9 leaves. Now Alice redacts the tree by removing one substring and only 8 leaves are left. From the structure of the remaining subtrees (for example Figure 1(b)), Bob tries to guess the size of the original tree. Now, if Alice employs a tree generation algorithm with the hiding property, Bob cannot succeed with probability more than 0.5. This hiding property is rather counter-intuitive. Since the size of the tree is involved in the tree generation and thus intuitively the information about the size of the tree is spread throughout the tree. It is quite surprising that the global information on size can be completely removed by deleting some nodes.

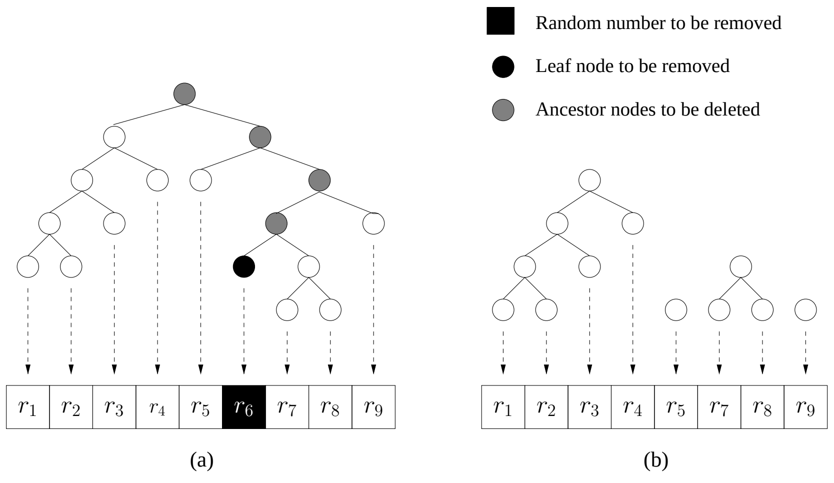
■ Random number to be removed
● Leaf node to be removed
● Ancestor nodes to be deleted
r_1 r_2 r_3 r_4 r_5 r_6 r_7 r_8 r_9
(a)
r_1 r_2 r_3 r_4 r_5 r_7 r_8 r_9
(b)

Fig. 1. Redacting the tree in (a) by removing r_6, gives rise to the redacted tree (b).

Contribution and Organization.

1. We propose a “collision resistant” hash $\mathcal{H}$ that maps strings to unordered

sets. From $\mathcal{H}$ we obtain $\mathcal{RSS}$, a redactable signature scheme for strings. Unlike previously known methods, $\mathcal{RSS}$ is able to hide the lengths of the removed substrings. We show that $\mathcal{RSS}$ is secure against chosen message attack (Theorem 2) and privacy preserving (Theorem 3) under assumptions weaker than the random oracle assumption. However, the signature size is large. It consists of $km + kt + \kappa$ bits, where κ is the size of the signature produced by a known redactable signature scheme for unordered sets, m is the number of objects in the redacted string, t is the number of substrings removed, and k is a security parameter (e.g. $k = 1024$).

2. We observe a hiding property of a random tree (Theorem 4). Based on the observation, we propose RGGM, a pseudo random number generator which can be viewed as a randomized version of GGM [7]. If multiple substrings of pseudo random numbers are to be removed, we can efficiently find a new seed that generates the retained numbers, and yet it is computationally difficult to derive the content and length of each removed substring from the new seed, except the locations of the removed substrings.

3. We propose $\mathcal{SRSS}$ by incorporating RGGM into $\mathcal{RSS}$. The expected size of the signature is in $\kappa + O(k + kt \log m)$. $\mathcal{SRSS}$ is secure against chosen message attack (Corollary 5) and privacy preserving (Corollary 6).

2 Related Work

Johnson et al. [9] introduced redactable signature schemes which enable verification of a redacted signed document. Signature scheme with similar property has also been proposed for XML documents [15], where the redaction operation is to remove XML nodes. Redactable signatures are examples of homomorphic signatures which are introduced by Rivest in his talks on "Two New Signature Schemes" [14] and formalized by Johnson et al. [9]. Micali et al. [12] gave a transitive signature scheme as the first construction of homomorphic signatures. They also asked for other possible "signature algebras". The notions on homomorphic signatures can be traced back to incremental cryptography, introduced by Bellare, Goldreich and Goldwasser [3, 4]. Recently, Ateniese et al. [2] introduced sanitizable signature scheme [10, 8, 16, 13] allowing a semi-trusted censor modifies the signed documents in a limited and controlled way.

The redactable signature scheme on strings is closely related to directed transitive signature scheme [12, 17]. It is possible to convert a directed transitive signature scheme to a redactable signature scheme on strings. However, existing directed transitive signature schemes do not provide privacy in the sense that the resulting signatures reveal some information about the removed substrings.

There are extensive works on random tree. Aldous [1] considered random trees satisfying this consistency property: removing a random leaf from $\mathcal{R}(k)$ gives $\mathcal{R}(k-1)$, where $\mathcal{R}(k)$ is a random tree with k leaves. Thus, given a tree with k leaves, it can be originated from a tree with $k+t$ leaves, and then with t

randomly chosen leaves removed, for any t. This consistency property is similar to the hiding property we seek. Unfortunately, it cannot be applied in our problem, since the leaves to be removed are not randomly chosen.

3 Formulation and Background

Johnson et al.[9] gave definitions on homomorphic signature schemes and their security for binary operators. The next two definitions (Definition 1 & 2) are based on the notations by Johnson et al.[9].

A string is a sequence of objects from an object space (or alphabet) $\mathbb{O}$. For example, $\mathbb{O}$ can be the set of ASCII characters, collection of words, or audio samples, etc. We assume that the first and last object in $\mathbf{x}$ can not be removed. This assumption can be easily met by putting a special symbol at the front and back of the string. After a few substrings are removed from $\mathbf{x}$, the string $\mathbf{x}$ may break into substrings, say $\mathbf{x}_1, \mathbf{x}_2, \ldots, \mathbf{x}_u$. The redacted string $(\widetilde{\mathbf{x}}, \mathbf{e})$, which we call *annotated string*[1], is represented by the string $\widetilde{\mathbf{x}} = \mathbf{x}_1 \| \mathbf{x}_2 \| \ldots \| \mathbf{x}_u$ and an *annotation* $\mathbf{e} = \langle m, b_1, b_2, \ldots, b_v \rangle$ where $\|$ denotes concatenation, b_i's is a strictly increasing sequence indicating the locations of the removed substrings, m is the number of objects in $\widetilde{\mathbf{x}}$, and $v \in \{u-1, u, u+1\}$. For each i, b_i indicates that a non-empty substring has been removed in between the b_i-th and $(1+b_i)$-th locations. If $b_1 = 0$ or $b_v = m$, this indicates that a non-empty substring has been removed at the beginning or end of the string respectively. For example, $(abcda, \langle 5, 0, 3 \rangle)$ is a redacted string of the original $xxxabcyyyda$. For convenient, we sometimes write a sequence of objects as $\langle x_1, x_2, x_3, \ldots, x_m \rangle$ or as a string $x_1 x_2 x_3 \ldots x_m$.

Let us define a binary relation $\succ$ between annotated strings. Given two annotated strings $\mathcal{X}_1 = (\mathbf{x}_1, \mathbf{e}_1)$ and $\mathcal{X}_2 = (\mathbf{x}_2, \mathbf{e}_2)$, we say $\mathcal{X}_1 \succ \mathcal{X}_2$, if either $\mathbf{x}_2$ can be obtained from $\mathbf{x}_1$ by removing a non-empty substring in $\mathbf{x}_1$, and the $\mathbf{e}_2$ is updated from $\mathbf{e}_1$ accordingly, or there is a $\mathcal{X}$ s.t. $\mathcal{X}_1 \succ \mathcal{X}$ and $\mathcal{X} \succ \mathcal{X}_2$.

Definition 1 (Redactable Signature Scheme [9]) *A redactable signature scheme with respect to binary relation $\vdash$, is a tuple of probabilistic polynomial time algorithms* $(\texttt{KGen}, \texttt{Sign}, \texttt{Verify}, \texttt{Redact})$, *such that*

1. *for any message x, $\sigma = \texttt{Sign}_{\mathcal{SK}}(x) \Rightarrow \texttt{Verify}_{\mathcal{PK}}(x, \sigma) = \texttt{TRUE}$;*
2. *for any messages x and y, such that $x \vdash y$,*

$$\texttt{Verify}_{\mathcal{PK}}(x, \sigma) = \texttt{TRUE} \wedge \sigma' = \texttt{Redact}_{\mathcal{PK}}(x, \sigma, y) \Rightarrow \texttt{Verify}_{\mathcal{PK}}(y, \sigma') = \texttt{TRUE},$$

where $(\mathcal{PK}, \mathcal{SK}) \leftarrow \texttt{KGen}(1^k)$ and k is the security parameter.

Both Johnson et al.[9] and Miyazaki et al.[13] presented a redactable signature scheme w.r.t superset relation. Johnson et al.[9] also gave security definition for homomorphic signature schemes. We adapt their definition for redactable signature scheme. Let $\vdash$ denote a binary relation. For any set S, let $span_{\vdash}(S)$ denote the set $\{x : \exists y \in S, \text{ s.t. } y \vdash x\}$.

[1] A string with an annotation which specifies the locations of redactions.

Definition 2 (Unforgeability of Redactable Signature Scheme [9]) *A redactable signature scheme* $\langle \texttt{KGen}, \texttt{Sign}, \texttt{Verify}, \texttt{Redact}\rangle$ *is* (t, q, ϵ)*-unforgeable against existential forgeries with respect to* $\vdash$ *under adaptive chosen message attack, if any adversary* $\mathcal{A}$ *that makes at most q chosen-message queries adaptively and runs in time at most t, has advantage* $Adv\mathcal{A} \leq \epsilon$*. The advantage of an adversary* $\mathcal{A}$ *is defined as the probability that, after queries on* ℓ $(\ell \leq q)$ *messages* $x_1, x_2, \ldots, x_\ell$*,* $\mathcal{A}$ *outputs a valid signature* σ *for some message* $x \notin span_\vdash(\{x_1, x_2, \ldots, x_\ell\})$*. Formally,*

$$Adv\mathcal{A} = \mathsf{Pr}\left[\begin{array}{c}(\mathcal{PK}, \mathcal{SK}) \leftarrow \texttt{KGen}(1^k);\ \mathcal{A}^{\texttt{Sign}_{\mathcal{SK}}} = (x, \sigma);\\ \texttt{Verify}_{\mathcal{PK}}(x, \sigma) = \texttt{TRUE} \textit{ and } x \notin span_\vdash(\{x_1, x_2, \ldots, x_\ell\})\end{array}\right],$$

where the probability is taken over the random coins used by KGen, Sign *and* $\mathcal{A}$*.*

Redactable signature schemes have an additional security requirement on privacy [2]: the adversary should not be able to derive any information about the removed substrings from a redacted string and its signature.

Definition 3 (Privacy Preserving) *A redactable signature scheme* $\langle \texttt{KGen}, \texttt{Sign}, \texttt{Verify}, \texttt{Redact}\rangle$ *is privacy preserving if, given the public key* $\mathcal{PK}$ *and any annotated strings* $\mathcal{X}_1, \mathcal{X}_2, \mathcal{X}$*, such that* $\mathcal{X}_1 \succ \mathcal{X}$ *and* $\mathcal{X}_2 \succ \mathcal{X}$*, the following distributions* $\mathcal{S}_1$ *and* $\mathcal{S}_2$ *are computationally indistinguishable:*

$$\mathcal{S}_1 = \{\sigma : \sigma = \texttt{Redact}_{\mathcal{PK}}(\mathcal{X}_1, \texttt{Sign}_{\mathcal{SK}}(\mathcal{X}_1; \mathbf{r}_1), \mathcal{X}; \mathbf{r}_2)\},$$
$$\mathcal{S}_2 = \{\sigma : \sigma = \texttt{Redact}_{\mathcal{PK}}(\mathcal{X}_2, \texttt{Sign}_{\mathcal{SK}}(\mathcal{X}_2; \mathbf{r}_1), \mathcal{X}; \mathbf{r}_2)\},$$

where $\mathbf{r}_1$ *and* $\mathbf{r}_2$ *are random bits used by* Sign *and* Redact *respectively, and public/private key* $(\mathcal{PK}, \mathcal{SK})$ *is generated by* KGen*.*

4 $\mathcal{RSS}$: Redactable Signature Scheme for Strings

We propose $\mathcal{RSS}$, a redactable signature scheme for strings that is able to hide the lengths of the removed substrings. Our approach is as follows: we first propose a hash function $\mathcal{H}$ that maps an annotated string $\mathcal{X}$ and an auxiliary input $\mathbf{y}$ to an unordered set. This hash is "collision resistant" and satisfies some properties on substring removal. Using $\mathcal{H}$ and some known redactable signature schemes for unordered sets, we have a redactable signature scheme for strings.

4.1 Hashing strings to unordered sets

Let $\mathcal{H}$ be a hash function that maps an annotated string $\mathcal{X}$ and an auxiliary input $\mathbf{y}$ to a (unordered) set of elements from some universe. The auxiliary could be a sequence of numbers from a finite ring, and is not of particular interest right now. In our construction (Table 1), $\mathcal{H}$ maps the input to a set of 3-tuples in $\mathbb{Z}_n \times \mathbb{Z}_n \times \mathbb{Z}_n$, where n is some chosen parameter.

Definition 4 (Collision Resistant) *$\mathcal{H}$ is (t, ϵ)-collision-resistant if, for any algorithm $\mathcal{A}$ with running time at most t,*

$$\mathsf{Pr}\left[\mathcal{X}_1 \not\succ \mathcal{X}_2 \ \wedge \ \mathcal{H}(\mathcal{X}_2, \mathbf{y}_2) \subset \mathcal{H}(\mathcal{X}_1, \mathbf{y}_1)\right] \leq \epsilon,$$

where $(\mathcal{X}_1, \mathcal{X}_2, \mathbf{y}_2)$ is the output of $\mathcal{A}$ on input $\mathbf{y}_1$, and the probability is taken over uniformly randomly chosen $\mathbf{y}_1$ and random bits used by $\mathcal{A}$.

To be used in constructing a secure scheme, besides collision resistance, the hash function $\mathcal{H}$ is also required to be,

1. redactable, that is, given $\mathcal{X}_1$, $\mathcal{X}_2$ and $\mathbf{y}_1$, such that $\mathcal{X}_1 \succ \mathcal{X}_2$, it is easy to find $\mathbf{y}_2$ such that $\mathcal{H}(\mathcal{X}_1, \mathbf{y}_1) \supset \mathcal{H}(\mathcal{X}_2, \mathbf{y}_2)$; and
2. privacy preserving, that is, $\mathcal{H}(\mathcal{X}_2, \mathbf{y}_2)$ must not reveal any information about the removed substring.

The property on privacy preserving is essential and used in the proof of Theorem 3. However, for simplicity, we will not explicitly formulate the requirement here.

4.2 Construction of $\mathcal{H}$

We present a hash function $\mathcal{H}(\cdot, \cdot)$ in Table 1 based on some hash functions h that output odd numbers. In practice, we may use popular cryptographic hash function like SHA-2 as h, but with the least significant bit always set to 1. For security analysis, we choose functions with certain security requirements as stated in Lemma 1.

Let n be a RSA modulus, and $h : \mathbb{Z}_n \rightarrow \mathbb{Z}_n$ be a hash function. Given $\mathbf{x} = x_1 x_2 \ldots x_m$ associated with annotation $\mathbf{e}$, $\mathbf{r} = r_1 r_2 r_3 \ldots r_m$, and $\mathbf{w} = w_1 w_2 w_3 \ldots w_m$, where for each i, $x_i, r_i, w_i \in \mathbb{Z}_n$ (i.e. $\mathbf{x}, \mathbf{r}$ and $\mathbf{w}$ are strings over alphabet $\mathbb{Z}_n$), we define $\mathcal{H}$ as

$$\mathcal{H}((\mathbf{x}, \mathbf{e}), (\mathbf{r}, \mathbf{w})) \triangleq \{t_i : t_i = (x_i,\ r_i,\ (w_i^{\prod_{j=1}^{i} h(r_j)} \mod n)), 1 \leq i \leq m\}.$$

Table 1. Definition of $\mathcal{H}(\cdot, \cdot)$.

Redactable requirement. Note that the hash $\mathcal{H}$ is redactable as mentioned in Section 4.1, that is, given $(\mathbf{x}_1, \mathbf{e}_1), (\mathbf{r}_1, \mathbf{w}_1)$ and $(\mathbf{x}_2, \mathbf{e}_2)$ where $(\mathbf{x}_1, \mathbf{e}_1) \succ (\mathbf{x}_2, \mathbf{e}_2)$, it is easy to find a $(\mathbf{r}_2, \mathbf{w}_2)$ such that

$$\mathbf{H}((\mathbf{x}_1, \mathbf{e}_1), (\mathbf{r}_1, \mathbf{w}_1)) \supset \mathbf{H}((\mathbf{x}_2, \mathbf{e}_2), (\mathbf{r}_2, \mathbf{w}_2)).$$

The design of $\mathcal{H}$ is "inspired" by the following observation. Let us view the sequence $\langle t_1, t_2, \ldots, t_m \rangle$ as the outputs of an iterative hash. We can rewrite t_i's in the form: $t_{i+1} = C(t_i, x_{i+1}, r_{i+1})$, where C is the basic block in the iterative hash. In the event that a substring, say at location $i-1$ and i, is to be removed, both (x_{i-1}, r_{i-1}) and (x_i, r_i) also have to be removed. Yet, we want the iterative hash can still be computed. This can be achieved with the help of the witness w_i's.

Remarks on r_i's. It is crucial that the value of r_i is explicitly represented in t_i for each i (Table 1). If the r_i's are omitted in the design, for instance, by using this alternative definition,

$$\widetilde{\mathcal{H}}((\mathbf{x}, \mathbf{e}), (\mathbf{r}, \mathbf{w})) \triangleq \{\hat{t}_i : \hat{t}_i = (x_i,\ (w_i^{\prod_{j=1}^{i} h(r_j)} \mod n))\},$$

then there would be no linkage between the r_i's and x_i's. Such lack of linkage can be exploited to find collisions.

LEMMA 1 *The hash function $\mathcal{H}$ as defined in Table 1, is $(poly_1(k), \frac{1}{poly_2(k)})$-collision-resistant for any positive polynomials $poly_1(\cdot)$ and $poly_2(\cdot)$, where k is the security parameter, i.e. the bit length of n, assuming that h is division intractable[2] and always outputs odd prime integers, and Strong RSA Problem is hard.*

Essentially, the proof reduces Strong RSA Problem or Division Problem [6] to the problem of finding collisions. Gennaro et al.[6] gave a way to construct a hash function that is division intractable and always outputs odd prime numbers. Thus the conditions of the Lemma 1 can be achieved.

4.3 Construction of $\mathcal{RSS}$

We construct a redactable signature scheme $\mathcal{RSS}$, which consists of four algorithms `KGen`, `Sign`, `Verify`, and `Redact`, for strings with respect to binary relation $\succ$ based on the hash function $\mathcal{H}$ defined in Table 1 and a redactable signature scheme for (unordered) sets with respect to superset relation $\supseteq$.

The signer chooses a RSA modulus n and an element g of large order in $\mathbb{Z}_n^*$. Both n and g are public. Let the object space be $\mathbb{Z}_n$, that is, a string is a sequence of integers from $\mathbb{Z}_n$. Let $h : \mathbb{Z}_n \rightarrow \mathbb{Z}_n$ be a hash which satisfies security requirement stated in Lemma 1. Note that in practice, it may be suffice to employ popular cryptographic hash like SHA-2 (but with the least significant bit of the output always set to 1) as the function h. Let `SSS` = (`keygen`, `sig`, `vrf`, `rec`) be a redactable signature scheme for unordered sets w.r.t superset relation $\supseteq$. The signer also needs to choose the public and secret key pair $(\mathcal{PK}, \mathcal{SK})$ of the underlying signature scheme `SSS`. The details of `KGen`, `Sign`, `Verify`, and `Redact` are presented in Table 2, Table 3, Table 4 and Table 5 respectively.

The final signature of a string $x_1 x_2 \ldots x_m$ consists of m random numbers $r_1, r_2, \ldots, r_m$, the *witnesses* $w_1, w_2, \ldots, w_m$ where $r_i, w_i \in \mathbb{Z}_n$ for each i, and a signature $\mathtt{s}$ constructed by `SSS`.

[2] Division intractability [6] implies collision resistance.

`KGen`. Given security parameter k.

1. Choose a RSA modulus n, and an element g of large order in $\mathbb{Z}_n$.
2. Run key generating algorithm `keygen` on input 1^k to get key $(\mathcal{PK}, \mathcal{SK})$.
3. Output $(n, g, \mathcal{PK})$ as public key and $\mathcal{SK}$ as private key.

Table 2. $\mathcal{RSS}$: `KGen`.

`Sign`. Given $\mathbf{x} = x_1 x_2 \ldots x_m$ and its associated annotation $\mathbf{e} = \langle m \rangle$.

1. Let $w_i = g$ for each i. Choose m distinct random numbers $r_1, r_2, \ldots, r_m$. Let $\mathbf{r} = r_1 r_2 r_3 \ldots r_m$ and $\mathbf{w} = w_1 w_2 w_3 \ldots w_m$. Compute
$$\mathbf{t} = \mathcal{H}((\mathbf{x}, \mathbf{e}), (\mathbf{r}, \mathbf{w})).$$
2. Sign the set $\mathbf{t}$ using `SSS` with the secret key $\mathcal{SK}$ to obtain $\mathbf{s}$:
$$\mathbf{s} = \mathtt{sig}_{\mathcal{SK}}(\mathbf{t}).$$
3. The final signature consists of the random numbers r_i's, witnesses w_i's, and the signature $\mathbf{s}$. That is,
$$(\mathbf{r}, \mathbf{w}, \mathbf{s}) \text{ or } (r_1, r_2, \ldots, r_m; \quad w_1, w_2, \ldots, w_m; \quad \mathbf{s})$$

Table 3. $\mathcal{RSS}$: `Sign`.

Initially, the witness is set to be $w_i = g$ for each i (Step 1 in Table 3). The witness will be modified during redactions. By comparing the neighboring value within the witness $\mathbf{w}$, we can deduce the locations of the removed substrings. Specifically, for any $1 < i \leq m$, $w_{i-1} \neq w_i$ if and only if a non-empty substring has been removed between x_{i-1} and x_i. Recall that the first and last object in the string cannot be removed (Section 3) and thus we do not have to consider cases when $i = 1$ and $i - 1 = m$.

Since the witness $\mathbf{w}$ should be consistent with the annotation $\mathbf{b}$, and the $\mathcal{H}$ is collision-resistant, it can be used to verify the integrity of $\mathbf{b}$, as in the Step 1 of Table 4.

THEOREM 2 *$\mathcal{RSS}$ is $(t, q, \frac{\epsilon_1}{1-\epsilon_2})$-unforgeable against existential forgeries with respect to relation $\succ$, if* `SSS` *is $(t + qt_0, q, \epsilon_1)$-unforgeable against existential forgeries with respect to superset relation $\supseteq$, and $\mathcal{H}$ is $(t + qt_1, \epsilon_2)$-collision-resistant, where t_0 is the running time of $\mathcal{H}$ and t_1 is the time needed by $\mathcal{RSS}$ to sign a document.*

Our construction of $\mathcal{H}$ (Table 1) is collision resistant (Lemma 1). Johnson et al.[9] showed their redactable signature scheme `Sig` (in Section 5 of [9]) is (t, q, ϵ)-unforgeable under reasonable assumptions (see Theorem 1 in [9]), for some proper parameters t, q and ϵ. Miyazaki et al.[13] also showed a similar

`Verify`. Given a string $\mathbf{x} = x_1 x_2 \ldots x_m$ associated with annotation $\mathbf{e}$, its signature $(\mathbf{r}, \mathbf{w}, \mathbf{s})$, the public information n, g, and the public key $\mathcal{PK}$ of `SSS`.

1. If $\mathbf{e}$ and $\mathbf{w}$ are not consistent, output `FALSE`.
2. Compute $\mathbf{t} = \mathcal{H}(\mathbf{x}, (\mathbf{r}, \mathbf{w}))$.
3. $(\mathbf{r}, \mathbf{w}, \mathbf{s})$ is a valid signature of $\mathbf{x}$ under $\mathcal{RSS}$, if and only if $\mathbf{s}$ is a valid signature of $\mathbf{t}$ under `SSS`, i.e.
$$\mathtt{vrf}_{\mathcal{PK}}(\mathbf{t}, \mathbf{s}) = \mathtt{TRUE}.$$

Table 4. $\mathcal{RSS}$: `Verify`.

`Redact`. Given a string $\mathbf{x} = x_1 x_2 \ldots x_m$ associated with annotation $\mathbf{e}$, and its signature $(\mathbf{r}, \mathbf{w}, \mathbf{s})$, where $\mathbf{r} = r_1 r_2 \ldots r_m$, $\mathbf{w} = w_1 w_2 \ldots w_m$, the public information n, g, public key $\mathcal{PK}$ for `SSS`, and (i, j) the location of the string to be removed (that is $x_i x_{i+1} \ldots x_j$ is to be removed).

1. Update $\mathbf{e}$ to obtain new annotation $\hat{\mathbf{e}}$. Compute $u = \prod_{k=i}^{j} h(r_k)$, to update the witnesses in the following way: for each $\ell > j$, update w_ℓ
$$\hat{w}_\ell \leftarrow w_\ell^u \mod n.$$
2. Let $\hat{\mathbf{x}} = x_1 x_2 \ldots x_{i-1} x_{j+1} \ldots x_m$, $\hat{\mathbf{r}} = r_1 r_2 \ldots r_{i-1} r_{j+1} \ldots r_m$ and $\hat{\mathbf{w}} = w_1 w_2 \ldots w_{i-1} \hat{w}_{j+1} \hat{w}_{j+2} \ldots \hat{w}_m$. Compute
$$\hat{\mathbf{t}} = \mathcal{H}((\hat{\mathbf{x}}, \hat{\mathbf{e}}), (\hat{\mathbf{r}}, \hat{\mathbf{w}})).$$
3. Compute
$$\hat{\mathbf{s}} = \mathtt{rec}_{\mathcal{PK}}(\mathbf{t}, \mathbf{s}, \hat{\mathbf{t}})$$
where $\mathbf{t} = \mathcal{H}((\mathbf{x}, \mathbf{e}), (\mathbf{r}, \mathbf{w}))$.
4. Output $(\hat{\mathbf{r}}, \hat{\mathbf{w}}, \hat{\mathbf{s}})$ as the signature of $(\hat{\mathbf{x}}, \hat{\mathbf{e}})$.

Table 5. $\mathcal{RSS}$: `Redact`.

result on the unforgeability of the redactable signature scheme they proposed. Hence, conditions in Theorem 2 can be satisfied.

THEOREM 3 *The redactable signature scheme $\mathcal{RSS}$ is privacy preserving (as defined in Definition 3), assuming that hash function h satisfies the property: the two distributions $X = g^{h(U_1)h(U_2)} \mod n$ and $Y = g^{h(U_1')} \mod n$ are computationally indistinguishable, where n is a RSA modulus, g is an element of large order in $\mathbb{Z}_n^*$ and U_i's and U_j''s are all independent uniform random variables over $\mathbb{Z}_n$.*

Note that the scheme `SSS` does not need to satisfy requirement on privacy, this is because information is already removed before algorithms of `SSS` are applied.

4.4 Efficiency

The size of **s** depends on `SSS`, and let us assume it requires κ bits. The number of distinct w_i's is about the same as the number of redactions occurred. So w_i's can be represented in $t(k + \lceil \log m \rceil)$ bits, where t is the number of substrings removed, and k is the bit length of n. Thus the total number of bits required is at most $k(m + t) + t\lceil \log m \rceil + \kappa$. The dominant term is km, which is the total size of the random numbers r_i's.

Disregarding the time taken by the scheme `SSS`, and the time required to compute the hash $h(\cdot)$, during signing, $O(m)$ of k-bits exponentiation operations are required. During redaction, if ℓ consecutive objects are to be removed between position i and j, and t' number of redactions have been made after position j, then the number of k-bit exponentiation operations is at most $\ell(t' + 1)$, which is in $O(\ell m)$. During verification, $O(tm)$ number of k-bits exponentiation operations are required. Hence, our scheme is suitable for small t, which is reasonable in practice. In sum, the main drawback of $\mathcal{RSS}$ is the size of its signature. In the next section, we will reduce its size using a random tree.

5 RGGM: Random tree with Hiding property

We propose RGGM, a variant of GGM tree [7] to generate a sequence of pseudo random numbers, where the structure of the tree is randomized. This generator provides us with the ability to remove multiple substrings of pseudo random numbers, while still being able to generate the retained numbers from a short seed. The expected size of the new seed is in $O(k + tk \log m)$ where t is the number of removed substrings, m is the number of pseudo random numbers, and k is a security parameter. More importantly, the new seed does not reveal any information about the size nor the content of the removed substrings.

Pseudo random number generation. To generate m pseudo random numbers we employ a method similar to that in the redactable signature scheme proposed by Johnson et al. [9], which is based on the GGM tree [7]. Let $G : \mathcal{K} \to \mathcal{K} \times \mathcal{K}$ be a length-doubling pseudo random number generator. First pick an arbitrary binary tree T with m leaves, where all internal nodes of T have exactly two children, the left and right child. Next, pick a seed $t \in \mathcal{K}$ uniformly at random, and associate it with the root. The pseudo random numbers $r_1, r_2, \ldots, r_m$ are then computed from t in the usual top-down manner along the binary tree.

Hiding random numbers. If r_i is to be removed, the associated leaf node and all its ancestors will be removed, as illustrated by the example in Figure 1(b). The values associated with the roots of the remaining subtrees, and a description of the structure of the subtrees, form the new seed, whereby the remaining random values r_j's ($j \neq i$) can be re-computed. By the property of G, it is computationally difficult to guess the removed value r_i from the new seed.

Unlike the method proposed by Johnson et al. [9], our tree T is randomly generated. If the tree is known to be balanced (or known to be of some fixed

```
TreeGen: Given $m$, output a binary tree $T$ with $m$ leaves:

  1. Pick a $p$ uniformly at random from $\{1, 2, \ldots, m-1\}$.
  2. Recursively generate a tree $T_1$ with $p$ leaves.
  3. Recursively generate a tree $T_2$ with $m-p$ leaves.
  4. Output a binary tree with $T_1$ as the left subtree and $T_2$ as the right subtree.
```

Table 6. `TreeGen`: a random tree generation algorithm

structure), some information on the number of leaf nodes removed can be derived from the redacted tree. Our random trees are generated by the probabilistic algorithm `TreeGen` in Table 6. Note that descriptions of the structure of the tree are required for the regeneration of the random values r_i's.

At the moment, for ease of presentation, the descriptions are stored together with the seed. This increases the size of the seed. To reduce the size, we can replace the description by another short random seed $\hat{t}$, which is assigned to the root. The random input required in Step 1 of the algorithm can be generated from $\hat{t}$ using G. A difference between the two methods of storing the (redacted) tree structure information is that in the former, we will have an information theoretic security result, whereas in the later, the security depends on G.

Our main observation is as follows: *after a substring of leaves is removed from the random tree, the remaining subtrees do not reveal (information theoretically) anything about the number of leaves removed, except the fact that at least one leaf has been removed at that location.*

Notations. Given a binary tree T, its leaf nodes can be listed from left to right to obtain a sequence. We call a subsequence of consecutive leaves a substring of leaves. After multiple substrings of leaves and all of their ancestor nodes are deleted, the remaining structures form a *redacted* tree[3] represented by two sequences, $\mathbf{T} = \langle T_1, T_2, \ldots, T_v\rangle$ and $\mathbf{b} = \langle m, b_1, b_2, \ldots, b_u\rangle$, where T_i's are the subtrees retained, and each b_i indicates that a substring was removed between the b_i-th and (b_i+1)-th locations in the remaining sequence of leaf nodes. Let q_i be the number of leaves that were removed in this substring. We call the sequence $\langle m, (b_1, q_1), (b_2, q_2), \ldots, (b_u, q_u)\rangle$ the *original annotation* of $\mathbf{b}$. Thus, the total number of leaf nodes removed is $\sum_{i=1}^{u} q_i$.

Let us consider this process. Given an original annotation $\mathbf{b}_1 = \langle m, (b_1, q_1), (b_2, q_2), \ldots, (b_u, q_u)\rangle$, a random tree T of size $m + \sum_{i=1}^{u} q_i$ is generated using `TreeGen`, and then redacted according to $\mathbf{b}_1$. Let $\mathtt{RED}(\mathbf{b}_1)$ be the redacted tree.

From an adversary's point of view, he has $\mathtt{RED}(\mathbf{b}_1)$, represented as $(\mathbf{T}, \mathbf{b})$, and wants to guess the q_i's in the original annotation $\mathbf{b}_1$. We can show that the additional knowledge of $\mathbf{T}$ does not improve his chances, compared to another adversary who only has the annotation $\mathbf{b}$ but not the tree $\mathbf{T}$. It is suffice to

[3] Although strictly speaking it is a forest.

show that, given any $\mathbf{b}$ and any two possible original annotations $\mathbf{b}_1$ and $\mathbf{b}_2$, the conditional probabilities of obtaining $(\mathbf{T}, \mathbf{b})$ are the same. That is,

THEOREM 4 *For any redacted tree* $(\mathbf{T}, \mathbf{b})$*, any distribution* $\mathcal{B}$ *on the original annotation, and* $\mathbf{b}_1 = \langle m, (b_1, q_1), (b_2, q_2), \ldots, (b_u, q_u)\rangle$*,* $\mathbf{b}_2 = \langle m, (b_1, q'_1), (b_2, q'_2), \ldots, (b_u, q'_u)\rangle$*,*

$$Prob(\mathtt{RED}(\mathcal{B}) = (\mathbf{T}, \mathbf{b}) \mid \mathcal{B} = \mathbf{b}_1) = Prob(\mathtt{RED}(\mathcal{B}) = (\mathbf{T}, \mathbf{b}) \mid \mathcal{B} = \mathbf{b}_2)$$

6 $\mathcal{SRSS}$: A Short Redactable Signature Scheme for Strings

$\mathcal{RSS}$ produces a large signature, whose main portion is a sequence of true random numbers r_i's. We can combine RGGM with $\mathcal{RSS}$ to produce a short signature by replacing the r_i's with pseudo random numbers generated by RGGM. Let us call this combined scheme $\mathcal{SRSS}$, short redactable signature scheme for strings. It is easy to show that $\mathcal{SRSS}$ is unforgeable and privacy preserving from Lemma 1, Theorem 2, Theorem 3, Theorem 4, and the fact that RGGM is a pseudo random number generator.

Unforgeability. From the definition of cryptographic secure pseudo random number generator and Theorem 2, we conclude that $\mathcal{SRSS}$ is unforgeable.

COROLLARY 5 *For any positive polynomials (in* κ*)* t *and* q*,* $\mathcal{SRSS}$ *is* $(t, q, \frac{\epsilon_1}{1-\epsilon_2})$*-unforgeable against existential forgeries with respect to* $\succ$*, if* $\mathtt{SSS}$ *is* $(t+qt_0, q, \epsilon_1)$*-unforgeable against existential forgeries with respect to* $\supseteq$*,* $\mathcal{H}$ *is* $(t + qt_1, \epsilon_2)$*-collision-resistant, and* G *is a cryptographic secure pseudo random number generator, where* t_0 *is the running time of* $\mathcal{H}$*,* t_1 *is the time needed by* $\mathcal{SRSS}$ *to sign a document, and* κ *is the security parameter.*

Privacy. From the definition of cryptographic secure pseudo random number generator, Theorem 3 and Theorem 4, we conclude that $\mathcal{SRSS}$ is privacy preserving.

COROLLARY 6 *The redactable signature scheme* $\mathcal{SRSS}$ *is privacy preserving (as defined in Definition 3), assuming that the hash function* h *satisfies the property: the two distributions* $X = g^{h(U_1)h(U_2)} \bmod n$ *and* $Y = g^{h(U'_1)} \bmod n$ *are computationally indistinguishable, and* G *is a cryptographic secure pseudo random number generator, where* n *is a RSA modulus,* g *is an element of large order in* $\mathbb{Z}_n^*$ *and* U_i*'s and* U'_j*'s are all independent uniform random variables over* $\mathbb{Z}_n$*, and* $h(\cdot)$ *is used to define* $\mathcal{H}$ *in Table 1.*

Efficiency. The improvement of $\mathcal{SRSS}$ is in signature size. Given the unredacted string, the size of the signature is $\kappa + 2k$, where κ is the signature size of $\mathtt{SSS}$, and k is the length of each seed. Recall that we need two seeds in RGGM, one for the generation of the numbers, and the other for the tree structure. If t substrings are removed, the signature size is $\kappa + tk + O(kt \log m)$, where the term tk is for the witness, and $O(kt \log m)$ is required for the RGGM.

7 Other variants

7.1 Allowing removal of empty substring

Both $\mathcal{RSS}$ and $\mathcal{SRSS}$ do not allow removal of empty substrings. In fact, it is considered to be a forgery if a censor declares that a substring has been removed but actually the censor does not remove anything. However, some applications may want to allow removal of empty substrings. This can be achieved by slight modifications to our schemes. To sign a string $x_1 x_2 \ldots x_m$, special symbol $\natural$ is inserted to obtain the expanded string $\widetilde{\mathbf{x}} = \natural x_1 \natural x_2 \natural \ldots \natural x_m \natural$ which will be signed directly using $\mathcal{RSS}$ or $\mathcal{SRSS}$. To remove a substring $\mathbf{x}_0$, the expanded substring of $\mathbf{x}_0$ is actually removed. In the case where a substring has already being removed in front or at the end of $\mathbf{x}_0$, the $\natural$ is not included at the front or the end accordingly. To remove an empty substring, simply remove the $\natural$ at intended location.

7.2 Hiding the fact that the string is redacted

There is a question on whether one should hide the location of a removed substring or even the occurrence of redaction. This requirement is also known as *invisibility* or *transparency* [2, 13]. For a small object space, if invisibility is satisfied, a censor may take a long signed string, remove some substrings to form an arbitrary "authentic" short string. Nevertheless, some applications may need invisibility.

Here is a simple variation of $\mathcal{RSS}$ that achieves this. To sign a string, simply add a special symbols $\sharp$ in-between any two consecutive objects. Sign the expanded string and then immediately redact it by removing all $\sharp$'s. Redaction and verification is the same as before. However, this variant produces a large signature even if we use $\mathcal{SRSS}$. Furthermore, the computation during verification is high. At least $\Omega(m^2)$ exponentiation operations are required.

To reduce the size of signature, there is an alternative: sign all the pairs of objects. To sign the string $\mathbf{x} = x_1 x_2 x_3 \ldots x_m$, first generate random numbers $r_1, r_2, \ldots, r_m$ such that $r_i \| x_i$'s are distinct. Next, let $\mathbf{t}$ be the set of all pairs $\{(r_i \| x_i, r_j \| x_j)\}_{i<j}$ and employ SSS to sign $\mathbf{t}$. When an object x_i is to be removed, simply remove all the pairs that involve x_i from $\mathbf{t}$. Since the role of r_i is to ensure that all elements are distinct, the size of each r_i can be smaller than the random numbers required by $\mathcal{RSS}$.

8 Discussion and Conclusion

We considered a simple but difficult requirement in redactable signature scheme: hiding the lengths of the removed substrings. We exploited an intriguing statistical property of random trees, and employed a hash from strings to unordered sets to achieve the requirement. Although the signature is short, its size still depends on the number of substrings removed and the length of the string. In contrast, there are known schemes for unordered sets, whose signature size is a constant. Hence, it is interesting to find out whether it is possible to close the gap.

The two main components, the hash $\mathcal{H}$ and the RGGM tree, proposed in this paper, could be of independent interests. The hash function may play a role in the design of transitive signature with additional property on privacy preservation. Many secure outsourced database applications involve Merkel tree or GGM tree. The hiding property of the RGGM tree may be useful in those applications.

References

1. David Aldous. The continuum random tree III. *The Annals of Probability*, 21:248–289, 1993.
2. Giuseppe Ateniese, Daniel Chou, Breno Medeiros, and Gene Tsudik. Sanitizable signatures. In *ESORICS*, pages 159–177, 2005.
3. Mihir Bellare, Oded Goldreich, and Shafi Goldwasser. Incremental cryptography: The case of hashing and signing. In *CRYPTO*, pages 216–233, 1994.
4. Mihir Bellare, Oded Goldreich, and Shafi Goldwasser. Incremental cryptography and application to virus protection. In *STOC*, pages 45–56, 1995.
5. Ee-Chien Chang, Chee Liang Lim, and Jia Xu. Short redactable signatures using random trees. Cryptology ePrint Archive, Report 2009/025, 2009. `http://eprint.iacr.org/`.
6. Rosario Gennaro, Shai Halevi, and Tal Rabin. Secure hash-and-sign signatures without the random oracle. In *EUROCRYPT*, pages 123++, 1999.
7. Oded Goldreich, Shafi Goldwasser, and Silvio Micali. How to construct random functions. *J. ACM*, 33(4):792–807, 1986.
8. Tetsuya Izu, Nobuyuki Kanaya, Masahiko Takenaka, and Takashi Yoshioka. Piats: A partially sanitizable signature scheme. In *ICICS*, pages 72–83, 2005.
9. Robert Johnson, David Molnar, Dawn Song, and David Wagner. Homomorphic signature schemes. In *CT-RSA*, pages 244–262, 2002.
10. Marek Klonowski and Anna Lauks. Extended sanitizable signatures. In *ICISC*, pages 343–355, 2006.
11. Ralph Merkle. Protocols for public key cryptosystems. In *SP*, page 122, 1980.
12. Silvio Micali and Ronald Rivest. Transitive signature schemes. In *CT-RSA*, pages 236–243, 2002.
13. Kunihiko Miyazaki, Goichiro Hanaoka, and Hideki Imai. Digitally signed document sanitizing scheme based on bilinear maps. In *ASIACCS*, pages 343–354, 2006.
14. Ronald Rivest. Two new signature schemes. Presented at Cambridge seminar, 2001. http://www.cl.cam.ac.uk/Research/Security/seminars/2000/rivest-tss.pdf.
15. Ron Steinfeld, Laurence Bull, and Yuliang Zheng. Content extraction signatures. In *ICISC*, pages 163–205, 2001.
16. Manabu Suzuki, Isshiki Toshiyuki, and Keisuke Tanaka. Sanitizable signature with secret information. In *SCIS*, 2006.
17. Xun Yi. Directed transitive signature scheme. In *CT-RSA*, pages 129–144, 2007.