Redactable Signature Scheme for Tree : Structured Data Based on Merkle Tree

Redactable Signature Scheme for Tree-Structured Data Based on Merkle Tree

Shoichi Hirose[1] and Hidenori Kuwakado[2]
[1]*Graduate School of Engineering, University of Fukui, Fukui, Japan*
[2]*Faculty of Informatics, Kansai University, Osaka, Japan*
hrs_shch@u-fukui.ac.jp, kuwakado@kansai-u.ac.jp

Keywords: Redactable Signature Scheme: Tree-Structured Data: Hash Function: Merkle Tree

Abstract: In 2008, Kundu and Bertino proposed a structural signature scheme for tree-structured data. A signature generated by the scheme is redactable: for given tree-structured data and its signature, it is possible to compute signatures of subtrees of the given tree without the secret signing key. Brzuska et al. formalized security requirements of such kind of redactable signature schemes. They also proposed a provably secure redactable signature scheme for tree-structured data using an ordinary signature scheme. This paper presents a new redactable signature scheme for tree-structured data using an ordinary signature scheme and a Merkle tree constructed by a keyed hash function such as HMAC. The proposed scheme assumes that the out-degree of each node in a tree is at most constant. It is also shown that the proposed scheme is provably secure under standard security assumptions of the underlying primitives. The proposed scheme first generates a digest of given tree-structured data based on the Merkle tree using the keyed hash function, and computes a single signature for the digest using the ordinary signature scheme. On the other hand, the total number of signatures required by previous provably secure schemes is at least as large as that of the nodes of the tree.

1 INTRODUCTION

Background. It is expected that database outsourcing using cloud service will be popular. An outsourced database is required to provide proofs of the correctness of answers to queries. A digital signature scheme is a cryptographic technique useful for this kind of purposes. A signature makes it possible to verify whether the corresponding data are corrupted or not. On the other hand, even if queries are made on the same data, answers vary according to contents of queries or users' access rights. It is obviously unreasonable, however, to prepare signatures of all possible answers in advance.

In 2008, Kundu and Bertino proposed a structural signature scheme for tree-structured data (Kundu and Bertino, 2008). A signature generated by their scheme is redactable: for given tree-structured data and its signature, it is possible to compute signatures of subtrees of the given tree without the secret signing key. Their security analysis of their scheme was informal, and some attacks against it were presented (Brzuska et al., 2010; Samelin et al., 2012a; Samelin et al., 2012b). Brzuska et al. formalized unforgeability, privacy and transparency as security requirements of such kind of redactable signature schemes and showed that privacy is implied by transparency (Brzuska et al., 2010). They also proposed a provably secure redactable signature scheme for tree-structured data using an ordinary signature scheme. Their scheme satisfies unforgeability and transparency if the underlying signature scheme satisfies unforgeability.

Our Contribution. This paper presents a new redactable signature scheme for tree-structured data using an ordinary signature scheme and the Merkle tree (Merkle, 1989a). The proposed scheme only allows redaction of cutting leaf nodes. Iteration of cutting leaf nodes enables cutting any subtree. In the proposed scheme, to achieve secure redaction, a keyed hash function such as HMAC (Bellare et al., 1996a) is used to construct Merkle trees. It is also shown that the proposed scheme satisfies unforgeability and transparency if the underlying signature scheme satisfies unforgeability and the underlying keyed hash function is a pseudorandom function and collision-resistant. HMAC satisfies both of the properties under reasonable standard assumptions.

Unlike the previous schemes, the proposed

scheme uses the Merkle tree based on a cryptographic hash function such as SHA-2 (FIPS PUB 180-4, 2012). An advantage of this approach is that the proposed scheme is more efficient than the previous schemes. In our scheme, an ordinary signature scheme is used only once for each signature generation to sign the digest corresponding to the root of a Merkle tree. The previous schemes need at least as many calls for the underlying signing procedure as the number of the nodes of the tree-structured data. Generally, the amount of computation required for hashing is much smaller than the amount of computation required for signing. On the other hand, a disadvantage of the proposed scheme is that the out-degree of each node in tree-structured data should be at most constant.

Related Work. The scheme of Kundu and Bertino (Kundu and Bertino, 2008) is based on the fact that a tree is uniquely identified by the pre-order and post-order of its nodes.

The scheme of Brzuska et al. (Brzuska et al., 2010) simply signs the nodes and the edges of tree-structured data separately using an ordinary signature scheme. When their scheme is applied to an ordered tree, the orders of children of nodes are also authenticated by ordinary signatures, which costs $O(\alpha\beta^2)$ calls for the underlying signing procedure in the worst case, where α is the number of the nodes and β is the maximum of out-degrees of the nodes. They only considered redaction of cutting leaf nodes.

The scheme of Samelin et al. (Samelin et al., 2012a) signs the lists of the nodes of a tree in their pre-order and post-order using a redactable signature scheme for lists. Their scheme allows more flexible redaction than the scheme of (Brzuska et al., 2010) and ours: Non-leaf nodes as well as leaf nodes can be redactable. Their scheme needs $O(\alpha)$ calls for the underlying signing procedure.

The scheme of Pöhls et al. (Pöhls et al., 2012) also allows redaction of non-leaf nodes. Their scheme is based on the Merkle tree and the out-degrees of nodes are not limited. Thus, their scheme is more flexible than ours. However, their construction uses a collision resistant one-way accumulator (Benaloh and de Mare, 1993; Barić and Pfitzmann, 1997) for the Merkle tree, which makes the scheme less efficient.

Redactable signature schemes for general graph-structured data have also been proposed recently (Kundu and Bertino, 2010; Kundu et al., 2012). Other types of redactable signature schemes can be found in (Steinfeld et al., 2001; Johnson et al., 2002; Miyazaki et al., 2003; Nojima et al., 2009; Chang et al., 2009; Samelin et al., 2012b), which are not explicitly intended for tree or graph-structured data.

Sanitizable signature schemes (Ateniese et al., 2005; Brzuska et al., 2009) originated from motivations similar to those of redactable signatures. Sanitizable signature schemes allow redaction only by privileged entities called censors or sanitizers.

Ahn et al. (Ahn et al., 2012) provided a general framework for computing on signed data including redactable signature schemes and sanitizable signature schemes. Their privacy notion called context hiding is stronger than transparency in that the former requires the unlinkability between an original signature and the derived signatures.

Organization. The rest of the paper is organized as follows. Section 2 gives some notations and definitions of signature schemes, collision-resistant hash functions and pseudorandom functions. Section 3 defines redactable signature schemes and their security requirements. The proposed scheme is presented in Sec. 4. Its security is discussed in Sec. 5. Section 6 concludes the paper.

2 PRELIMINARIES

Let $\mathbb{N}$ be the set of natural integers. Let $\boldsymbol{F}(\mathcal{X},\mathcal{Y})$ be the set of the functions from $\mathcal{X}$ to $\mathcal{Y}$.

Let $\mathcal{F} = \langle F_\ell \rangle_{\ell\in\mathbb{N}}$ be a sequence of keyed functions $F_\ell : \mathcal{K}_\ell \times \mathcal{X}_\ell \to \mathcal{Y}_\ell$, where $\mathcal{K}_\ell$ is the key space. F_ℓ can also be regarded as a set of functions from $\mathcal{X}_\ell$ to $\mathcal{Y}_\ell$: $\{F_\ell(K,\cdot) \,|\, K \in \mathcal{K}_\ell\}$. It is assumed to be easy to sample an element uniformly at random from $\mathcal{X}_\ell$ and $\mathcal{K}_\ell$.

For a set S, the notation $s \leftarrow S$ represents an assignment of an element chosen uniformly at random from S to a variable s. For an algorithm $\mathcal{M}$, the notation $y \leftarrow \mathcal{M}(x)$ represents an assignment of an output of $\mathcal{M}$ with an input x to a variable y.

2.1 Collision-Resistant Hash Function

Collision resistance is a property of many-to-one functions. It informally means that it is difficult to find a pair of distinct inputs transformed to a same output.

Definition 1. *Let $\mathcal{F} = \langle F_\ell \rangle_{\ell\in\mathbb{N}}$, where F_ℓ is a keyed hash function. If*

$$\Pr[\mathcal{A}(1^\ell, K) = (x,x') \wedge F_\ell(K,x) = F_\ell(K,x') \wedge x \neq x']$$

is negligibly small as a function of ℓ for any efficient collision-finder $\mathcal{A}$, then $\mathcal{F}$ is called a sequence of collision-resistant hash functions. K is uniformly distributed over $\mathcal{K}_\ell$.

2.2 Pseudorandom Function

Definition 2. *For a sequence of keyed functions* $\mathcal{F} = \langle F_\ell \rangle_{\ell \in \mathbb{N}}$, *let*

$$\mathrm{Adv}^{\mathrm{prf}}_{\mathcal{F}}(\mathcal{A}) = \left| \Pr[\mathcal{A}^{F_\ell(K,\cdot)}(1^\ell) = 1] - \Pr[\mathcal{A}^{\rho}(1^\ell) = 1] \right|$$

be the advantage of a distinguisher $\mathcal{A}$ *against* $\mathcal{F}$. K *is uniformly distributed over* $\mathcal{K}_\ell$, *and* ρ *is uniformly distributed over* $\boldsymbol{F}(\mathcal{X}_\ell, \mathcal{Y}_\ell)$. *If* $\mathrm{Adv}^{\mathrm{prf}}_{\mathcal{F}}(\mathcal{A})$ *is negligibly small as a function of* ℓ *for any efficient* $\mathcal{A}$, *then* $\mathcal{F}$ *is called a sequence of pseudorandom functions.*

During security analysis of the proposed scheme, we will deal with distinguishers with independent multiple oracles. Let

$$\mathrm{Adv}^{m\text{-prf}}_{\mathcal{F}}(\mathcal{A}) = \Big| \Pr[\mathcal{A}^{F_\ell(K_1,\cdot),\ldots,F_\ell(K_m,\cdot)}(1^\ell) = 1] - \Pr[\mathcal{A}^{\rho_1,\ldots,\rho_m}(1^\ell) = 1] \Big| \ ,$$

where $(K_1, K_2, \ldots, K_m)$ are uniformly distributed over $\mathcal{K}_\ell{}^m$ and $(\rho_1, \rho_2, \ldots, \rho_m)$ are uniformly distributed over $\boldsymbol{F}(\mathcal{X}_\ell, \mathcal{Y}_\ell)^m$. The following lemma is a paraphrase of Lemma 3.3 in (Bellare et al., 1996b):

Lemma 1. *Let* $\mathcal{A}$ *be a distinguisher for* $\mathcal{F}$ *with access to* m *oracles. Then, a distinguisher* $\mathcal{B}$ *for* $\mathcal{F}$ *can be constructed with* $\mathcal{A}$ *as a subroutine such that*

$$\mathrm{Adv}^{m\text{-prf}}_{\mathcal{F}}(\mathcal{A}) = m \cdot \mathrm{Adv}^{\mathrm{prf}}_{\mathcal{F}}(\mathcal{B}) \ .$$

The running time of $\mathcal{B}$ *is approximately total of the running time of* $\mathcal{A}$ *and the time required to compute* F_ℓ *to answer to the queries made by* $\mathcal{A}$. $\mathcal{B}$ *makes at most* $\max\{q_i \mid 1 \le i \le m\}$ *queries to its oracle, where* q_i *is the number of the queries made by* $\mathcal{A}$ *to its* i*-th oracle.*

2.3 Iterated Hash Function and HMAC

The proposed redactable signature scheme uses a hash function with arbitrary input length satisfying that it is collision-resistant and its keyed mode is a pseudorandom function.

Widely deployed hash functions such as SHA-1/2 are called iterated hash functions. An iterated hash function H consists of iteration of a compression function f with fixed length input and output as follows: $H(M) = f(f(\cdots f(f(IV, M_1), M_2), \ldots), M_l)$, where $M = M_1 \| M_2 \| \cdots \| M_l$ is an input message and IV is a fixed initial value. A compression function is a kind of hash function with inputs of fixed length. This iteration is called Merkle-Damgård construction (Damgård, 1989; Merkle, 1989b). An iterated hash function with appropriate input preprocessing (padding with the Merkle-Damgård strengthening (Menezes et al., 1996)) is collision-resistant if the underlying compression function is collision-resistant (Damgård, 1989; Merkle, 1989b).

HMAC (Bellare et al., 1996a) is a keyed function for generating message authentication codes, which is constructed with an iterated hash function in the following way:

$$\mathsf{HMAC}(K, M) = H((K \oplus \mathtt{opad}) \| H((K \oplus \mathtt{ipad}) \| M)) \ ,$$

where H is a hash function, K is a secret key and `opad` and `ipad` are distinct constants. It was shown that HMAC is a pseudorandom function if the underlying compression function is a pseudorandom function (Bellare, 2006). Additionally, it is easy to see that HMAC is collision-resistant in the sense that it is difficult to find a pair of distinct inputs (K, M) and (K', M') such that $\mathsf{HMAC}(K, M) = \mathsf{HMAC}(K', M')$ if H is collision-resistant.

2.4 Signature Scheme

A signature scheme Sig consists of three algorithms $(\mathsf{K}, \mathsf{S}, \mathsf{V})$ defined as follows.

Key Generation $(sk, pk) \leftarrow \mathsf{K}(1^\ell)$.
For a given security parameter ℓ, the key generation algorithm K generates a pair of a secret (signing) key sk and a corresponding public (verification) key pk.

Signing $(M, \sigma) \leftarrow \mathsf{S}(sk, M)$.
Given a secret key sk and a message M, the signing algorithm S outputs M and its signature σ.

Verification $d \leftarrow \mathsf{V}(pk, M, \sigma)$, where $d \in \{0, 1\}$.
Given a public key pk, a message M and a signature σ, the verification algorithm V decides whether σ is a valid signature for M with respect to pk. If valid, V outputs 1. Otherwise, it outputs 0.

Existential unforgeability against adaptive chosen message attacks is defined as follows.

Definition 3. *A signature scheme* $\mathsf{Sig} = (\mathsf{K}, \mathsf{S}, \mathsf{V})$ *is existentially unforgeable against adaptive chosen message attacks if the probability that the experiment* $\mathtt{EU}^{\mathsf{Sig}}_{\mathcal{A}}$ *given in Algorithm 1 outputs* 1 *is negligibly small as a function of* ℓ *for any efficient forger* $\mathcal{A}$.

3 REDACTABLE SIGNATURE SCHEME FOR TREE-STRUCTURED DATA

3.1 Tree-Structured Data

The proposed redactable signature scheme is used to sign directed-tree-structured data. A directed tree has

Algorithm 1 Experiment $\mathtt{EU}_{\mathcal{A}}^{\mathsf{Sig}}(\ell)$

$(sk, pk) \leftarrow \mathsf{K}(1^\ell)$
$(M, \sigma) \leftarrow \mathcal{A}^{\mathsf{S}(sk,\cdot)}(pk)$
▷ Let $M_1, M_2, \ldots, M_q$ be $\mathcal{A}$'s queries to S.
if $\mathsf{V}(pk, M, \sigma) = 1 \wedge M \neq M_i$ for $1 \leq i \leq q$ **then**
  return 1
else
  return 0

a unique node with in-degree 0, which is called the root. In the remaining part, a directed tree is simply called a tree. For tree-structured data T and T', $T' \preceq T$ means that T' is a subtree of T and T' and T share the root.

3.2 Redactable Signature Scheme for Tree Structured Data

A redactable signature scheme tSig for tree-structured data consists of four algorithms $(\mathsf{tK}, \mathsf{tS}, \mathsf{tV}, \mathsf{tC})$ defined below.

Key Generation $(sk, pk) \leftarrow \mathsf{tK}(1^\ell)$.
Given a security parameter ℓ, the key generation algorithm tK generates a secret (signing) key sk and a corresponding public (verification) key pk.

Signing $(T, \sigma) \leftarrow \mathsf{tS}(sk, T)$.
Given a secret key sk and tree-structured data T, the signing algorithm tS outputs T and its signature σ.

Verification $d \leftarrow \mathsf{tV}(pk, T, \sigma)$, where $d \in \{0, 1\}$.
Given a public key pk, tree-structured data T and a signature σ, the verification algorithm tV decides whether σ is a valid signature for T with respect to pk. If valid, then tV outputs 1. Otherwise, it outputs 0.

Cutting $(T', \sigma') \leftarrow \mathsf{tC}(pk, T, \sigma, L)$.
Given a public key pk, tree-structured data T, a signature σ and a leaf L of T, the cutting algorithm tC outputs $T' = T \backslash L$ and its signature σ'.

Notice that the secret key sk is not given to the cutting algorithm.

3.3 Security

Brzuska et al. formalized unforgeability, privacy and transparency as security requirements of redactable signature schemes for tree-structured data (Brzuska et al., 2010). They also showed that transparency implies privacy. Namely, if a redactable signature scheme satisfies transparency, then it also satisfies privacy.

Definition 4 (Unforgeability (Brzuska et al., 2010)). *A redactable signature scheme for tree-structured data* $\mathsf{tSig} = (\mathsf{tK}, \mathsf{tS}, \mathsf{tV}, \mathsf{tC})$ *is existentially unforgeable against adaptive chosen message attacks if the probability that the experiment* $\mathtt{EU}_{\mathcal{A}}^{\mathsf{tSig}}$ *given in Algorithm 2 outputs* 1 *is negligibly small as a function of* ℓ *for any efficient forger* $\mathcal{A}$.

Algorithm 2 Experiment $\mathtt{EU}_{\mathcal{A}}^{\mathsf{tSig}}(\ell)$

$(sk, pk) \leftarrow \mathsf{tK}(1^\ell)$
$(T, \sigma) \leftarrow \mathcal{A}^{\mathsf{tS}(sk,\cdot)}(pk)$
▷ Let $T_1, T_2, \ldots, T_q$ be $\mathcal{A}$'s queries to tS.
if $\mathsf{tV}(pk, T, \sigma) = 1 \wedge T \not\preceq T_i$ for $1 \leq i \leq q$ **then**
  return 1
else
  return 0

Transparency informally states that no one should be able to tell whether a signature of a tree is created only with a signing algorithm or with both a signing algorithm and a cutting algorithm.

Definition 5 (Transparency (Brzuska et al., 2010)). *A redactable signature scheme for tree-structured data* $\mathsf{tSig} = (\mathsf{tK}, \mathsf{tS}, \mathsf{tV}, \mathsf{tC})$ *is transparent if*

$$\mathrm{Adv}^{\mathsf{tr}}_{\mathsf{tSig}}(\mathcal{A}) = \left| \Pr[\mathtt{Tr}^{\mathsf{tSig}}_{\mathcal{A}}(\ell) = 1] - 1/2 \right|$$

is negligibly small as a function of ℓ *for any efficient distinguisher* $\mathcal{A}$. *The experiment* $\mathtt{Tr}^{\mathsf{tSig}}_{\mathcal{A}}$ *is given in Algorithm 3.*

Algorithm 3 Experiment $\mathtt{Tr}^{\mathsf{tSig}}_{\mathcal{A}}(\ell)$

$(sk, pk) \leftarrow \mathsf{tK}(1^\ell)$
$b \leftarrow \{0, 1\}$
$d \leftarrow \mathcal{A}^{\mathsf{tS}(sk,\cdot), \mathsf{SorC}(\cdot,\cdot,sk,b)}(pk)$
if $d = b$ **then**
  return 1
else
  return 0

function $\mathsf{SorC}(T, L, sk, b)$
  if $b = 0$ **then**
    $(T, \sigma) \leftarrow \mathsf{tS}(sk, T)$
    $(T', \sigma') \leftarrow \mathsf{tC}(pk, T, \sigma, L)$
  else
    $T' \leftarrow T \backslash L$
    $(T', \sigma') \leftarrow \mathsf{tS}(sk, T')$
  return (T', σ')

4 PROPOSED SCHEME

The proposed scheme requires that the out-degree of each node of given tree-structured data is at most d. The proposed scheme is suitable for positional trees. A positional tree is an ordered tree such that children of a node are assigned distinct integers (e.g., in $\{0,1,\ldots,d-1\}$ in the current case) compatible with their order. Nevertheless, the proposed scheme is also applicable to unordered trees, which will be mentioned later.

The algorithms for key generation, signing, verification and cutting of the proposed scheme are presented below. In the descriptions, a node of the tree $T=(V_T,E_T)$ is denoted by v_i and the subscript i is used as an ID of the node. In tree-structured data, nodes and edges are assigned some data items in general. It is assumed that data items assigned to a node include the data items assigned to its incoming edge.

Key Generation $(H,K;sk,pk) \leftarrow \mathsf{tK}(1^\ell)$.

H is an iterated hash function composed of a compression function chosen uniformly at random from a set of functions corresponding to a given security parameter ℓ. K is a secret key used for HMAC using H. We will slightly abuse the notation and denote HMAC using H with key K by H_K. sk and pk are a secret key and a corresponding public key of a signature scheme Sig, respectively. Sig is an ordinary signature scheme. H is public, and K is secret for a signer.

Signing $(T,\sigma) \leftarrow \mathsf{tS}(H,K,sk,T)$.

The signing process proceeds as follows:

1. For each node of the tree-structured data T, if it has less than d children, then add dummy child nodes and the corresponding dummy edges so that it has exactly d children. This process is also applied to the leaves of T. Let V'_T and E'_T represent the set of dummy nodes and the set of dummy edges, respectively.
2. Select a nonce $r \in \{0,1\}^n$ uniformly at random, where $n=\ell^{O(1)}$, and for each node $v_i \in V_T \cup V'_T$, compute the key $r_i = H_K(r\|i)$. n should be large enough to prevent collision of nonces. It is usually sufficient if $n=|K|$.
3. Construct a Merkle tree using HMAC in the following way:
   (a) For each dummy node $v_i \in V'_T$, compute the digest $a_i = H_{r_i}(\lambda)$, where λ is an empty string. No data items are assigned to dummy nodes and their incoming edges.
   (b) For each node $v_i \in V_T$, compute the digest
   $$a_i = H_{r_i}(D_i\|a_{i,0}\|\cdots\|a_{i,d-1}) \ ,$$
   where D_i is data assigned to v_i, and $a_{i,0},\ldots,a_{i,d-1}$ are digests corresponding to the children of v_i.
4. Compute a signature σ_ε of the digest a_ε corresponding to the root v_ε of T using the signature scheme Sig.

The signature σ of T consists of

- σ_ε,
- digests corresponding to the dummy nodes, and
- keys corresponding to the nodes of T.

Namely, $\sigma = (\sigma_\varepsilon, \{a_i \,|\, v_i \in V'_T\}, \{r_i \,|\, v_i \in V_T\})$. Notice that the secret keys corresponding to the dummy nodes are never disclosed.

Figure 1 shows an example of binary-tree-structured data augmented with dummy nodes and edges by the proposed signing algorithm. Dummy nodes and edges are represented by dotted lines. a_i's and r_i's are digests and keys, respectively, included in the signature.

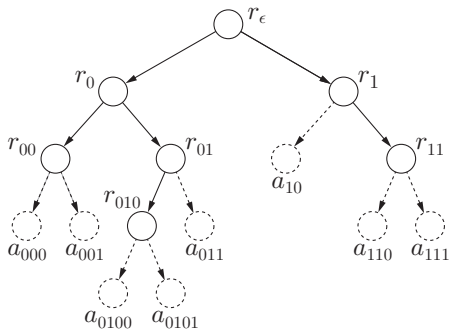

Figure 1: An example of binary-tree-structured data augmented with dummy nodes and edges by the proposed signing algorithm. Dummy nodes and edges are represented by dotted lines.

Verification $d \leftarrow \mathsf{tV}(H,pk,T,\sigma)$.

The verification process proceeds as follows:

1. For the signature $\sigma = (\sigma_\varepsilon, \{a_i \,|\, v_i \in V'_T\}, \{r_i \,|\, v_i \in V_T\})$, compute the digest a'_ε corresponding to the root from the digensts in $\{a_i \,|\, v_i \in V'_T\}$ and the keys in $\{r_i \,|\, v_i \in V_T\}$.
2. Output 1 if σ_ε is a valid signature of a'_ε with respect to pk by Sig. Otherwise, output 0.

Cutting $(T',\sigma') \leftarrow \mathsf{tC}(H,pk,T,\sigma,L)$.

Since L is a leaf of T and $T'=T\backslash L$, the signature σ' of T' consists of

- σ_ε,
- the digests corresponding to L and the dummy nodes except for the dummy children of L, and
- the keys corresponding to the nodes in $V_T \backslash \{L\}$.

Namely, $\sigma' = (\sigma_\varepsilon, \{a_i \,|\, v_i \in (V'_T \backslash \mathrm{Child}(L)) \cup L\}, \{r_i \,|\, v_i \in V_T \backslash \{L\}\})$, where $\mathrm{Child}(L)$ is the set of children of L. It is easy to see that σ' can easily be obtained from σ. The cutting is simply a process to regard the leaf cut off as a dummy node. Figure 2 gives an example of binary-tree-structured data obtained from the tree in Figure 1 by cutting a leaf labeled by 010.

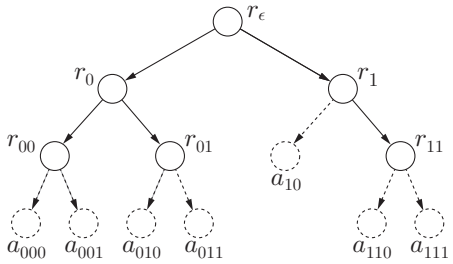

Figure 2: An example of binary-tree-structured data obtained from the tree given in Figure 1 by cutting a leaf labeled by 010. a_i's and r_i's are digests and keys, respectively, included in the signature.

Remark 1. *The proposed scheme presented above can also be applied to unordered trees with slight modification of HMAC. When constructing a Merkle tree in Step 3(b) of Signing, compute the digest as follows:*

$$a_i = H_{r_i}(D_i \| a_{i,j_0} \| \cdots \| a_{i,j_{d-1}}) \ ,$$

where $a_{i,j_0} \| \cdots \| a_{i,j_{d-1}}$ is concatenation of sorted $a_{i,0}, \ldots, a_{i,d-1}$ in the lexicographical order. This modification is based on the commutative hash function introduced by Goodrich and Tamassia (Anagnostopoulos et al., 2001). HMAC with this modification remains pseudorandom as a keyed function and collision-resistant since the modification only makes the domain smaller.

5 SECURITY ANALYSIS OF PROPOSED SCHEME

In this section, the security of the proposed redactable signature scheme is proved under standard assumptions on the security of the building blocks. Adversaries are assumed to be computationally bounded.

5.1 Unforgeability

The following theorem shows that unforgeability of the proposed scheme is reducible to unforgeability of the underlying ordinary signature scheme and collision resistance of the underlying hash function.

Theorem 1. *If* Sig *is existentially unforgeable against adaptive chosen message attacks and H is collision-resistant, then* tSig *is existentially unforgeable.*

Proof. Let $\mathcal{A}$ be any forger of tSig. For $\mathtt{EU}^{\mathsf{tSig}}_{\mathcal{A}}(\ell)$, let

$$(H, K; sk, pk) \leftarrow \mathsf{tK}(1^\ell) \ ,$$
$$(\hat{T}, \hat{\sigma}) \leftarrow \mathcal{A}^{\mathsf{tS}(H,K,sk,\cdot)}(H, pk) \ .$$

Let $T_1, T_2, \ldots, T_q$ be queries to the oracle tS made by $\mathcal{A}$. Let $a_{\varepsilon,i}$ be the digest corresponding to the root of T_i and $\hat{a}_\varepsilon$ be the digest corresponding to the root of $\hat{T}$. Let $A(q) = \{a_{\varepsilon,i} \,|\, 1 \le i \le q\}$.

Suppose that $\hat{T} \not\preceq T_i$ for $1 \le i \le q$. Then, for the success probability of $\mathcal{A}$,

$$\begin{aligned}
&\Pr[\mathsf{tV}(H, pk, \hat{T}, \hat{\sigma}) = 1] \\
&= \Pr[\hat{a}_\varepsilon \notin A(q) \wedge \mathsf{tV}(H, pk, \hat{T}, \hat{\sigma}) = 1] + \\
&\qquad \Pr[\hat{a}_\varepsilon \in A(q) \wedge \mathsf{tV}(H, pk, \hat{T}, \hat{\sigma}) = 1] \\
&\le \Pr[\mathsf{tV}(H, pk, \hat{T}, \hat{\sigma}) = 1 \,|\, \hat{a}_\varepsilon \notin A(q)] + \Pr[\hat{a}_\varepsilon \in A(q)] \ .
\end{aligned}$$

The first term $\Pr[\mathsf{tV}(H, pk, \hat{T}, \hat{\sigma}) = 1 \,|\, \hat{a}_\varepsilon \notin A(q)]$ is equal to the success probability of a forger $\mathcal{B}$ for Sig. $\mathcal{B}$ can be constructed with $\mathcal{A}$ as a subroutine. The running time of $\mathcal{B}$ is total of the running time of $\mathcal{A}$ and the time required to execute the signing algorithm tS of tSig using the signing oracle S of Sig to answer to the queries made by $\mathcal{A}$. The number of the queries to S made by $\mathcal{B}$ equals that of the queries to tS made by $\mathcal{A}$.

The second term $\Pr[\hat{a}_\varepsilon \in A(q)]$ is equal to the success probability of a collision-finder $\mathcal{C}$ of H. $\mathcal{C}$ can also be constructed with $\mathcal{A}$ as a subroutine. The running time of $\mathcal{C}$ is total of the running time of $\mathcal{A}$ and the time required to execute the signing algorithm tS of tSig. □

If the random oracle model is required for the existential unforgeability of the underlying signature Sig, then Theorem 1 is valid only in the random oracle model.

5.2 Transparency

The following theorem implies that the transparency of the proposed scheme is reducible to the pseudorandom-function property of the underlying HMAC.

Theorem 2. *If HMAC is a pseudorandom function, then* tSig *is transparent.*

Proof. Let $\mathcal{A}$ be any distinguisher for tSig. In the experiment $\mathtt{Tr}_{\mathcal{A}}^{\mathsf{tSig}}(\ell)$, suppose that $\mathcal{A}$ makes q_1 queries to tS and q_2 queries to SorC. Suppose that the total number of the nodes of tree-structured data in queries made by $\mathcal{A}$ is N.

Let $\widetilde{\mathsf{tSig}}$ be same as tSig except for using a random function ρ instead of HMAC H_K when generating the secret keys r_i for the nodes in the trees. The domain and range of ρ equal those of H_K, respectively. ρ behaves as a random oracle: it selects the output uniformly at random for a new input. Then,

$$\mathrm{Adv}^{\mathrm{tr}}_{\mathsf{tSig}}(\mathcal{A}) \leq \left|\Pr[\mathtt{Tr}_{\mathcal{A}}^{\widetilde{\mathsf{tSig}}}(\ell)=1]-1/2\right| + \left|\Pr[\mathtt{Tr}_{\mathcal{A}}^{\mathsf{tSig}}(\ell)=1]-\Pr[\mathtt{Tr}_{\mathcal{A}}^{\widetilde{\mathsf{tSig}}}(\ell)=1]\right| \ .$$

For the second term, there exists a distinguisher $\mathcal{D}$ for HMAC using H such that

$$\left|\Pr[\mathtt{Tr}_{\mathcal{A}}^{\mathsf{tSig}}(\ell)=1]-\Pr[\mathtt{Tr}_{\mathcal{A}}^{\widetilde{\mathsf{tSig}}}(\ell)=1]\right| = \mathrm{Adv}^{\mathrm{prf}}_{H}(\mathcal{D}) \ .$$

$\mathcal{D}$ can be constructed with $\mathcal{A}$ as a subroutine. The running time of $\mathcal{D}$ is total of the running time of $\mathcal{A}$ and the time required to execute the signing algorithm tS of tSig and the sign-or-cut algorithm SorC using its oracle to answer to the queries made by $\mathcal{A}$ in the experiment. The oracle of $\mathcal{D}$ is H_K for tSig and it is ρ for $\widetilde{\mathsf{tSig}}$. $\mathcal{D}$ makes at most $N(d+1)$ queries to its oracle.

Let $\widehat{\mathsf{tSig}}$ be same as tSig ($\widetilde{\mathsf{tSig}}$) except for generating the secret keys r_i for the nodes in the trees uniformly at random. Then,

$$\left|\Pr[\mathtt{Tr}_{\mathcal{A}}^{\widetilde{\mathsf{tSig}}}(\ell)=1]-1/2\right| \leq \left|\Pr[\mathtt{Tr}_{\mathcal{A}}^{\widehat{\mathsf{tSig}}}(\ell)=1]-1/2\right| + \left|\Pr[\mathtt{Tr}_{\mathcal{A}}^{\widetilde{\mathsf{tSig}}}(\ell)=1]-\Pr[\mathtt{Tr}_{\mathcal{A}}^{\widehat{\mathsf{tSig}}}(\ell)=1]\right| \ .$$

Notice that $\widehat{\mathsf{tSig}}$ and $\widetilde{\mathsf{tSig}}$ are completely indistinguishable as long as there exists no collision in the nonces r. Thus, the second term of the inequality above is bounded from above by

$$\Pr[\text{Collision in } r] \leq (q_1+q_2)^2/2^{n+1} \ ,$$

where n is the length of the nonce r. Hence,

$$\mathrm{Adv}^{\mathrm{tr}}_{\mathsf{tSig}}(\mathcal{A}) \leq \left|\Pr[\mathtt{Tr}_{\mathcal{A}}^{\widehat{\mathsf{tSig}}}(\ell)=1]-1/2\right| + \mathrm{Adv}^{\mathrm{prf}}_{H}(\mathcal{D}) + (q_1+q_2)^2/2^{n+1} \ .$$

For the experiment $\mathtt{Tr}_{\mathcal{A}}^{\widehat{\mathsf{tSig}}}(\ell)$, let us consider the following distinguisher $\mathcal{B}$ for H. $\mathcal{B}$ has q_2 oracles. They are either $H_{s_1}, H_{s_2}, \ldots, H_{s_{q_2}}$ or $\rho_1, \rho_2, \ldots, \rho_{q_2}$, where $s_1, s_2, \ldots, s_{q_2}$ are chosen independently and uniformly at random, and $\rho_1, \rho_2, \ldots, \rho_{q_2}$ are independent random oracles. $\mathcal{B}$ runs $\mathtt{Tr}_{\mathcal{A}}^{\widehat{\mathsf{tSig}}}(\ell)$ except that, for the j-th query to SorC made by $\mathcal{A}$, $\mathcal{B}$ computes the digest corresponding to the leaf cut off using its j-th oracle. Then,

$$\left|\Pr[\mathtt{Tr}_{\mathcal{A}}^{\widehat{\mathsf{tSig}}}(\ell)=1]-1/2\right| = \mathrm{Adv}^{q_2\text{-prf}}_{H}(\mathcal{B})$$

since

$$\Pr[\mathcal{B}^{O_1,O_2,\ldots,O_{q_2}}=1] = \begin{cases} \Pr[\mathtt{Tr}_{\mathcal{A}}^{\widehat{\mathsf{tSig}}}(\ell)=1] & \text{if } O_j = H_{s_j} \text{ for } 1 \leq j \leq q_2 \\ 1/2 & \text{if } O_j = \rho_j \text{ for } 1 \leq j \leq q_2 \ . \end{cases}$$

From Lemma 1, there exists a distinguisher for H such that $\mathrm{Adv}^{q_2\text{-prf}}_{H}(\mathcal{B}) = q_2\mathrm{Adv}^{\mathrm{prf}}_{H}(\mathcal{C})$. $\mathcal{C}$ makes at most 1 query. The running time of $\mathcal{C}$ is approximately total of the running time of $\mathcal{B}$ and the time required to compute H to answer to the queries made by $\mathcal{B}$.

Thus,

$$\mathrm{Adv}^{\mathrm{tr}}_{\mathsf{tSig}}(\mathcal{A}) \leq q_2\mathrm{Adv}^{\mathrm{prf}}_{H}(\mathcal{C}) + \mathrm{Adv}^{\mathrm{prf}}_{H}(\mathcal{D}) + \frac{(q_1+q_2)^2}{2^{n+1}} \ .$$

This completes the proof. □

6 CONCLUSION

A redactable signature scheme for tree-structured data has been proposed in this paper. It has also been shown that it satisfies both unforgeability and transparency. The proposed scheme is based on the Merkle tree and expected to be quite efficient compared to previous provably secure schemes. Future work includes experimental performance evaluation of the proposed scheme together with previous schemes. It does not seem difficult to extend the proposed scheme to directed acyclic graphs, which is also left as future work.

Acknowledgments

The authors would like to thank the anonymous reviewers for their valuable comments. This work was supported by JSPS KAKENHI Grant Number 20300003 and 25330150.

REFERENCES

Ahn, J. H., Boneh, D., Camenisch, J., Hohenberger, S., Shelat, A., and Waters, B. (2012). Computing on authenticated data. In Cramer, R., editor, *TCC*, volume 7194

of *Lecture Notes in Computer Science*, pages 1–20. Springer.

Anagnostopoulos, A., Goodrich, M. T., and Tamassia, R. (2001). Persistent authenticated dictionaries and their applications. In Davida, G. I. and Frankel, Y., editors, *ISC*, volume 2200 of *Lecture Notes in Computer Science*, pages 379–393. Springer.

Ateniese, G., Chou, D. H., de Medeiros, B., and Tsudik, G. (2005). Sanitizable signatures. In di Vimercati, S. D. C., Syverson, P. F., and Gollmann, D., editors, *ESORICS*, volume 3679 of *Lecture Notes in Computer Science*, pages 159–177. Springer.

Barić, N. and Pfitzmann, B. (1997). Collision-free accumulators and fail-stop signature schemes without trees. In Fumy, W., editor, *EUROCRYPT*, volume 1233 of *Lecture Notes in Computer Science*, pages 480–494. Springer.

Bellare, M. (2006). New proofs for NMAC and HMAC: Security without collision-resistance. In *CRYPTO 2006 Proceedings, Lecture Notes in Computer Science 4117*, pages 602–619. The full version is "Cryptology ePrint Archive: Report 2006/043" at `http://eprint.iacr.org/`.

Bellare, M., Canetti, R., and Krawczyk, H. (1996a). Keying hash functions for message authentication. In Koblitz, N., editor, *CRYPTO*, volume 1109 of *Lecture Notes in Computer Science*, pages 1–15. Springer.

Bellare, M., Canetti, R., and Krawczyk, H. (1996b). Pseudorandom functions revisited: The cascade construction and its concrete security. In *Proceedings of the 37th IEEE Symposium on Foundations of Computer Science*, pages 514–523.

Benaloh, J. C. and de Mare, M. (1993). One-way accumulators: A decentralized alternative to digital sinatures (extended abstract). In Helleseth, T., editor, *EUROCRYPT*, volume 765 of *Lecture Notes in Computer Science*, pages 274–285. Springer.

Brassard, G., editor (1990). *Advances in Cryptology - CRYPTO '89, 9th Annual International Cryptology Conference, Santa Barbara, California, USA, August 20-24, 1989, Proceedings*, volume 435 of *Lecture Notes in Computer Science*. Springer.

Brzuska, C., Busch, H., Dagdelen, Ö., Fischlin, M., Franz, M., Katzenbeisser, S., Manulis, M., Onete, C., Peter, A., Poettering, B., and Schröder, D. (2010). Redactable signatures for tree-structured data: Definitions and constructions. In Zhou, J. and Yung, M., editors, *ACNS*, volume 6123 of *Lecture Notes in Computer Science*, pages 87–104.

Brzuska, C., Fischlin, M., Freudenreich, T., Lehmann, A., Page, M., Schelbert, J., Schröder, D., and Volk, F. (2009). Security of sanitizable signatures revisited. In Jarecki, S. and Tsudik, G., editors, *Public Key Cryptography*, volume 5443 of *Lecture Notes in Computer Science*, pages 317–336. Springer.

Chang, E.-C., Lim, C. L., and Xu, J. (2009). Short redactable signatures using random trees. In Fischlin, M., editor, *CT-RSA*, volume 5473 of *Lecture Notes in Computer Science*, pages 133–147. Springer.

Damgård, I. (1989). A design principle for hash functions. In (Brassard, 1990), pages 416–427.

FIPS PUB 180-4 (2012). Secure hash standard (SHS).

Johnson, R., Molnar, D., Song, D. X., and Wagner, D. (2002). Homomorphic signature schemes. In Preneel, B., editor, *CT-RSA*, volume 2271 of *Lecture Notes in Computer Science*, pages 244–262. Springer.

Kundu, A., Atallah, M. J., and Bertino, E. (2012). Leakage-free redactable signatures. In Bertino, E. and Sandhu, R. S., editors, *CODASPY*, pages 307–316. ACM.

Kundu, A. and Bertino, E. (2008). Structural signatures for tree data structures. *Proceedings of the Very Large Data Base Endowment*, 1(1):138–150.

Kundu, A. and Bertino, E. (2010). How to authenticate graphs without leaking. In Manolescu, I., Spaccapietra, S., Teubner, J., Kitsuregawa, M., Léger, A., Naumann, F., Ailamaki, A., and Özcan, F., editors, *EDBT*, volume 426 of *ACM International Conference Proceeding Series*, pages 609–620. ACM.

Menezes, A. J., van Oorschot, P. C., and Vanstone, S. A. (1996). *Handbook of Applied Cryptography*. CRC Press.

Merkle, R. C. (1989a). A certified digital signature. In (Brassard, 1990), pages 218–238.

Merkle, R. C. (1989b). One way hash functions and DES. In (Brassard, 1990), pages 428–446.

Miyazaki, K., Susaki, S., Iwamura, M., Matsumoto, T., Sasaki, R., and Yoshiura, H. (2003). Digital document sanitizing problem. Technical Report ISEC2003–20, IEICE.

Nojima, R., Tamura, J., Kadobayashi, Y., and Kikuchi, H. (2009). A storage efficient redactable signature in the standard model. In Samarati, P., Yung, M., Martinelli, F., and Ardagna, C. A., editors, *ISC*, volume 5735 of *Lecture Notes in Computer Science*, pages 326–337. Springer.

Pöhls, H. C., Samelin, K., de Meer, H., and Posegga, J. (2012). Flexible redactable signature schemes for trees - extended security model and construction. In Samarati, P., Lou, W., and Zhou, J., editors, *SECRYPT*, pages 113–125. SciTePress.

Samelin, K., Pöhls, H. C., Bilzhause, A., Posegga, J., and de Meer, H. (2012a). On structural signatures for tree data structures. In Bao, F., Samarati, P., and Zhou, J., editors, *ACNS*, volume 7341 of *Lecture Notes in Computer Science*, pages 171–187. Springer.

Samelin, K., Pöhls, H. C., Bilzhause, A., Posegga, J., and de Meer, H. (2012b). Redactable signatures for independent removal of structure and content. In Ryan, M. D., Smyth, B., and Wang, G., editors, *ISPEC*, volume 7232 of *Lecture Notes in Computer Science*, pages 17–33. Springer.

Steinfeld, R., Bull, L., and Zheng, Y. (2001). Content extraction signatures. In Kim, K., editor, *ICISC*, volume 2288 of *Lecture Notes in Computer Science*, pages 285–304. Springer.