Redactable Signatures for Tree-Structured Data: Definitions and Constructions

Christina Brzuska, Heike Busch, Özgür Dagdelen, Marc Fischlin, Martin Franz, Stefan Katzenbeisser, Mark Manulis, Cristina Onete, Andreas Peter, Bertram Poettering, and Dominique Schröder

Technical University of Darmstadt
Center for Advanced Security Research Darmstadt (CASED)

Abstract. Kundu and Bertino (VLDB 2008) recently introduced the idea of structural signatures for trees which support public redaction of subtrees (by third-party distributors) while pertaining the integrity of the remaining parts. An example is given by signed XML documents of which parts should be sanitized before being published by a distributor not holding the signing key. Kundu and Bertino also provide a construction, but fall short of providing formal security definitions and proofs. Here we revisit their work and give rigorous security models for the redactable signatures for tree-structured data, relate the notions, and give a construction that can be proven secure under standard cryptographic assumptions.

1 Introduction

The XML data format is increasingly used to store and organize data. This development is most notable in the context of XML databases, which store the entire content in XML files. In some applications, both the integrity and the authenticity of the stored data must be ensured; this can in principle be achieved by signing the tree with a conventional cryptographic signature. In some scenarios, the content of the tree is privacy sensitive and an access control mechanism determines which part of the tree may be accessed by a specific user. The database management system must therefore be able to prune a tree upon access, so that those parts of the tree that the user is not allowed to see are removed prior to access. Still, it should be possible to prove the authenticity of the remaining data with respect to the original signer, without having to re-sign the document.

This can in principle be resolved by applying sanitizable signatures [2], which allow to overwrite certain parts of the data with a special null symbol while retaining the integrity of the data's signature. Such schemes also guarantee that the signature does not allow the recovery of deleted parts. Unfortunately, pure sanitization of data is insufficient to guarantee privacy: the recipient of the data clearly sees (due to the presence of the null symbol) that some data has been removed from the tree. This mere fact may already be an unwanted privacy leak.

Consider the following example taken from [6]. An XML file describes the health records of a single person. The root node's successors encode visits to

a medical institution, whereas their successors encode results of medical tests performed at those institutions. Since nodes in XML files are ordered (and the tree structure may be publicly known), the recipient of the sanitized tree will be able to associate positions of null symbols in his tree with medical checks performed by the institution. For example, if a null symbol appears at a tree position associated with an HIV test, the recipient knows that such a test was performed, even without knowing its outcome. This information is already a privacy intrusion. Even worse breaches occur when several null symbols appear in positions associated with checkups for a certain disease: the recipient then quite accurately reconstructs the diagnosis.

To avoid such problems, it is therefore necessary to have a sanitizable signature that hides any performed santizations. Kundu and Bertino [6] proposed such a signature scheme for trees. However, they did not formally define the desired security properties; consequently, they were unable to formally prove the security of their scheme. Indeed, we show in this paper that their construction does not meet a strong security requirement.

Our Results. In this paper, we revisit the problem and give precise definitions for the privacy requirements. That is, besides the unforgeability of structural tree signatures, we also use game-based definitions to describe the notions of privacy (deleted data cannot be recovered) and transparency (recipients cannot even determine whether parts have been redacted). Our definitions are furthermore strong in the sense that they, for instance, allow the adversary to adaptively ask for multiple signatures for chosen tree structures.

Given our definitions of the desired security properties, we then formally relate these notions, showing that transparency implies privacy, whereas the converse is not true. We also show that the scheme of Kundu and Bertino [6] does not achieve transparency, even if the adversary may only ask for a single signature.

We then provide a secure construction of a tree signature scheme for ordered trees supporting redaction of subtrees. Our construction can be implemented with any EU-CMA signature scheme and provides reasonable efficiency. While for general trees with n nodes and large out-degree in nodes, it requires $\mathcal{O}(n^2)$ signature generations, for trees with bounded out-degree the number of signatures is linear in n. Our construction also permits incremental signing of trees, i.e., if a leaf is added to a signed tree, the signatures on the remaining tree can be re-used and the new signature generated in $\mathcal{O}(n)$ time. We leave it as an open problem to find schemes with better efficiency, still meeting our security notions.

Related Work. Deleting parts of a document while maintaining the integrity and authenticity of the remaining data is an issue that has been approached under different setup assumptions and goals. There have been various approaches to designing redactable signatures [9,10,8], where only linearly ordered documents and the deletion of substrings are considered. Still, they do not require hiding the amount of data removed from the document, i.e., one is able to derive the lengths of the removed strings, or where these were removed from. The former

aspect has been addressed in [4] and [3], where the privacy requirement also includes the length of the hidden portions. A solution furthermore hiding their positions is sketched in [4] and this idea is also used as a building block in our construction for tree.

Further works by Ateniese et al. [1] and the extension to sanitizable signatures due to Brzuska et al. [2], where one can modify authenticated data in a controlled way, are influential to our security models. However, sanitization in such contexts requires the input of a secret key, whereas we allow for data to be manipulated by public means. Moreover, sanitizable signatures usually do not hide the amount of sanitized data.

Organization. The paper is organized as follows. After introducing the necessary notation in Section 2 we formally define the functionality of structural signatures for trees in Section 3. We discuss several formal security notions together with their relations in Section 4, and propose a provably secure construction in Section 5. Finally, we show in Appendix A that the scheme by Kundu and Bertino [6] does not achieve our notion of security.

2 Preliminaries

Trees. A *tree* T is a connected graph $G = (V_T, E_T)$ which consists of a nonempty finite set $V_T = \{v_1, \ldots, v_r\}$ of vertices, a set $E_T = \{e_1, \ldots, e_s\}$ of edges and does not contain cycles. We simply write V (resp. E) instead of V_T (resp. E_T) if the context is clear. Edges are denoted $e = (v_i, v_j) \in V \times V$. A tree T_ρ is *rooted* if one vertex $\rho \in V$ (the *root*) is distinguished from the others. The path-distance from node $v \in V$ to the root node ρ is called the *depth* of v. If $e = (v_i, v_j)$ is an edge, then the node that is closer to ρ is called the *parent* of the other node, while the latter is called a *child* of the former. If two vertices have the same parent, then these two vertices are called *siblings*. A *leaf* L is a vertex with no children. The root is the only node without parents. If the children of each vertex in T_ρ are ordered in respect to some linear order relation, then the tree is called *ordered*. Since this paper only concerns trees that are both rooted and ordered, we consider in the following all trees as rooted and ordered. We further assume that all edges $e = (v_i, v_j)$ are directed away from the root, i.e. v_i is parent of v_j. If two trees T and T' are isomorphic (where the isomorphism also maintains the root and the node order), we write $T \simeq T'$ (or $T = T'$). By $T\backslash L$, we denote the tree resulted after cutting leaf L from T; thus the vertex and edge sets of $T\backslash L$ are $V_T\backslash\{L\}$ and $\{(v_i, v_j) \in E_T \mid v_j \neq L\}$. Furthermore, we write $T' \prec T$ for trees T and T' if either $T' \simeq T\backslash L$, or $T' \prec (T\backslash L)$ for some leaf L of T. Consequently, we denote by $T' \preceq T$ the case where $T' \prec T$ or $T' \simeq T$. Note that writing $T' \preceq T$ means saying that T' is a rooted ordered subtree of T with the same root.

Signature Schemes. A signature scheme DS is a tuple $(\mathsf{Kg}, \mathsf{Sign}, \mathsf{Vf})$ of efficient algorithms, where the key generation algorithm $\mathsf{Kg}(1^\lambda)$ returns a key pair (sk, pk);

the signing algorithm $\mathsf{Sign}(sk, m)$ takes as input a signing key sk and a message $m \in \{0,1\}^\lambda$, and returns a signature σ; and the verification algorithm $\mathsf{Vf}(pk, m, \sigma)$ takes public key pk, message m and signature σ, and returns 0 or 1. We assume that the signature scheme is complete, i.e. for any $(sk, pk) \leftarrow \mathsf{Kg}(1^\lambda)$, any message $m \in \{0,1\}^\lambda$, and any $\sigma \leftarrow \mathsf{Sign}(sk, m)$, we have: $\mathsf{Vf}(pk, m, \sigma) = 1$. Note that it is always possible to sign messages of arbitrary length by applying a collision-resistant hash function $h : \{0,1\}^* \mapsto \{0,1\}^\lambda$ to the message prior to signing. The security of signature schemes $(\mathsf{Kg}, \mathsf{Sign}, \mathsf{Vf})$ is defined following [5], as usual. In this model, an adversary may adaptively invoke a signing oracle and is successful if it manages to compute a signature on a *new* message.

Definition 1 (Unforgeability). *A signature scheme* DS *is* unforgeable under adaptive chosen message attacks *(EU-CMA) if for any efficient algorithm* $\mathcal{A}$ *the probability that the experiment* $\mathsf{Forge}_{\mathcal{A}}^{\mathsf{DS}}$ *evaluates to 1 is negligible (as a function of* λ*), where*

Experiment $\mathsf{Forge}_{\mathcal{A}}^{\mathsf{DS}}(\lambda)$
$(sk, pk) \leftarrow \mathsf{Kg}(1^\lambda)$
$(m^*, \sigma^*) \leftarrow \mathcal{A}^{\mathsf{Sign}(sk,\cdot)}(pk)$
Return 1 *iff* $\mathsf{Vf}(pk, m^*, \sigma^*) = 1$ *and* $\mathcal{A}$ *has never queried* $\mathsf{Sign}(sk, \cdot)$ *on* m^*.

The probability is taken over all coin tosses of Kg, Sign, *and* $\mathcal{A}$.

3 Structural Signatures for Trees

Kundu and Bertino proposed in [6] special signatures for trees, where parts of the tree can be cut off without invalidating the signature on the rest of the tree and without having to re-sign using the private key. To make formal security claims, we first formally define structural signature schemes for trees. These schemes sign trees and also support one public operation on signed trees: any user may remove parts of the tree and derive a signature for the pruned tree without access to the private key. We define such schemes for the operation of cutting single leaves only; iterating the cutting operation then allows for the removal entire subtrees.

Definition 2 (Structural Signature Scheme for Trees). *A structural signature scheme for trees* $\mathsf{strucSig}$ *consists of four efficient algorithms* $(\mathsf{sKg}, \mathsf{sSign}, \mathsf{sVf}, \mathsf{sCut})$ *such that:*

Key Generation. *The key generation algorithm* $\mathsf{sKg}(1^\lambda)$ *outputs a private key sk and a corresponding public key pk:*

$$(sk, pk) \leftarrow \mathsf{sKg}(1^\lambda).$$

Signing. *Algorithm* $\mathsf{sSign}(sk, T)$ *takes as input a secret key sk and a tree* T*. It outputs a structural signature* σ *(with* $T' = T$*):*

$$(T', \sigma) \leftarrow \mathsf{sSign}(sk, T).$$

VERIFICATION. *The verification algorithm* sVf *outputs a bit* $d \in \{0,1\}$ *verifying that* σ *is a valid structural signature on a tree* T *with respect to a public key* pk*:*

$$d \leftarrow \mathsf{sVf}(pk, T, \sigma).$$

CUTTING. *The input of the algorithm* $\mathsf{sCut}(pk, T, \sigma, L)$ *is a public key* pk*, a tree* T*, a signature* σ*, as well as a leaf* L *of* T*. It returns the tree* $T' = T \backslash L$ *and a signature* σ'*:*

$$(T', \sigma') \leftarrow \mathsf{sCut}(pk, T, \sigma, L).$$

We say that a structural signature scheme is correct if:

SIGNING CORRECTNESS. For any $\lambda \in \mathbb{N}$, any key pair $(sk, pk) \leftarrow \mathsf{sKg}(1^\lambda)$, any tree T, and any $(T', \sigma) \leftarrow \mathsf{sSign}(sk, T)$ we have $\mathsf{sVf}(pk, T, \sigma) = 1$.

CUTTING CORRECTNESS. For any $\lambda \in \mathbb{N}$, any key pair $(sk, pk) \leftarrow \mathsf{sKg}(1^\lambda)$, any tree T, any σ with $\mathsf{sVf}(pk, T, \sigma) = 1$, any leaf L of T, and any pair $(T', \sigma') \leftarrow \mathsf{sCut}(pk, T, \sigma, L)$, we require $\mathsf{sVf}(pk, T', \sigma') = 1$.

Again note that iterative leaf-cutting results in the removal of entire subtrees. It is obvious that any subtree T' of T which can be generated by successive executions of sCut satisfies $T' \preceq T$, and vice versa.

Note that our cutting algorithm relies only on the public key of the signer. In the medical example above, this allows the database to generate authentic tree parts without accessing the private key of the medical personnel.

4 Security of Structural Signature

We define in this section the security properties of structural signature schemes via unforgeability, privacy, and transparency. Informally, these security requirements state:

UNFORGEABILITY. No one should be able to compute a valid signature on a tree without having access to the secret key. That is, even if an outsider can request signatures on different trees, it remains impossible to forge a signature. This is analogous to the standard unforgeability requirement for signature schemes.

PRIVACY. No one should be able to gain any knowledge about parts cut off the tree from its structural signature without having access to these parts. Our definition is similar to the standard indistinguishability notion for encryption schemes.

TRANSPARENCY. Nobody should be able to decide whether a signature of a tree has been created from scratch, or through an sCut. This means that a party who receives a signed tree cannot tell whether he received a freshly signed tree or a subtree of a signed tree where some parts have already been cut off.

In the following we will define these notions formally. We note that our definitions resemble the ones of Brzuska et al. [2] for sanitizable signatures which, in turn, refine previous notions [1,10] for sanitizable and redactable signatures. Yet, our notion here takes into account the (tree) structure of documents and allows *public* sanitizations.

4.1 Unforgeability

The unforgeability definition for structural signatures is defined analogously to the standard security requirement for signature schemes. Informally, it states that no one should be able to compute a valid signature σ on a tree T without having access to the secret key sk. This condition must hold even if the adversary can request signatures on q (possibly adaptively chosen) other trees. The forgery must be non-trivial in the sense that it is not the result of a sequence of cutting operations on a tree for which the adversary has previously requested a signature (recall that the cut algorithm operates on public data only).

Definition 3 (Unforgeability). *A structural signature scheme* $\mathsf{strucSig} = (\mathsf{sKg}, \mathsf{sSign}, \mathsf{sVf}, \mathsf{sCut})$ *is* unforgeable under adaptively chosen tree attacks *if for any efficient algorithm* $\mathcal{A}$ *the probability that the experiment* $\mathsf{Unforgeability}_{\mathcal{A}}^{\mathsf{strucSig}}$ *evaluates to 1 is negligible (as a function of* λ*), where*

Experiment $\mathsf{Unforgeability}_{\mathcal{A}}^{\mathsf{strucSig}}(\lambda)$
$(pk, sk) \leftarrow \mathsf{sKg}(1^{\lambda})$
$(T, \sigma) \leftarrow \mathcal{A}^{\mathsf{sSign}(sk,\cdot)}(pk)$
for $i = 1, 2, \ldots, q$*, denote by* T_i *resp.* σ_i *the queries to, resp. answers from, the oracle* sSign
return 1 *iff*
$\mathsf{sVf}(pk, T, \sigma) = 1$ *and*
for all $i = 1, 2, \ldots, q$ *we have* $T \not\preceq T_i$

The probability is taken over all coin tosses of $\mathsf{sKg}, \mathsf{sSign}$*, and* $\mathcal{A}$*.*

4.2 Hiding Properties

Preventing leakage of information roughly means that it should be infeasible for anybody to recover further information on the cut parts of the tree from the structural signature. Here we propose two different notions, the first definition being weaker than the second one. Intuitively, the first notion hides the contents of the cut parts, but not necessarily the cut operations themselves, whereas our stronger notion also hides whether cutting operations have been performed.

Privacy. A basic requirement in the medical data example is that a party cannot gain any information on the parts of the tree that were cut off. This is formalized by demanding that, given a subtree with a signature and two possible source trees, one cannot decide from which source tree the subtree stems from. Intuitively, it follows that one cannot derive any information about the cut parts.

The definition of privacy for structural signatures is based on the indistinguishability definition for encryption schemes: an adversary $\mathcal{A}$ can choose two pairs $(T_0, L_0), (T_1, L_1)$ of trees and leaves such that $T_0 \backslash L_0 \simeq T_1 \backslash L_1$, i.e., removal of the leaves results in isomorphic trees. Furthermore, $\mathcal{A}$ has access to a *left-or-right* oracle, which, given those two trees, consistently either returns a

cut signature for the left pair ($b = 0$) or for the right pair ($b = 1$). The scheme offers privacy, if no adversary can decide whether the oracle returns the left or the right cut tree.

Definition 4 (Privacy). *A structural signature scheme* $\mathsf{strucSig} = (\mathsf{sKg}, \mathsf{sSign}, \mathsf{sVf}, \mathsf{sCut})$ *is* private *if for any efficient algorithm* $\mathcal{A}$ *the probability that the experiment* $\mathsf{LeakPriv}^{\mathsf{strucSig}}_{\mathcal{A}}$ *evaluates to 1 is negligibly close to* $1/2$ *(as a function of* λ*), where*

Experiment $\mathsf{LeakPriv}^{\mathsf{strucSig}}_{\mathcal{A}}(\lambda)$	$\mathsf{SignCut}(T_{j,0}, L_{j,0}, T_{j,1}, L_{j,1}, sk, b)$
$(sk, pk) \leftarrow \mathsf{sKg}(1^\lambda)$	*if* $T_{j,0} \backslash L_{j,0} \not\simeq T_{j,1} \backslash L_{j,1}$ *abort*
$b \leftarrow \{0,1\}$	$(T_{j,b}, \sigma_{j,b}) \leftarrow \mathsf{sSign}(sk, T_{j,b})$
$d \leftarrow \mathcal{A}^{\mathsf{sSign}(sk,\cdot),\mathsf{SignCut}(\cdot,\cdot,\cdot,\cdot,sk,b)}(pk)$	*return* $(T'_b, \sigma'_b) \leftarrow \mathsf{sCut}(pk, T_{j,b}, \sigma_{j,b}, L_{j,b})$
return 1 *if* $d = b$.	

The probability is taken over all coin tosses of $b, \mathsf{sKg}, \mathsf{sSign}, \mathsf{SignCut}$ *and* $\mathcal{A}$*. (Note that for trees the graph isomorphism problem can be decided in polynomial time.)*

Note that, similar to the case of encryption, a hybrid argument shows that allowing the adversary to perform multiple cutting operations per oracle call is equivalent to the case in which only a single cutting operation is performed.

Transparency. The above notion of privacy does not prevent the following information leakage in the medical example: a party may learn that data about the patient's psychological treatment has been deleted from his subtree, although he cannot deduce the actual data. To capture a stronger notion of leakage prevention, we present a definition which not only protects the structure of the tree, but also the operations that may have been performed on it. Intuitively, an adversary should be unable to decide whether he is given a signed tree whose signature has been derived by an sCut operation, or a freshly signed tree. Let $(T', \sigma_{T'}) \leftarrow \mathsf{sCut}(pk, T, \sigma, L)$ be a signed tree derived from the signed tree (T, σ) by application of the leaf-cutting algorithm, and let $(T', \sigma_S) \leftarrow \mathsf{sSign}(sk, T')$ be a signature of T', generated from scratch (without doing any leaf-cutting). The task for the adversary is to distinguish both cases.

Definition 5 (Transparency). *A structural signature scheme* $\mathsf{strucSig} = (\mathsf{sKg}, \mathsf{sSign}, \mathsf{sVf}, \mathsf{sCut})$ *is* transparent *if for any efficient algorithm* $\mathcal{A}$ *the probability that the experiment* $\mathsf{LeakTrans}^{\mathsf{strucSig}}_{\mathcal{A}}$ *evaluates to 1 is negligibly close to* $1/2$ *(as a function of* λ*), where*

Experiment $\mathsf{LeakTrans}^{\mathsf{strucSig}}_{\mathcal{A}}(\lambda)$	$\mathsf{SignOrCut}(T, L, sk, b)$
$(sk, pk) \leftarrow \mathsf{sKg}(1^\lambda)$	*if* $b = 0$:
$b \leftarrow \{0,1\}$	$(T, \sigma) \leftarrow \mathsf{sSign}(sk, T)$
$d \leftarrow \mathcal{A}^{\mathsf{sSign}(sk,\cdot),\mathsf{SignOrCut}(\cdot,\cdot,sk,b)}(pk)$	$(T', \sigma') \leftarrow \mathsf{sCut}(pk, T, \sigma, L)$
return 1 *if* $d = b$.	*if* $b = 1$:
	$T' = T \backslash L$
	$(T', \sigma') = \mathsf{sSign}(sk, T')$
	return (T', σ')

The probability is taken over all coin tosses of $b, \mathsf{sKg}, \mathsf{sSign}, \mathsf{SignOrCut}$*, and* $\mathcal{A}$*.*

As for privacy, a hybrid argument again shows that this notion is robust in the sense that it already implies security against adversaries that pass several cut operations in a single oracle call instead of only one.

Note further that it is easy to see that the construction of Kundu and Bertino [6] does not satisfy this strong definition of transparency; for an analysis see Appendix A.

As mentioned, transparency provides strong hiding guarantees and is desirable in many cases. However, for various application examples privacy is in fact sufficient, namely in all cases, where the receiver already expects partly sanitized documents. This is the case for e.g. all anonymization procedures where, a party's data (patient's name) is removed. Therefore, privacy is a sufficient requirement for some applications and by using a private, non-transparent scheme, one thereby gains in efficiency. Thus, both security requirements deserve a formal treatment.

4.3 Relationships of the Security Requirements

In this section we show that transparency is strictly stronger than privacy. We first prove formally that transparency implies privacy. Then we separate the notions by turning a structural signature scheme that offers privacy into one which still has this property, but which violates transparency.

It is clear that unforgeability does not follow from privacy (and thus not from transparency). Take, for example, the trivial scheme which outputs constants as signatures, say, $\sigma = 0$; the cut algorithm for this scheme prunes the three and also outputs $\sigma = 0$. This scheme is clearly transparent, but easily forgeable. Vice versa, it holds that unforgeability implies neither privacy, nor transparency (e.g., take an unforgeable scheme and modify the cut algorithm to append the original tree to the output signature).

Proposition 1 (Transparency⇒Privacy). *Any transparent structural signature scheme is also private.*

Proof. Assume towards contradiction that there exists a transparent structural signature scheme $\mathsf{strucSig}$ which is not private, i.e., there exists an efficient adversary $\mathcal{A}$ that breaks the privacy of $\mathsf{strucSig}$ with non-negligible probability $1/2 + 1/\mathsf{poly}(\lambda)$ for some polynomial $\mathsf{poly}(\lambda)$. We derive a contradiction showing how to construct a successful algorithm $\mathcal{B}$ against transparency. The input of $\mathcal{B}$ is a public key pk. It runs a black-box simulation of $\mathcal{A}$ on input pk and picks a random bit b^*. Whenever $\mathcal{A}$ invokes its signing oracle $\mathsf{strucSig}$ on a tree T and some leaf L, then $\mathcal{B}$ answers this query with its sSign oracle. For every query $(T_0, L_0), (T_1, L_1)$ that $\mathcal{A}$ sends to its $\mathsf{SignCut}$ oracle, $\mathcal{B}$ forwards (T_{b^*}, L_{b^*}) to its external $\mathsf{SignOrCut}$ oracle and sends the answer to $\mathcal{A}$. Eventually, $\mathcal{A}$ stops outputting a decision bit d. Algorithm $\mathcal{B}$ outputs $a^* = 0$ iff $d = b^*$.

For the analysis first observe that $\mathcal{B}$ is efficient because $\mathcal{A}$ runs in polynomial time and handling all queries can also be done efficiently. We now look at the probability of $\mathcal{B}$ being successful:

– Given that $b = 0$, then the $\mathsf{SignOrCut}$ oracle always signs and applies the cutting algorithm afterwards. Thus, the simulation from $\mathcal{A}$'s point of view is identical to the attack against privacy (with random bit $b^* = 0$). Hence,

$$\mathrm{Prob}[\,a^* = 0 \mid b = 0\,] = \mathrm{Prob}[\,\mathcal{A} = b^* \mid b = 0\,] \geq 1/2 + 1/\mathsf{poly}(\lambda).$$

In other words, the probability of success is lower-bounded by $\mathcal{A}$'s success probability.

– Given on the other hand $b = 1$, then $\mathsf{SignOrCut}$ signs the modified tree T' directly. Bit b^* is information theoretically hidden from $\mathcal{A}$. This follows because the privacy experiment demands that the modified trees have to be identical. Thus, the input of the signing algorithm is independent of b^*:

$$\mathrm{Prob}[\,a^* = 1 \mid b = 1\,] = \frac{1}{2}.$$

The overall success probability of $\mathcal{B}$ is now at least

$$\begin{aligned}\mathrm{Prob}[\,\mathcal{B} = b\,] &= \mathrm{Prob}[\,b = 0\,] \cdot \mathrm{Prob}[\,\mathcal{B} = 0 \mid b = 0\,] \\ &\quad + \mathrm{Prob}[\,b = 1\,] \cdot \mathrm{Prob}[\,\mathcal{B} = 1 \mid b = 1\,] \\ &\geq \frac{1}{2} \cdot \left(\frac{1}{2} + \frac{1}{\mathsf{poly}(\lambda)}\right) + \frac{1}{2} \cdot \frac{1}{2} = \frac{1}{2} + \frac{1}{2 \cdot \mathsf{poly}(\lambda)},\end{aligned}$$

which is non-negligibly larger than $1/2$. □

The following proposition separates both notions by showing that not all private structural signature schemes are also transparent:

Proposition 2 (Privacy⇏Transparency). *Suppose that there exists a private structural signature scheme. Then there exists a private scheme which is not transparent.*

Proof. To prove this separation, we modify a structural signature scheme that provides privacy in such a way that it does leak the information what kind of operation has been performed. To do so, we append a bit to the signature indicating whether a sign ($b = 0$) or cut ($b = 1$) operation has been performed.

More precisely, let $\mathsf{strucSig} = (\mathsf{sKg}, \mathsf{sSign}, \mathsf{sVf}, \mathsf{sCut})$ be a secure structural signature scheme that preserves privacy. We then define the scheme $\mathsf{strucSig}' = (\mathsf{sKg}', \mathsf{sSign}', \mathsf{sVf}', \mathsf{sCut}')$ as follows:

$\mathsf{sKg}'(1^\lambda)$
  return $\mathsf{sKg}(1^\lambda)$

$\mathsf{sVf}'(pk, T, \sigma)$
  parses $\sigma = (\sigma' \| b)$
    (with $b \in \{0,1\}$)
  return $\mathsf{sVf}(pk, T, \sigma')$

$\mathsf{sSign}'(sk, T)$
  return $\mathsf{sSign}(sk, T) \| 1$

$\mathsf{sCut}'(pk, T, \sigma, L)$
  parses $\sigma = (\sigma' \| b)$
    (with $b \in \{0,1\}$)
  return $\mathsf{sCut}(pk, T, \sigma', L) \| 0$

It follows easily from the construction that $\mathsf{strucSig}'$ is efficient and preserves privacy. The scheme, however, is clearly not transparent, because the last bit of a signature directly indicates which operation has been performed. The algorithm $\mathcal{A}$ breaking transparency queries its $\mathsf{SignOrCut}$ oracle on an arbitrary tree and some leaf of the tree. It then parses $\sigma = \sigma' \| b$ and outputs $d = b$. Obviously, this attacker breaks transparency with probability 1. □

5 Constructing Secure Structural Signatures

In this section, we present our structural signature scheme for ordered trees which is unforgeable, transparent and private.

5.1 Construction

The idea of our construction is as follows (see Figure 1): We use an ordinary EU-CMA signature scheme to sign all edges in the tree. In order to avoid match-and-mix attacks between several trees, we endow each vertex with a fresh random number (A) and sign for each edge the contents (which could be, in the above-mentioned XML scenario, the XML tags and attributes) of its adjacent vertices (B) together with the associated random numbers.

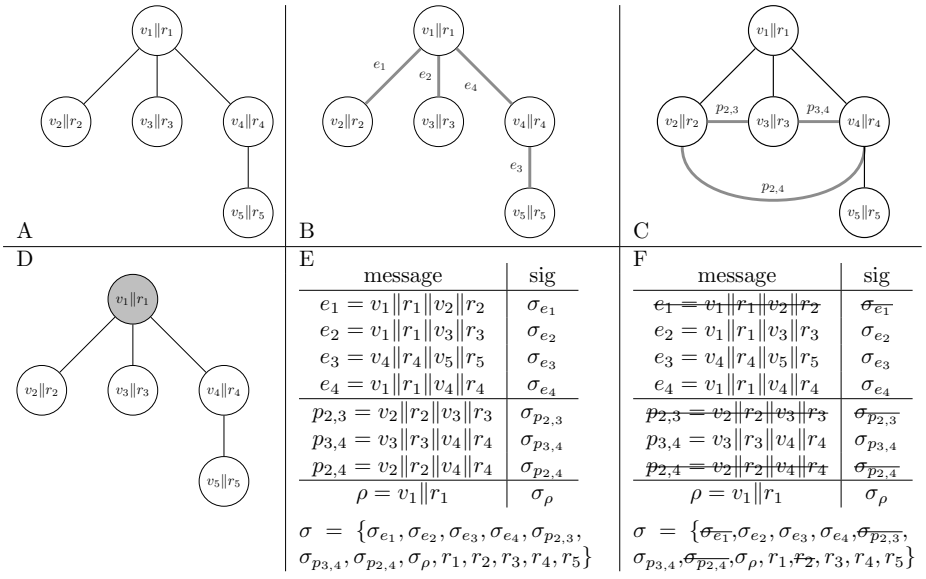

Fig. 1. This figure demonstrates a simple application of the algorithms: (A) randomizing vertices, (B) signing edges, (C) signing order of siblings, (D) signing the root, (E) assembling the final signature, and (F) computing the signature when cutting the leftmost leaf. A more detailed description of the steps is given in Section 5.1.

As we consider ordered trees, we also have to protect the order of siblings of a node; we do this by signing elements of the linear order relation between siblings of a node (C). Finally, the root node and its random value need to be signed (D), as trees containing only a single node do not have any edges. The security of the presented construction relies only on the existence of a standard signature scheme, and no further cryptographic assumptions are required.

We start with defining some further notation. We use the notation v interchangeably to denote a node and its content. Let $\mathcal{P}$ denote the set of all parent nodes having more than one child and let $\mathcal{V}_P = (v_{P,1}, \ldots, v_{P,q})$ be the ordered sequence of child nodes of a node $P \in \mathcal{P}$. We write $\mathcal{R}_P \subseteq \mathcal{V}_P \times \mathcal{V}_P$ for the linear order relation on $\mathcal{V}_P$, i.e., $(v_{P,i}, v_{P,j}) \in \mathcal{R}_P$ if and only if $i < j$. We often denote the elements of $\mathcal{R}_P$ as $J := (v_{P,J_1}, v_{P,J_2})$. Similarly, we write r_{P,J_1} and r_{P,J_2} to denote the random values we will assign to v_{P,J_1} and v_{P,J_2}. Furthermore, we write r_ρ for the randomness associated to the root node.

Let $(\mathsf{Kg}, \mathsf{Sign}, \mathsf{Vf})$ be a signature scheme. We construct a structural signature scheme $\mathsf{strucSig} = (\mathsf{sKg}, \mathsf{sSign}, \mathsf{sVf}, \mathsf{sCut})$ as follows:

KEY GENERATION. On input the security parameter 1^λ, sKg runs the signature scheme's key generation algorithm and outputs $(sk, pk) \leftarrow \mathsf{Kg}(1^\lambda)$.

SIGNING. The signing algorithm works as follows:

$\mathsf{sSign}(sk, T)$:
// T is given as graph $G = (V, E)$
For each vertex $v \in V$:
  $r_v \leftarrow \{0,1\}^\lambda$
$S := \text{""}$
// sign all edges
perform a post-order traversal of the tree:
for each edge $e := (v, w) \in E$ do
  $m_e = v \| r_v \| w \| r_w$
  $\sigma_e \leftarrow \mathsf{Sign}(sk, 0 \| m_e)$
  $S := \sigma_e \| S$
// sign the order of child nodes of a vertex
perform a post-order traversal of the tree:
for each vertex $P \in \mathcal{P}$ and all $J \in \mathcal{R}_P$ do
  $m_{P,J} := v_{P,J_1} \| r_{P,J_1} \| v_{P,J_2} \| r_{P,J_2}$
  $\sigma_{P,J} \leftarrow \mathsf{Sign}(sk, 1 \| m_{P,J})$
  $S := \sigma_{P,J} \| S$
// sign the root node
  $\sigma_\rho \leftarrow \mathsf{Sign}(sk, 2 \| \rho \| r_\rho)$
return $(T, \sigma_\rho \| S \| r_{v_1} \| \ldots \| r_{v_{|V|}})$

VERIFICATION. The verification algorithm works as follows:

$\mathsf{sVf}(pk, T, \sigma)$:
parse σ as $\sigma = \sigma_\rho \| S \| r_{v_1} \| \ldots \| r_{v_{|V|}}$
// check signature on the root node
if $\mathsf{Vf}(pk, 2 \| \rho \| r_\rho, \sigma_\rho) = 0$ *return* 0
// check signatures over the order of child nodes
// $\mathcal{V}_P$ of a parent node P
perform a post-order traversal of the tree:
for each vertex $P \in \mathcal{P}$ in reverse order do
  for all $J \in \mathcal{R}_P$
    parse S as $\sigma_{P,J} \| S'$
    $m_{P,J} := v_{P,J_1} \| r_{P,J_1} \| v_{P,J_2} \| r_{P,J_2}$
    if $\mathsf{Vf}(pk, 1 \| m_{P,J}, \sigma_{P,J}) = 0$ *return* 0
    $S = S'$
// check signature for each edge $e \in E$
perform a post-order traversal of the tree:
for each edge $e = (v, w) \in T$ in reverse order do
  parse S as $\sigma_e \| S'$
  $m_e = v \| r_v \| w \| r_w$
  if $\mathsf{Vf}(pk, 0 \| m_e, \sigma_e) = 0$ *return* 0
  $S = S'$
if $S = \text{""}$ *return* 1 *else return* 0

CUTTING. The cutting algorithm takes a tree T and its valid structural signature σ_T as well as a leaf node $L \in V$ as input. sCut outputs $T' := T \setminus L$ as well as a redacted signature σ'_T, which is constructed from σ_T by removing the signatures σ_e for $e = (P, L) \in E$, $\sigma_{P,J}$ for $J \in \mathcal{R}_P$ with $J = (v, L)$ or $J = (L, v)$ as well as σ_ρ, if $L = \rho$. In addition, r_L is removed from the signature.

It is obvious that the construction provides both signing and cutting correctness, as defined in Section 3.

Efficiency. The complexity of our structural signature scheme is linear in the number of nodes and quadratic in the number of siblings per node. For binary trees, the scheme remains linear in the number of nodes $|V|$, where exactly $\frac{3}{2}(|V|-1)+1$ signature operations are needed; for a tree with a bounded out-degree it also remains linear. We note that the construction in [6] is linear in the number of nodes as well, but is not provably transparent (see Appendix A).

We remark that the idea of signing all pairs of siblings to achieve transparency has been already sketched in [4] *for linear ordered documents*. There, the authors also present a scheme for linear ordered documents with a linear number of signature generations, denoted $\mathcal{RSS}$, which is based on redactable signatures for (non-ordered) sets. If this underlying scheme for sets provides transparency, then so does $\mathcal{RSS}$, and we can then use $\mathcal{RSS}$ in our construction to achieve a transparent scheme with improved efficiency.[1]

5.2 Proof of Security

We show in this section that our construction is unforgeable, transparent, and private.

Theorem 1. *The structural signature scheme* $\mathsf{strucSig} = (\mathsf{sKg}, \mathsf{sSign}, \mathsf{sVf}, \mathsf{sCut})$ *defined above is unforgeable, transparent, and private.*

We prove this theorem via the following propositions.

Proposition 3. *If* $(\mathsf{Kg}, \mathsf{Sign}, \mathsf{Vf})$ *is an unforgeable signature scheme, then the above construction is an unforgeable structural signature scheme.*

Proof. Let $\mathcal{A}$ be a successful adversary against *unforgeability* of structural signatures. Then, we build a successful adversary $\mathcal{B}$ breaking *EU-CMA* of the underlying signature scheme. The simulation works as follows. When $\mathcal{A}$ queries a tree T to its oracle $\mathsf{sSign}(sk, \cdot)$, then $\mathcal{B}$ draws distinct, but random numbers r_v for each vertex $v \in V$, sends the queries $0\|m_e$, $1\|m_{P,J}$, $2\|\rho\|r_\rho$ for $e \in E$, $P \in \mathcal{P}$ and $J \in \mathcal{R}_P$ to its signing oracle $\mathsf{Sign}(sk, \cdot)$, retrieves all signatures, combines them as in the signing algorithm, and returns the tree signature to $\mathcal{B}$. In the end, $\mathcal{A}$ returns (T, σ). We show that if this is a forgery, then a forgery for the underlying signature scheme has occurred. Furthermore, this forgery can be computed efficiently by an extraction algorithm. $\mathcal{B}$ returns the output of the subsequently defined algorithm $\mathsf{Extract}$ to the game. The remaining part of the proof will show that if $\mathcal{A}$ is successful, then so is $\mathcal{B}$.

The crucial idea is to prove that if (T, σ) is a successful forgery, then T contains one of the following elements: a root ρ that was not a root of any previously asked T_i; a signature over an edge m_e not contained in any query tree

[1] The authors in [4] also claim a version of a more efficient scheme, called $\mathcal{SRSS}$, to be transparent, but we were unable to verify this claim.

T_i; or a signature over a siblings' order relation $m_{P,L}$ not contained in any T_i. In the following we provide an extraction algorithm, which extracts such a forgery from the tree signature. We will show afterwards that if (T, σ) is a valid forgery of a tree signature, then one of the three cases must occur and the algorithm successfully outputs a forgery of the underlying signature scheme.

In the algorithm we use the following notation: For $d \in \mathbb{N}^+$, we denote by T_d the tree obtained from T by removing all nodes of depth larger than d; let $V_{d,T}$ be the set of vertices of depth d in tree T and $E_{d,T}$ be the set of edges, such that one node is at depth d and the other, at depth $d-1$. For a node v, denote the edge to its parent node by e_v. Denote by $\mathcal{V}_P^T$ the ordered sequence of child nodes of a single parent node P in tree T, and write $\mathcal{R}_P^T$ for its linear order.

EXTRACTION. The extraction algorithm $\mathsf{Extract}$ on input $(pk, (T, \sigma), (T_1, \sigma_1), ..., (T_n, \sigma_n))$ works as follows:

if $0 \leftarrow \mathsf{sVf}(pk, (T, \sigma))$, *return failure*
else if $\forall i\, \rho_T \neq \rho_{T_i}$, *return* $(2 \| \rho \| r_\rho, \sigma_\rho)$
else if $\rho_T = \rho_{T_i}$, *then* $I := i$
  for d *from* 1 *to depth of* $T_I + 1$ *do*
    if $E_{d,T} \not\subseteq E_{d,T_I}$
    find $e \in E_{d,T} \setminus E_{d,T_I}$ *and return* $(0 \| m_e, \sigma_e)$
    else if $E_{d,T} \subseteq E_{d,T_I}$
      if $\exists P \in V_{d-1,T}$ *such that* $\mathcal{R}_P^T \not\subseteq \mathcal{R}_P^{T_I}$
      find $P \in V_{d-1,T}$ *and* $J \in \mathcal{R}_P^T \setminus \mathcal{R}_P^{T_I}$
      return $(1 \| m_{P,J}, \sigma_{P,J})$
return failure

Lemma 1. *On inputs* $pk, (T_1, \sigma_1), ..., (T_n, \sigma_n)$ *and a valid forgery* (T, σ) *of a tree signature,* $\mathsf{Extract}$ *returns a valid forgery against* EU-CMA *security of* $(\mathsf{Kg}, \mathsf{Sign}, \mathsf{Vf})$.

A proof of the lemma can be found in Appendix B. This proves the proposition. □

Proposition 4. *The structural signature scheme as defined above is transparent.*

Proof. Transparency follows from a simple investigation of distributions: As on identical inputs, the distribution of the output of $\mathsf{SignOrCut}$ with $b = 0$ is identical to the distribution of the outputs of $\mathsf{SignOrCut}$ with $b = 1$, transparency follows. □

Note that transparency even holds for unbounded adversaries such that the redacted data remains confidential information-theoretically.

The following corollary follows directly from Propositions 1 and 4.

Corollary 1. *The structural signature scheme described above is private.*

5.3 Dynamic Update of Signed Trees

Our structural signature scheme for trees allows for the easy addition of new leaf nodes (and consequently new subtrees) by the signer. This update can be performed efficiently in the sense that the signer does not need to refresh any of the existing signatures (on edges and sibling order) that constitute the structural signature. The signer only has to sign new edges and new sibling order relationships resulting from the inclusion of some leaf node (or subtree) into the original tree, and update the structural signature with these signatures. In the following we provide a pseudo-code of the signature update algorithm that adds a new leaf L into some existing tree-signature pair (T, σ_T) and updates the signature. Note that this algorithm can be used to iteratively update T and σ_T with subtrees containing more than one leaf.

$\mathsf{sSignUpd}(sk, T, \sigma_T, P, v_P, L)$:
// T is given as graph $G = (V, E)$, σ_T is the signature of T, P is a node
// in T, v_P is either a child node of P or $\perp$, L is a new leaf node that
// should be inserted in T as a sibling node following v_P or as the new
// first child node of P if $v_P = \perp$
parse σ_T as $\sigma_\rho \| S_s \| S_e \| r_{v_1} \| \ldots \| r_{v_{|V|}}$
// here S_s contains concatenated signatures on the order of siblings in T
// and S_e contains concatenated signatures of edges in T
// for every vertex v_i, r_{v_i} denotes the randomness for that vertex
update $T := (V \cup \{L\}, E \cup \{(P, L)\})$ with L
$r_L \leftarrow \{0,1\}^\lambda$
// sign the order of all siblings of new leaf node L and add them to S_s
update $\mathcal{R}_P$ to $\mathcal{R}_P^*$ using new relationships $\{(L, v_{P,i})\}_i$ and $\{(L, v_{P,j})\}_j$
for all $J \in \mathcal{R}_P^* \setminus \mathcal{R}_P$ do
  $m_{P,J} := v_{P,J_1} \| r_{P,J_1} \| v_{P,J_2} \| r_{P,J_2}$
  $\sigma_{P,J} \leftarrow \mathsf{Sign}(sk, 1 \| m_{P,J})$
  insert $\sigma_{P,J}$ into S_s preserving the post-order of the latter
// sign the edge (P, L)
$m_e = P \| r_P \| L \| r_L$
$\sigma_e \leftarrow \mathsf{Sign}(sk, 0 \| m_e)$
insert σ_e into S_e preserving the post-order of the latter
insert r_L into sequence of random numbers preserving their post-order
denote the re-ordered random values by r'_{v_i} for $i \in \{1, \ldots |V| + 1\}$
return $(T, \sigma_\rho \| S_s \| S_e \| r'_{v_1} \| \ldots \| r'_{v_{|V|+1}})$

We note that the above algorithm preserves the hiding properties (privacy and transparency) of the input tree-signature pairs (T, σ_T). Indeed, each edge and each linear order relation have their own signatures. Similarly to the sCut algorithm, which removes irrelevant signatures from the original set, the sSignUpd updates the set with new signatures. An adversary is thus unable to distinguish whether some σ_T is output by a single execution of sSign or of several consecutive executions of sSignUpd.

Such dynamic updates of signed trees are a very valuable feature of our scheme since in the envisioned applications, such as XML records with medical data, the documents are frequently updated with new diagnoses or treatment procedures.

Acknowledgments

We thank Moti Yung and the anonymous reviewers for valuable comments. Marc Fischlin and Dominique Schröder are supported by the Emmy Noether Program Fi 940/2-1 of the German Research Foundation (DFG). This work was supported by CASED (`http://www.cased.de`).

References

1. Giuseppe Ateniese, Daniel H. Chou, Breno de Medeiros, and Gene Tsudik. *Sanitizable Signatures*. ESORICS, Volume 3679 of Lecture Notes in Computer Science, pages 159–177. Springer, 2005.
2. Christina Brzuska, Marc Fischlin, Tobias Freudenreich, Anja Lehmann, Marcus Page, Jakob Schelbert, Dominique Schröder, and Florian Volk. *Security of Sanitizable Signatures Revisited*. Public Key Cryptography, Volume 5443 of Lecture Notes in Computer Science, pages 317–336. Springer, 2009.
3. Elisa Bertino and A. Kundu. *A New Model for Secure Dissemination of XML Content. Systems, Man, and Cybernetics, Part C: Applications and Reviews, IEEE Transactions on*, 38:292–301, 05 2008.
4. 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/. A preliminary version has appeared at CT-RSA 2009, Lecture Notes in Computer Science, Springer-Verlag.
5. Shafi Goldwasser, Silvio Micali, and Ronald L. Rivest. *A Digital Signature Scheme Secure Against Adaptive Chosen-Message Attacks. SIAM J. Comput.*, 17(2):281–308, 1988.
6. Ashish Kundu and Elisa Bertino. *Structural signatures for tree data structures. Proceedings of the VLDB Endowment*, 1(1):138–150, 2008.
7. Ashish Kundu and Elisa Bertino. *Leakage-Free Integrity Assurance for Tree Data Structures*. Technical Report 2009-1, CERIAS, 2009.
8. Kunihiko Miyazaki, Goichiro Hanaoka, and Hideki Imai. *Invisibly Sanitizable Digital Signature Scheme. IEICE Transactions*, 91-A(1):392–402, 2008.
9. K. Miyazaki, S. Susaki, M. Iwamura, T. Matsumoto, R. Sasaki, and H. Yoshiura. *Digital documents sanitizing problem*. Technical Report ISEC2003-20. IEICE, 2003.
10. Ron Steinfeld, Laurence Bull, and Yuliang Zheng. *Content Extraction Signatures*. ICISC, Volume 2288 of Lecture Notes in Computer Science, pages 285–304. Springer, 2001.

A Randomized Traversal Numbers

Kundu and Bertino [6] use traversal numbers to assign unique numbers to nodes; subsequently, all edges (including the content of adjacent nodes and their unique

numbers) are signed. Consider for example a binary tree with a root v_0 and left child v_l and right child v_r; we will denote the unique numbers assigned to the vertices as r_0, r_l, r_r. Traversing the tree in pre-order we obtain the sequence of associated numbers $(r_0, r_l, r_r) = (1, 2, 3)$. Similarly, if we perform a post-order traversal, we obtain $(r_0, r_l, r_r) = (3, 1, 2)$. Given both pre- and post-order traversal numbers one can reconstruct the tree.

As noted in [6], since the post-order traversal number 2 of the right child v_r in the example above reveals that this node had a sibling, even after v_l has been pruned, such numbers inhibit transparency. Therefore, [6] introduce random traversal numbers which are basically order-preserving random numbers. These numbers remain unchanged during the document's life time. In the full version [7] of the paper the authors outline three implementations of such random traversal numbers:

- Sorted random numbers: Generate sufficiently random numbers, sort them, and assign them to nodes.
- Order-preserving encryption: Assign ordered random numbers to the nodes and apply order-preserving encryption.[2]
- Addition of random numbers: iteratively assign numbers to nodes by taking the previous traversal number and adding a random offset.

We discuss next that none of the methods above yields a transparent solution.[3] Consider again our simple example of a tree with root, left child and right child. Suppose for the moment that the scheme uses post-order traversal. From an abstract point of view, this traversals assign random numbers $(r_l, r_r, r_0) \leftarrow R$ to the nodes v_l, v_r, v_0 (visited in this order) according to some distribution R. This distribution is balanced in the first argument for the examples above, i.e., letting $\mu = \mathbf{E}[r_l]$ be the expected random number assigned to v_l (and assuming for sufficiently large random numbers simply that $\mathrm{Prob}[\, r_l = \mu] = 0$), we have $\mathrm{Prob}[\, r_l \leq \mu] = \mathrm{Prob}[\, r_l \geq \mu] = \frac{1}{2}$. Furthermore, we have $\mathrm{Prob}[\, r_r \geq \mu] \geq \frac{3}{4}$ in the examples above, since r_r is the largest among two random traversal numbers and is thus only smaller than μ if both r_l and r_r are below μ.

We can now break transparency for the simple three-node tree as follows. In the experiment, we either get a structural signature for the tree containing only the nodes v_0 and v_r or a signature for the whole tree, but where v_l has been cut. To solve the challenge, we check the randomized post-order traversal number of the only child v_r and output 0 if $r_r \geq \mu$, and 1 otherwise. We remark that, following Kerckhoff's principle, the expectation μ should be assumed to be known by the adversary (since μ can be derived from the signing algorithm).

As for the analysis note that, if the signer creates a signature by cutting a previously generated signature for the full tree, then r_r is larger than μ with probability $\frac{3}{4}$. If, on the other hand, the pruned tree is signed from scratch,

[2] The description of this step is rather sketchy but all possible interpretations seem to suffer from the same problems discussed below.

[3] Note that transparency, although not defined rigorously, *is* mentioned as a desired security property in [6].

then r_r is distributed as r_v in the original tree and thus $\text{Prob}[\, r_r \geq \mu] = \frac{1}{2}$. It follows that we predict the secret choice b with probability $\frac{5}{8}$, which contradicts Definition 5. Note that the attack even works on a very simple tree and requires only one tree signature.

Note further that [6] also discusses other uniquely determining traversal combinations, like in-order traversal for binary trees together with post-order traversal; the aforementioned attack applies in a similar fashion.

B Proof of Lemma 1

Essentially, the proof is an induction over the depth of the tree, i.e. we show at each level d of the tree, that either we already found a forgery against *EU-CMA* security of $(\mathsf{Kg}, \mathsf{Sign}, \mathsf{Vf})$, or there are still nodes at level $d+1$ in T, i.e. there is at least one node at depth d having at least one child node. The finiteness of the tree will assure that at some point, we find a forgery.

Within the proof, we use the following statement in an essential way: Let T' be a connected subgraph of T containing the distinguished vertex ρ and respecting the sibling order. By a simple induction proof over the number of nodes $|V \setminus V'|$, one can show that starting with T, via successive leaf cutting operations, one can obtain T'. This entails that any such connected subgraph of any of the T_i would not be a valid forgery, as $T \preceq T_i$. Therefore, in the following, we may use $\preceq$ and "subtree of" interchangeably.

We assume that (T, σ) is a valid forgery with respect to queries $(T_i, ..., T_n)$ to sSign, and we show that $\mathsf{Extract}$ extracts a valid forgery against the underlying signature scheme. The proof follows the structure of the $\mathsf{Extract}$ algorithm:

First of all, (T, σ) is a valid forgery which entails that it is validly signed. Therefore, $0 \leftarrow \mathsf{sVf}(pk, (T, \sigma))$ is impossible, and $\mathsf{Extract}$ does not return failure in the first execution step.

If for all $i = 1, ..., n$, $\rho_T \neq \rho_{T_i}$, then $(2 \| \rho \| r_\rho, \sigma_\rho)$ is a valid forgery against the underlying signature scheme, because $\mathsf{Sign}(sk, \cdot)$ queries beginning with 2 are only asked for root nodes, as queries for edges and siblings order relation do not start with 2.

If for some $i = 1, ..., n$, $\rho_T = \rho_{T_i}$ and $r_{\rho_T} = r_{\rho_{T_i}}$, we fix $I := i$ as to adapt the $\mathsf{Extract}$ algorithm notation. Before getting to the induction proof, we introduce some helpful notation first: For $d \in \mathbb{N}$, let T^d denote T cut at depth d, i.e. all nodes having distance strictly greater than d from the root node ρ_T are cut. We show by induction on the depth of the tree T that the following statement is true up to negligible probability: At each level $d \geq 0$, either

(i) we already found a forgery against *EU-CMA* security of $(\mathsf{Kg}, \mathsf{Sign}, \mathsf{Vf})$, or
(ii) T^d is a subtree of T_I.

Note that the latter always entails that the depth of T is strictly greater than d.

Base case. As $\rho_T = \rho_{T_I}$, T^0 is a subtree of T_I. Therefore, the statement is clearly true for $d = 0$.

Induction hypothesis. We assume that at level $d-1$, $1 \leq d$, we already found a forgery against *EU-CMA* security of $(\mathsf{Kg}, \mathsf{Sign}, \mathsf{Vf})$ or T^{d-1} is a subtree of T_I.

Induction step. If we already found a forgery at stage $d-1$ or lower, the statement is trivially true for d. It is thus sufficient to treat the case, where there is no forgery until level $d-1$ and T^{d-1} is a subtree of T_I. As this entails that there is at least one node in T at level $d-1$ having at least one child, there is at least one node in T at level d. We now consider all of these nodes:

1. **New edge.** Assume there is an edge $e \in E_{d,T} \setminus E_{d,T_I}$, which we denote by $(v, w) = e$. We claim that $(0\|v\|r_v\|w\|r_w, \sigma)$ was no output of $\mathsf{Sign}(sk, \cdot)$: First of all, only queries for edges need to be considered, as only those start by 0. Furthermore, only queries for edges in T_I are relevant, as by construction, r_w and r_v are unique and do not appear in other trees except for negligible probability. Therefore $0\|v\|r_v\|w\|r_w$ can only be an edge query for an edge in T_I. Furthermore, r_v and r_w are unique within T_I. If $0\|v\|r_v\|w\|r_w$ had been queried to $\mathsf{Sign}(sk, \cdot)$, this would mean that e is contained in E_{T_I} and because of the randomness' uniqueness within T_I, it follows $e \in E_{d,T_I}$, a contradiction. Thus, $(0\|v\|r_v\|w\|r_w, \sigma)$ is a valid forgery.
2. **Wrong order of siblings.** Now, assume that there is no forgery until level $d-1$ and T^{d-1} is a subtree of T_I and furthermore, $E_{d,T} \subseteq E_{d,T_I}$. For the sake of contradiction, assume that there is a $P \in V_{d-1,T}$ and a $J \in \mathcal{R}_P^T \setminus \mathcal{R}_P^{T_I}$. We claim that $1\|v_{P,J_1}\|r_{P,J_1}\|v_{P,J_2}\|r_{P,J_2}$ had not been asked to $\mathsf{Sign}(sk, \cdot)$: first of all, as in the previous case, only signatures related to T_I need to be considered up to negligible probability. And within T_I, only order relation signatures may have caused such a query to $\mathsf{Sign}(sk, \cdot)$. Furthermore, by uniqueness of the random values r_{P,J_1} and r_{P,J_2} within T_I, such strings may only be signed as order relation strings within $P \in V_{d-1,T}$, whereas $J \in \mathcal{R}_P^T \setminus \mathcal{R}_P^{T_I}$ was assumed. Thus, $(1\|v_{P,J_1}\|r_{P,J_1}\|v_{P,J_2}\|r_{P,J_2}, \sigma_{P,J})$ is a valid forgery against the underlying signature scheme.
3. If T^{d-1} is a subtree of T_I and if at stage d, we neither found a forgery in the two previous cases, then T^d is a subtree of T_I, as all nodes added while getting from T^{d-1} to T^d do exist in the same position and same order in T_I. For a more formal argument, T^{d-1} being a subtree of T_I entails that there is an embedding from T^{d-1} into T_I, and the subset relations $E_{d,T} \subseteq E_{d,T_I}$ and $\mathcal{R}_P^T \subseteq \mathcal{R}_P^{T_I}$ define a unique way to extend this embedding to an embedding from T^d into T_I.

Finally, we argue that at least at one stage d, we find a valid forgery against the underlying signature scheme, if (T, σ) is a valid forgery. T^d cannot be a subtree of T_I at all levels d, as the trees are finite and thus, at some point, $T = T_d$. Thus, $T^d \preceq T_I$ would contradict the assumption that T is a valid forgery. Thus, we are sure to find a forgery at some level, which concludes the proof of the lemma.