Enforcing Policies in Content Manipulation Signature Schemes*

Martin Stanek

Department of Computer Science, Comenius University
Mlynská dolina, 842 48 Bratislava, Slovak Republic. (Email: stanek@dcs.fmph.uniba.sk)

(Received Sept. 26, 2005; revised and accepted Oct. 28, 2005)

Abstract

Recently proposed content manipulation signatures schemes have a broad range of applications, such as authenticated content extraction, adaptive multimedia content delivery, and XML authentication. Most of the constructions are based on Merkle trees. We show that these constructions are unable to enforce manipulation (extraction) policies, i.e. rules defining what manipulations are permitted/prohibited. We propose a generalization of Merkle trees, so called Relaxed Merkle trees. We present their ability to enforce some useful classes of policies, analyze their expressive power, and discuss few enhancements.

Keywords: Merkle tree, policy enforcement, signature schemes

1 Introduction

Merkle tree [3] is a complete labelled binary tree such that the values assigned to the internal nodes are one-way functions of the values assigned to their children. The one-way function is usually instantiated as a standard cryptographic hash function (e.g. SHA-256). The values assigned to the leaves are (depending on application) messages, documents, or fragments of a document.

Merkle trees are a versatile tool for many cryptographic applications. Recently, various constructions employing Merkle trees were proposed, such as content extraction signatures and XML authentication [5], homomorphic signature schemes [1], and systems for authentication of adaptive multimedia content [6]. Computational problems on Merkle trees, such as traversal problem [7], attracted a lot of attention, too.

Our contribution. We show that certain schemes for content manipulation signatures (particularly those studied in [5, 6]) are not able to enforce policies regarding what manipulations of content are permitted/prohibited. We propose an enhanced version of Merkle trees, which we call Relaxed Merkle trees. These trees can easily substitute Merkle trees in above mentioned content manipulation signature schemes. We present examples of some policies of practical significance together with corresponding Relaxed Merkle trees. Finally, we analyze the expressive power of Relaxed Merkle trees.

The paper is structured as follows. Section 2 contains a brief overview of Merkle trees and notation used in the paper. In Section 3 we present two content manipulation signature schemes and show their inability to enforce manipulation policies. Relaxed Merkle trees are introduced in Section 4, together with examples of policies which can be enforced by such trees. Finally, in Section 5, we analyze the expressive power of Relaxed Merkle trees.

2 Preliminaries and Background

Definitions and notation presented in this section are used through the whole paper. Let H be a one-way and collision resistant hash function. Let $(\mathsf{Pk}, \mathsf{Sk})$ denote a key pair of public and private keys for a standard signature scheme (e.g. RSA, DSA). Let $\mathrm{Sig}_{\mathsf{Sk}}(m)$ denote a signature algorithm that outputs a signature σ of message m, and let $\mathrm{Vrf}_{\mathsf{Pk}}(m, \sigma)$ denote a corresponding signature verification algorithm.

Let M be a document consisting of n fragments: $M = \langle m_0, m_1, \ldots, m_{n-1} \rangle$. For convenience, we switch freely between numeric and corresponding binary string representation of indices. For any subdocument $M' \subseteq M$ we denote the set of all indices of fragments in M' by $I_{M'}$, i.e. $M' = \langle m_i \mid i \in I_{M'} \rangle$.

A (strictly) binary tree T is a tree with the property, that each node except leaves has exactly two (left and right) children. By labelling each left edge by 0 and each right edge by 1, all nodes are uniquely identified by "path" strings consisting of digits along the path from root. We denote a node of T by v_w, where w is the path string. Thus, the left and right children of an interior node v_w are v_{w0} and v_{w1}, respectively. The empty string ε identifies the root v_ε of T.

A complete binary tree T is said to have height h if it

*supported by VEGA 1/0131/03

has 2^h leaves and $2^h - 1$ interior nodes.

Merkle Tree. Let us assume that the number of fragments of document M is $n = 2^h$. A Merkle Tree is a complete binary tree of height h with nodes labelled by binary strings. The label assigned to node v_w is denoted by $\mathcal{V}(v_w)$. The labels are computed bottom-up according following rule (the operator "||" denotes a concatenation):

$$\mathcal{V}(v_w) = \begin{cases} H(m_w), & \text{if } v_w \text{ is a leaf } (|w| = h); \\ H(\mathcal{V}(v_{w0}) \,||\, \mathcal{V}(v_{w1})), & \text{otherwise (i.e. } |w| < h). \end{cases}$$

The string $\mathcal{V}(v_\varepsilon)$ assigned to the root represents authentication data for document M. It is signed by document originator in content manipulation signature schemes, see Section 3.

Let $L(v_w)$ be the set of all leaves in a subtree with root v_w. We say the value (label) of node v_w hides leaves $L(v_w)$, since $\mathcal{V}(v_\varepsilon)$ can be computed from $\mathcal{V}(v_w)$ (and other values) without knowing the values of $L(v_w)$.

Let $M' \subseteq \langle m_0, \ldots, m_{n-1}\rangle$ be a subsequence (subdocument) of document M. The set of authentication nodes for M' (or simply authentication set for M'), denoted by $\mathrm{Auth}(M')$, is the minimal set of nodes such that leaves hidden by the values of these nodes correspond to fragments $M \setminus M'$, i.e.

$$\bigcup_{v_w \in \mathrm{Auth}(M')} L(v_w) = \{v_w \mid w \in I_{M \setminus M'}\}.$$

It is easily seen that for any M', the root value $\mathcal{V}(v_\varepsilon)$ can be computed from M' and values (labels) of nodes from $\mathrm{Auth}(M')$. The values of authentication nodes for M' are denoted by $\mathrm{Aval}(M') = \{\mathcal{V}(v_w) \mid v_w \in \mathrm{Auth}(M')\}$.

Example 1. *Let $n = 8$. Let $M' = \langle m_0, m_2, m_3\rangle = \langle m_{000}, m_{010}, m_{011}\rangle$. Then the value $\mathcal{V}(v_{001})$ hides leaf v_{001}, and the value $\mathcal{V}(v_1)$ hides leaves $v_{100}, v_{101}, v_{110}, v_{111}$. Thus,* $\mathrm{Auth}(M') = \{v_{001}, v_1\}$ *and* $\mathrm{Aval}(M') = \{\mathcal{V}(v_{001}), \mathcal{V}(v_1)\}$. *The root value $\mathcal{V}(v_\varepsilon)$ can be computed from M' and* $\mathrm{Aval}(M')$ *as follows:*

$$\mathcal{V}(v_\varepsilon) = H(H(H(H(m_0)||\mathcal{V}(v_{001}))||H(H(m_2)||H(m_3)))||\mathcal{V}(v_1))$$

3 Signatures Based on Merkle Trees

Merkle tree is a versatile tool for signatures schemes allowing certain manipulation of document content. We briefly describe two recently proposed schemes. We show these schemes are unable to enforce manipulation policies.

3.1 Content Extraction Signatures

Steinfeld et al. [5] proposed a content extraction signatures. The originator signs a document M. The scheme allows any untrusted intermediary to produce an extracted signature on selected fragments of M. The extracted signature can be verified by any third party, while hiding the removed parts of the document. Similar scheme was proposed by Johnson et al. [1].

Let Com be a message commitment algorithm satisfying standard cryptographic properties – hiding and binding. Given a message m, $\mathsf{Com}(m, r)$ denotes the commitment to message m under random value r. Let $(\mathsf{Sk}, \mathsf{Pk})$ be a pair of private and public keys of the originator.

Let $M = \langle M_0, \ldots, M_{n-1}\rangle$ be a document. The originator commits to the document fragments: $m_i = \mathsf{Com}(M_i, r_i)$. The originator computes Merkle tree for $\langle m_0, \ldots, m_{n-1}\rangle$. The scheme employs so-called content extraction access structure (CEAS) which the originator uses to specify permitted subdocuments, i.e. subdocuments allowed for extraction. The CEAS is an encoding of the subsets of indices in the original document M. Thus, $M' \subseteq M$ is permitted subdocument if and only if $I_{M'} \in \mathsf{CEAS}$. The signature of M is $\sigma = \mathsf{CEAS}, \sigma_c, r_1, \ldots, r_{n-1}$, where σ_c is the signature of the root value in Merkle tree concatenated with CEAS: $\sigma_c = \mathrm{Sig}_{\mathsf{Sk}}(\mathsf{CEAS} \,||\, \mathcal{V}(v_\varepsilon))$.

The intermediary can produce a signature σ' of extracted document $M' \subseteq M$ from (M, σ) as follows:

$$\sigma' = \mathsf{CEAS}, \sigma_c, \mathrm{Aval}(M'), \langle r_i\rangle_{i \in I_{M'}}. \tag{1}$$

The verification of extracted signature consist of the following steps. First, the verifier computes $m_i = \mathsf{Com}(M_i, r_i)$, for $i \in I_{M'}$. Then (s)he recomputes $\mathcal{V}(v_\varepsilon)$ from m_i's and $\mathrm{Aval}(M')$. Actual verification employs two checks:

$$I_{M'} \in \mathsf{CEAS} \quad \& \quad \mathrm{Vrf}_{\mathsf{Pk}}(\mathsf{CEAS} \,||\, \mathcal{V}(v_\varepsilon), \sigma_c).$$

The problem. Including CEAS enables the verifier to verify whether the extraction policy was followed. On the other hand, the signature created according Equation (1) proves the authenticity of M' regardless CEAS. Although the check $I_{M'} \in \mathsf{CEAS}$ fails, the successful verification of σ_c proves that the originator signed a document containing M'. Thus, the scheme allows to check compliance to the extraction policy, but it does not enforce it. In many scenarios it is desirable that nobody (except the originator) should be able to prove the authenticity of any $M' \subseteq M$ such that $I_{M'} \notin \mathsf{CEAS}$.

Example 2. *Let $n = 4$, and $M = \langle M_0, M_1, M_2, M_3\rangle$. Let $\mathsf{CEAS} = \{\{0, 1\}, \{0, 1, 2\}, \{0, 1, 2, 3\}\}$ reflects the requirement that fragments M_0, M_1 must be contained in every subdocument M', and M_2 must be contained whenever $M_3 \in M'$. Let σ be the signature of M produced by originator according the scheme. The intermediary (anyone) can produce following signature σ^* of subdocument $M^* = \langle M_1, M_2\rangle$:*

$$\sigma' = \mathsf{CEAS}, \sigma_c, \mathrm{Aval}(M^*), \langle r_1, r_2\rangle,$$

where $\mathrm{Aval}(M^*) = \{\mathcal{V}(v_{00}), \mathcal{V}(v_{11})\}$. *Certainly, the first check in verification of extracted signature fails: $I_{M^*} = \{1, 2\} \notin \mathsf{CEAS}$. Nevertheless the second one*

$\mathrm{Vrf}_{Pk}(\mathsf{CEAS} \,||\, \mathcal{V}(v_\varepsilon), \sigma_c)$ *is successful, and it proves the authenticity of* M^**. The verifier knows that* M^* *was not intended to be extracted from the original document, but these fragments were signed by the originator (as a part of* M*). The scheme is unable to enforce* CEAS*. The scheme should prevent producing authenticity proofs for subdocuments prohibited by* CEAS*.*

Remark. Steinfeld et al. [5] proposed some other constructions for content extraction signatures. All of them exhibit the same drawback – the extraction policy is included through CEAS, and cannot be enforced in verification.

3.2 Authentication of Adaptive Multimedia Content

Suzuki et al. [6] proposed a multimedia delivery system that preserves the end-to-end authenticity of original content while allowing content adaptation by intermediaries. The system works on a meta-data level, specifying how media components (files) are handled. Meta-data is provided prior actual content delivery. The scheme assumes three principals – the originator of multimedia content, an intermediary (a proxy) and an end-user. The content (e.g. a sport event or a concert) is modelled as a sequence of fragments. The proxy is allowed to remove fragments of content and to insert its own content (e.g. advertisements) on prescribed positions. The end-user should be able to verify the authenticity of the original content (or remaining portions of it) and the content inserted by the proxy. Moreover, the scheme should prevent undetected removing of proxy's content. As usual, the principals do not trust each other.

The authors solve these requirements by introducing "placeholders" in original content. The originator creates document $M = \langle m_0, \ldots, m_{n-1} \rangle$, where some of the fragments are placeholders for proxy's content. The placeholders contain the public key of the proxy.

The originator signs M as follows: $\sigma = \mathrm{Sig}_{\mathsf{Sk}}(\mathcal{V}(v_\varepsilon))$, where Sk is the originator's private key. Deletion of fragments is guaranteed by the "hiding" property of Merkle tree. The proxy can create signature for $M' \subseteq M$ in the following way: $\sigma' = \sigma, \mathrm{Aval}(M')$. Verification employs reconstruction of $\mathcal{V}(v_\varepsilon)$ from M' and $\mathrm{Aval}(M')$, and checking $\mathrm{Vrf}_{\mathsf{Pk}}(\mathcal{V}(v_\varepsilon), \sigma')$. The proxy inserts its content by attaching and signing it separately. The end-user checks the validity of both signatures. A prevention of malicious deletion of proxy's content (e.g. removing advertisements) is achieved through proxy's signatures of all placeholders. Then, any placeholder without a corresponding signature constitutes evidence that the proxy's content was illegitimately deleted.

We do not consider the attitude "no other party can remove proxy's content without detection" (as stated in [6]) sufficient for such multimedia content delivery system. The problem, just like in previous scheme, lies in the ability to remove any content in such a way, that the authenticity of delivered content can be verified. By enforcing policy we mean that illegitimate deletion of content is not only detectable, but the authenticity of remaining content cannot be verified.

The authors of [6] seem to be aware of the problem with possible content tampering. Therefore, they envision the use of tamper-proof devices together with policy encoding and checking.

Remark. A more efficient variant of the scheme instantiates the placeholders using hash-sign-switch technique [4] based on trapdoor hash function [2]. Details can be found in [6]. Nevertheless the prevention of malicious deletion of proxy's content is the same. Thus, the manipulation policy cannot be enforced.

4 Relaxed Merkle Trees

In order to address policy enforcement in content manipulation signature schemes we propose a generalization of Merkle trees – Relaxed Merkle trees (RMT). The main idea behind RMT is to make $\mathcal{V}(v_\varepsilon)$ directly dependent on some fragments m_i so that the signature verification cannot be performed without them. As we show, RMT offer more that just "these fragments are mandatory" policies.

Definition 1. *A Relaxed Merkle tree* T *based on hash function* H *for document* $M = \langle m_0, \ldots, m_{n-1} \rangle$ *is a strictly binary tree with binary strings assigned to the nodes of* T *in the following way:*

1) *for every fragment* m_i *there exists exactly one leaf* v_w*, such that* $00 \,||\, m_i$ *or* $01 \,||\, H(m_i)$ *is assigned to* v_w *(and for every leaf there is some* m_i*, such that* $00 \,||\, m_i$ *or* $01 \,||\, H(m_i)$ *is assigned to the leaf);*

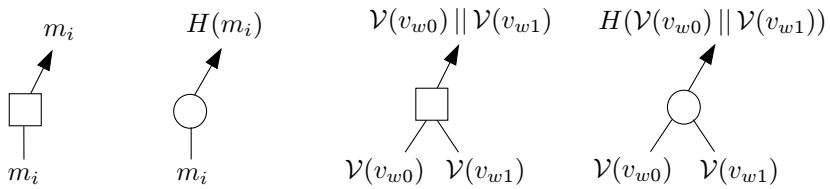
2) *the values assigned to internal nodes are computed either with or without hashing from their children's values as* $\mathcal{V}(v_w) = 11 \,||\, H(\mathcal{V}(v_{w0}) \,||\, \mathcal{V}(v_{w1}))$ *or* $\mathcal{V}(v_w) = 10 \,||\, (\mathcal{V}(v_{w0}) \,||\, \mathcal{V}(v_{w1}))$*;*

3) *the value assigned to the root of* T *is computed as hash of its children's values:* $\mathcal{V}(v_\varepsilon) = H(01 \,||\, (\mathcal{V}(v_0) \,||\, \mathcal{V}(v_1)))$*; the root value for single fragment document is defined as* $\mathcal{V}(v_\varepsilon) = H(00 \,||\, m_0)$*.*

Moreover, the concatenations are unambiguous, i.e. for all binary strings from $x \,||\, y = z \,||\, w$ *follows* $x = z$*, and* $y = w$*; and for any binary strings* $x \,||\, (y \,||\, z) \neq (x \,||\, y) \,||\, z$*.*

The conditions regarding concatenations can be easily met by suitable encoding. The role of various two-bit prefixes is to facilitate proof of collision resistance of RMT (see below).

To simplify our presentation we employ graphical representation of RMT. The nodes of RMT are represented by squares or circles, depending on the way how the node's value is computed (circles denote nodes with hashing, and squares denote nodes without hashing), see Figure 1.



(a) leaves (b) internal nodes

Figure 1: Graphical representation of Relaxed Merkle trees (prefixes are omitted)

Remark. The complexity of $\mathcal{V}(v_\varepsilon)$ computation in RMT is the same as the complexity of computing root value in original Merkle tree, if the hash function H is an iterated hash function. All we need for this computation is to remember intermediate results in recursive (post-order) algorithm.

Collision resistance of RMT. RMT can be viewed as special hash functions. Collision resistance is an important property of RMT, and enables their use in signature schemes. Since we can construct many RMT for particular document, the definition of collision resistance must take into account the tree associated with document. Let $H_T(M)$ be the root value of RMT T for document M.

Definition 2. *RMT are collision resistant if it is hard to find distinct document-tree pairs* $(M,T) \neq (M^*,T^*)$*, such that* $H_T(M) = H_{T^*}(M^*)$.

It can be easily seen that collision resistance of RMT depends on collision resistance of underlying hash function H.

Theorem 1. *Let H be a collision resistant hash function. Then RMT based on this hash function are collision resistant.*

Proof. (Sketch) Let us assume that RMT are not collision resistant. Hence, there are $(M,T) \neq (M^*,T^*)$ such that $H_T(M) = H_{T^*}(M^*)$. Both trees T and T^* must be either single node or multiple nodes. Otherwise, we get a collision in H (because of different prefixes in inputs). Single node trees leads easily to collision in H (when $M \neq M^*$) or equal document-tree pairs (when $M = M^*$) – a contradiction.

For multiple node trees we get $\mathcal{V}(v_\varepsilon) = \mathcal{V}(v^*_\varepsilon)$, i.e. $H(01 \,||\, (\mathcal{V}(v_0) \,||\, \mathcal{V}(v_1))) = H(01 \,||\, (\mathcal{V}(v_0) \,||\, \mathcal{V}(v_1)))$. Assuming collision resistance of H and unambiguous concatenations in RMT we get $\mathcal{V}(v_0) = \mathcal{V}(v^*_0)$ and $\mathcal{V}(v_1) = \mathcal{V}(v^*_1)$. We proceed similarly deeper through trees T and T^*. Since different computation possibilities are encoded by distinct prefixes, we either find a collision in H or conclude that $T = T^*$ and $M = M^*$ – a contradiction. □

Remark. Omitting two-bit prefixes from construction of RMT leads to collisions. For example, let T and T^* be trees with three nodes. All nodes in T are computed with hashing. In T^*, only root and right leaf are computed with hashing.

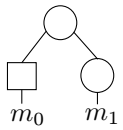


Figure 2: Relaxed Merkle tree T

Thus, $H_T(\langle m_0, m_1\rangle) = H(H(m_0) \,||\, H(m_1))$, and $H_{T^*}(\langle m^*_0, m^*_1\rangle) = H(m^*_0 \,||\, H(m^*_1))$. If $m^*_0 = H(m_0)$ and $m^*_1 = m_1$ we get a collision.

Remark. One-wayness of RMT easily follows from one-way property of underlying hash function H.

Computation of authentication set on RMT. Applications of signature schemes based on Merkle trees need a procedure for computation of the authentication set of some subdocument M', i.e. Auth(M'). Knowing Auth(M') allows direct computation of Aval(M'). An important observation is that for some RMT there are subdocuments $M' \subseteq M$ such that Auth(M') and subsequently Aval(M') do not exist.

Example 3. *Let T be an RMT (depicted in Figure 2) for two fragments document $M = \langle m_0, m_1\rangle$. The root value is $\mathcal{V}(v_\varepsilon) = H(01 \,||\, ((00 \,||\, m_0) \,||\, (01 \,||\, H(m_1))))$. It is easily seen that the authentication set for subdocument $M' = \langle m_1\rangle$ does not exist because the leaf v_0 cannot be hidden by any value – $\mathcal{V}(v_0)$ contains the fragment itself, and $\mathcal{V}(v_\varepsilon)$ hides v_1 as well. Looking at $\mathcal{V}(v_\varepsilon)$ it is obvious that m_0 is directly required in $\mathcal{V}(v_\varepsilon)$ computation, and therefore cannot be omitted in any subdocument. On the other hand, the authentication set for subdocument $M' = \langle m_0\rangle$ exists:* Auth$(M') = \{v_1\}$.

Nonexistence of Auth(M') for some subdocuments M' is the key property that allows enforcing policies in content manipulation signature schemes, see Section 4.1. An algorithm for computing the authentication set (if such set exist) for given tree and subdocument is shown in Appendix.

4.1 Policies

The essential part of content manipulation signature scheme is the ability of untrusted intermediary to compute signature σ' of subdocument $M' \subseteq M$ from the sig-

nature σ of document M. The decision of originator on which subdocuments can be extracted with valid signatures is called policy. The policy can be modelled as a Boolean function $P : \{0,1\}^n \to \{0,1\}$ in the following way: for any $M' \subseteq M$ an intermediary can produce a valid signature σ' of M', if and only if $P(x_{M'}) = 1$. The n-ary vector $x_{M'} = (x_0, \ldots, x_{n-1})$ is assigned to M' in the following way: $x_i = 1 \Leftrightarrow i \in I_{M'}$. The subdocument M' is called permitted, if $P(x_{M'}) = 1$, otherwise it is called prohibited.

We have shown that including policy description into signature (e.g. through CEAS) is not enough, see Section 3.1. We want to make impossible for an intermediary to produce evidence that prohibited subdocument M' was signed by the originator. This is what we refer as "policy enforcement". The idea is to use such RMT for document M such that the authentication set for $M' \subseteq M$ exists if and only if $P(x_{M'}) = 1$. The nonexistence of authentication set for certain subdocument M' forces the intermediary to provide additional fragments from $M \setminus M'$ in order to allow the recomputation of root value $\mathcal{V}(v_\varepsilon)$. Since only for permitted subdocuments the corresponding authentication sets exist, additional fragments together with M' will form a permitted document.

Definition 3. *Let T be an RMT for document $M = \langle m_0, \ldots, m_{n-1}\rangle$, and let $P : \{0,1\}^n \to \{0,1\}$ be a policy. We say that T enforces policy P if for any $M' \subseteq M$: $P(x_{M'}) = 1 \Leftrightarrow \mathrm{Auth}(M')$ exists.*

An RMT uniquely determines an enforced policy. We can replace Merkle trees by RMT in content manipulation signature schemes, such those discussed in Section 3, and enforce policies intended by the originator. Moreover, using CEAS becomes redundant. Certainly, minor changes might be necessary, e.g. including RMT description.

4.2 Examples of Policies

The examples presented in this section show that RMT can enforce some practical policies. Hence, this approach is useful in "real world" scenarios.

There is one to one correspondence between particular policy variable and document's fragment. Therefore, for convenience we express policies, i.e. Boolean functions, in terms of document's fragments.

In order to improve the readability, we will omit the two-bit prefixes in descriptions of RMT nodes' values.

Advertisements. Let M be a document (e.g. a movie or a journal) containing some advertisements. Let $M = \langle m_1, \ldots, m_k, a_1, \ldots, a_t\rangle$ (without any particular order), where m_i's are "data" fragments and a_i's are adverts in document M. The originator intention is to allow those extraction of M, that contain all adverts. The policy for this requirement (the advertisements policy) is

$$P(m_1, \ldots, m_k, a_1, \ldots, a_t) = a_1 \wedge \ldots \wedge a_t.$$

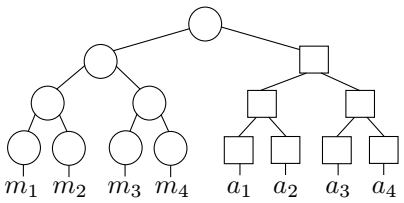


Figure 3: Relaxed Merkle tree for advertisements policy ($k = t = 4$)

Relaxed Merkle tree for this policy is depicted in Figure 3. The left subtree corresponds to the Merkle tree of document $\langle m_1, \ldots, m_k\rangle$. The right subtree is a simple concatenation of adverts $a_1, \ldots, a_t$. In order to verify signature of the root value $\mathcal{V}(v_\varepsilon) = H(\mathcal{V}(v_0) \,||\, a_1 \,||\, \ldots \,||\, a_t)$, a verifier needs to know the values of all adverts. On the other hand, any data fragments can be hidden by values of suitable chosen authentication nodes, see Section 2. Thus, the RMT effectively enforces the advertisements policy.

Interview. Let M be an interview. We can look on interview as a document containing a header h (with information such as place, date, interviewer, interviewee), and a list of question and answer pairs $\langle q_i, a_i\rangle$. Thus, $M = \langle h, q_1, a_1, \ldots, q_k, a_k\rangle$. A natural policy for an interview is to allow precisely those content extractions which contain the header and for every extracted answer, the corresponding question must be extracted as well. Certainly, questions can be extracted without their answers, but header must be always included. This interview policy is described by the following Boolean function:

$$P(h, q_1, a_1, \ldots, q_k, a_k) = h \wedge (a_1 \Rightarrow q_1) \wedge \ldots \wedge (a_k \Rightarrow q_k).$$

RMT for interview policy is depicted in Figure 4. The extension for any k is straightforward. It is easy to verify that RMT enforces the interview policy. The header must be included in any extraction, since the root value directly depends on it: $\mathcal{V}(v_\varepsilon) = H(h \,||\, \ldots)$. When particular a_i is present in an extraction, the q_i must be present as well, because the value of the q_i's and a_i's parent, i.e. $H(q_i \,||\, H(a_i))$, cannot be computed without q_i. On the other hand, q_i can be extracted without corresponding answer – a_i can be hidden by $H(a_i)$. To conclude, any subset of pairs $\langle q_i, a_i\rangle$ can be hidden by some intermediary nodes, thus allowing their omission from an extraction.

Merkle policy.

Merkle tree is a special case of Relaxed Merkle tree. The policy enforced by Merkle tree is denoted as Merkle policy. For document $M = \langle m_0, \ldots, m_{t-1}\rangle$ this policy can be expressed as Boolean function $P(m_0, \ldots, m_{t-1}) = m_0 \vee \ldots \vee m_{t-1}$. The policy specifies that any extracted subdocument is permitted.

Remark. The examples presented in this subsection silently ignored the problem of empty subdocument, i.e.

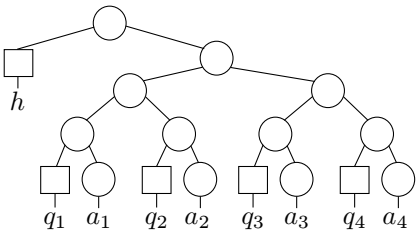


Figure 4: Relaxed Merkle tree for interview policy ($k = 4$)

the value of $P(0, \ldots, 0)$. Formally, any policy P must satisfy the condition $P(0, \ldots, 0) = 1$, see Section 5. On the other hand, we do not care whether an intermediary is able to prove the authenticity of empty subdocument (i.e. authenticity of $\mathcal{V}(v_\varepsilon)$). This is a similar problem as in standard signature schemes (e.g. RSA, ElGamal), where random message forgery can produce a valid signature, but leads to unpredictable hash value of the message (thus not a real forgery, having one-way hash function).

5 On Expressive Power of RMT

Let T be an RMT. The policy enforced by T will be denoted by P_T. Having complete document M, the value of root node, i.e. $\mathcal{V}(v_\varepsilon)$, can be always computed. On the other hand, since $\mathcal{V}(v_\varepsilon)$ is a hash of its children, any intermediary can produce a signature of empty subdocument M' in the following way: $\sigma' = \langle \sigma, \mathcal{V}(v_\varepsilon) \rangle$. Lemma 1 summarizes these observations.

Lemma 1. *Let T be an RMT. Then $P_T(1, \ldots, 1) = 1$, and $P_T(0, \ldots, 0) = 1$.*

5.1 Compositions

Since trees have a recursive structure, a natural problem arises: how to express policy P_T by means of policies P_{T_0} and P_{T_1}, where T_0 and T_1 are left and right subtrees of T, respectively. Such "composition" is useful for computing policy enforced by particular RMT, or producing RMT for given policy (if such RMT exists).

Since subtrees can have root nodes computed without hashing our analysis deals with such "square-rooted" trees as well. Recall, according our graphical notation, we denote nodes computed with (without) hashing as circle (square) nodes.

Let $f(x_1, \ldots, x_n)$ be an n-ary Boolean function. The function obtained from f by setting its output to 1 for all-zero input is denoted by f^+, i.e. $f^+(x_1, \ldots, x_n) = f(x_1, \ldots, x_n) \vee \neg(x_1 \vee \ldots \vee x_n)$.

Following theorems easily follows from the construction of RMT:

Theorem 2. *Let T be a single node RMT, with corresponding fragment x.*

1) Let T be a square node. Then $P_T = x$.

2) Let T be a circle node. Then $P_T = 1$.

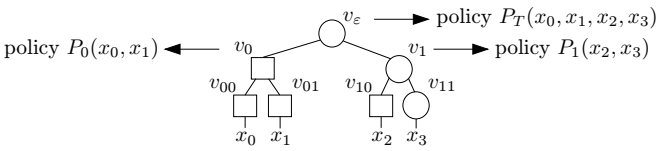


Figure 5: Composition of policies

Theorem 3. *Let T be an RMT with left subtree T_0 and right subtree T_1.*

1) Let the root of T be a square node. Then $P_T = P_{T_0} \wedge P_{T_1}$.

2) Let the root of T be a circle node. Then $P_T = (P_{T_0} \wedge P_{T_1})^+$.

Example 4. *Figure 5 illustrates the composition theorems. From Theorem 2 we get policies for trees with root nodes v_{00}, v_{01}, v_{10}, v_{11}: $P_{v_{00}} = x_0$, $P_{v_{01}} = x_1$, $P_{v_{10}} = x_2$, $P_{v_{11}} = 1$. Applying Theorem 3 we obtain policies for trees with roots v_0, and v_1: $P_0(x_0, x_1) = P_{v_{00}} \wedge P_{v_{01}} = x_0 \wedge x_1$, and $P_1(x_2, x_3) = (P_{v_{10}} \wedge P_{v_{11}})^+ = (x_2 \wedge 1)^+ = (x_3 \Rightarrow x_2)$. Finally, the policy for tree T is: $P_T(x_0, x_1, x_2, x_3) = (P_0 \wedge P_1)^+ = (x_0 \wedge x_1 \wedge (x_3 \Rightarrow x_2))^+$, which is a policy corresponding to CEAS from Example 2, see Section 3.1.*

5.2 Limitations and Extensions

A natural demand is to enable the broadest possible set of policies expressed and enforced by RMT. Starting with documents consisting of two fragments, $M = \langle x_1, x_2 \rangle$, four possible policies follow from Lemma 1: $x_1 \Leftrightarrow x_2$, $x_1 \Rightarrow x_2$, $x_2 \Rightarrow x_1$, and 1. There are four different RMT with two leaves, and they correspond to these policies. Thus, for every two-fragments policy satisfying Lemma 1 there is an RMT which enforces it.

A situation becomes more complicated with three-fragments documents. There are policies that cannot be enforced by any RMT.

Lemma 2. *Let $M = \langle x_1, x_2, x_3 \rangle$ be a document. The policy $P(x_1, x_2, x_3) = (x_1x_2 \vee x_1x_3 \vee x_2x_3)^+$ cannot be enforced by any RMT.*

Proof. Let T be an RMT with three nodes enforcing P. One node must be in depth 1. Since the policy is symmetric, we can assume without loss of generality that the fragment assigned to this leaf is x_1. Let T' denotes the left or right subtree of T with two leaves. If the leaf for x_1 is a square node, from Theorems 2 and 3 we get $P_T = (x_1 \wedge P_{T'})^+$. Then $P_T(0, 1, 1) \neq P(0, 1, 1)$, a contradiction. On the other hand, if the leaf for x_1 is a circle node, we get $P_T = (1 \wedge P_{T'})^+ = P^+_{T'}$. Then $P_T(0, 1, 0) = P_T(1, 1, 0)$, so $P_T \neq P$ – a contradiction. □

A way to overcome the limitations of RMT are extensions enhancing the structure of RMT. First idea is to

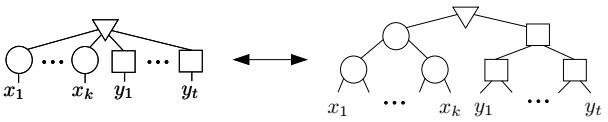


Figure 6: Equivalence of trees with multiple children and RMT

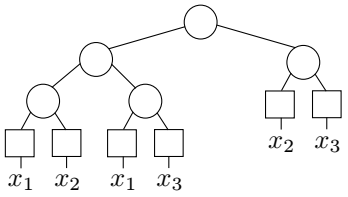


Figure 7: Tree for policy $P(x_1, x_2, x_3) = (x_1x_2 \vee x_1x_3 \vee x_2x_3)^+$

enable more than two child nodes for one parent node. The value assigned to a node is a hash of concatenation of children's values (a circle node) or simple concatenation without hashing (a square node). It is easy to verify, that this extension is just "syntactical sugar" – equivalent representation can be found on RMT, see Figure 6 (a triangle node represents any node type, i.e. either circle or square).

Second, more powerful idea is to enable multiple assignments of fragments to the leaves. Indeed, this leads to trees that enforce policies, the RMT are unable to enforce. Let us take the example from Lemma 2: $P(x_1, x_2, x_3) = (x_1x_2 \vee x_1x_3 \vee x_2x_3)^+$. The tree depicted in Figure 7 enforces exactly policy P.

6 Conclusion

We presented a generalized version of Merkle trees – Relaxed Merkle trees. This generalization allows to enforce policies in content manipulation signature schemes. We analyzed expressive power of RMT and show their limitation and possible extensions.

References

[1] R. Johnson, D. Molnar, D. Song, and D. Wagner, "Homomorphic signatures scheme," in *Proceedings of the RSA Security Conference, Cryptographers Track*, LNCS 2271, pp. 244-262, Springer-Verlag, 2003.

[2] H. Krawczyk and T. Rabin, "Chameleon hashing and signatures," in *Proceedings of NDSS 2000*, pp. 143-154, 2000.

[3] R. Merkle, "A digital signature based on conventional encryption function," in *CRYPTO'87*, LNCS 293, pp. 369-378, Springer-Verlag, 1987.

[4] A. Shamir and Y. Tauman, "Improved online/offline signature schemes," in *CRYPTO 2001*, LNCS 2139, pp. 355-367, Springer-Verlag, 2001.

ComputeAuth
input: T, M'
output: Auth(M') (if it exists)

$(A_\varepsilon, N_\varepsilon) \leftarrow$ ComputeSets(T)
if $N_\varepsilon = M'$ **then** return A_ε
else return "Auth(M') does not exist"

Figure 8: Algorithm ComputeAuth

ComputeSets(S)

if S is a leaf **then**
  if S is a "circle" node $\wedge$ fr(S) $\notin M'$
  then return ($\{S\}$, $\emptyset$)
  else return ($\emptyset$, $\{\text{fr}(S)\}$)
else
  $(A_0, N_0) \leftarrow$ ComputeSets(left subtree of S)
  $(A_1, N_1) \leftarrow$ ComputeSets(right subtree of S)
  if $v_\varepsilon(S)$ is "circle" node $\wedge$ $(N_0 \cup N_1) \setminus M' \neq \emptyset$
  then return ($\{v_\varepsilon(S)\}$, $\emptyset$)
  else return ($A_0 \cup A_1$, $N_0 \cup N_1$)

Figure 9: Function ComputeSets

[5] R. Steinfeld, L. Bull, and Y. Zheng, "Content extraction signatures," in *Proceedings of the 4th International Conference on Information Security and Cryptology – ICISC 2001*, LNCS 2288, pp. 285-304, Springer-Verlag, 2001.
(updated version: IACR ePrint 2002/016)

[6] T. Suzuki, Z. Ramzan, H. Fujimoto, C. Gentry, T. Nakayama, and R. Jain, "A system for end-to-end authentication of adaptive multimedia content," in *Proceedings of the Communication and Multimedia Security – CMS '04*, IFIP vol. 175, Springer-Verlag, 2005.

[7] M. Szydlo, "Merkle tree traversal in log space and time," in *EUROCRYPT 2004*, LNCS 3027, pp. 541-554, Springer-Verlag, 2004.

Appendix: Algorithm for Computing Auth(M') on RMT

Let T be an RMT for document $M = \langle m_0, \ldots, m_{n-1} \rangle$. Let M' be a subdocument of M. The algorithm ComputeAuth computes Auth(M') for given T and M' in time $O(n)$. Since the authentication set for some trees and subdocuments does not exist, the algorithm tests this situation as well.

The algorithm recursively computes sets A_w and N_w (function ComputeSets). Set A_w is the set of all nodes that hide all fragments absent from M' in subtree with root v_w. Set N_w is the set of all fragments not hidden by A_w in subtree with root v_w.

Having computed A_ε and N_ε, the authentication set for M' exists if and only if $N_\varepsilon = M'$. Then, Auth(M') = A_ε.

A description of function ComputeSets, see Figure 9, requires following notations. Let $v_\varepsilon(S)$ denote a root node of a tree S. Let $\mathrm{fr}(v)$ be a fragment assigned to leaf v, i.e. $\mathrm{fr}(v) = m_i$ if either $00 \,||\, m_i$ or $01 \,||\, H(m_i)$ is assigned to v.

Martin Stanek received his Ph.D. in Computer Science from Comenius University. He is currently a teaching assistant of Department of Computer Science, Faculty of Mathematics, Physics and Informatics, Comenius University. His current research interests include cryptography and information security.