Tatra Mt. Math. Publ. **37** (2007), 35–48

TATRA MOUNTAINS
Mathematical Publications

GENERALIZED MERKLE TREES AND THEIR APPLICATIONS

Ľubica Staneková — Martin Stanek

ABSTRACT. Merkle trees [R. Merkle: *A digital signature based on conventional encryption function*. In: Advances in Cryptology–CRYPTO '87 (C. Pomerance, ed.), Lect. Notes Comput. Sci., Vol. 293, Springer-Verlag, Berlin, 1988, pp. 369–378] are used in many cryptographic applications. We introduce and analyze Generalized Merkle trees–a natural enhancement of Merkle trees. These trees enable to enforce policies, i.e., rules specifying which combinations of leaves of a tree can be authenticated. Such policy enforcement has direct application to signatures schemes based on Merkle trees. It allows a better control of permitted/prohibited manipulations of a signed document. As a consequence, we can eliminate the so-called content extraction access structure in content extraction signature scheme by [R. Steinfeld, L. Bull, Y. Zheng: Content extraction signatures. In: Proceedings of the 4th International Conference on Information Security and Cryptology–ICISC '01 (K. Kwangjo, ed.), Lect. Notes Comput. Sci., Vol. 2288, Springer-Verlag, Berlin, 2001, pp. 285–304. (updated version: IACR ePrint 2002/016)], and offer better protection of illegitimate content modification in adaptive multimedia content delivery system by [T. Suzuki, Z. Ramzan, H. Fujimoto, C. Gentry, T. Nakayama, R. Jain: *A system for end-to-end authentication of adaptive multimedia content*. In: Proceedings of the Communication and Multimedia Security–CMS '04 (Y. Sawaragi, ed.), Kluwer Academic Publishers, 2004]. We analyze expressive power, and discuss synthesis of Generalized Merkle trees.

1. Introduction

A Merkle tree [3] is a complete 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

2000 Mathematics Subject Classification: 94A60.
Keywords: Merkle trees, hashing, signature schemes.
supported by APVT 023302.
supported by VEGA 1/3106/06.

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. Various constructions employing Merkle trees were proposed, such as content extraction signatures [4], homomorphic signature schemes [1], and systems for authentication of adaptive multimedia content [5]. Computational problems on Merkle trees, such as traversal problem [6, 2], attracted a lot of attention, too.

Our Contribution. We introduce and analyze Generalized Merkle trees––a natural enhancement of Merkle trees. These trees enable to enforce policies, i.e., rules specifying which combinations of leaves of a tree can be authenticated. The policy enforcement allows better control of what manipulations of a document are prohibited. As a consequence, for example, we can eliminate the so-called content extraction access structure in [4], or offer better protection of illegitimate content modification in [5].

The paper is structured as follows. Section 2 contains a brief overview of Merkle trees and notation used in the paper. Generalized Merkle trees are introduced in Section 3. An algorithm for constructing the authentication set is presented in section 4. Section 5 contains thorough analysis of policies enforced on generalized Merkle trees, together with examples of various practical policy enforcements. We integrate generalized Merkle trees with a context extraction signature scheme [4] in Section 6, resulting in a scheme without need of content extraction access structure.

2. Preliminaries

Let H be a one-way and collision resistant hash function. Let M be a document consisting of n fragments: $M = \langle m_0, m_1, \dots, m_{n-1} \rangle$. 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 the 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 has 2^h leaves and $2^h - 1$ interior nodes.

Merkle Tree. Let us assume that the number of fragments of a 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 to the following rule (the operator "$||$" denotes 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 the document M – similar to standard hash. Moreover, the tree structure allows computation of any subdocument's authentication data (see bellow). The root value $\mathcal{V}(v_\varepsilon)$ can be, for example, signed, to form a basis for various content manipulation signature schemes [1, 4, 5].

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 the 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 set of nodes such that leaves hidden by the values of these nodes correspond to fragments in $M \setminus M'$, 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. 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 the leaf v_{001}, and the value $\mathcal{V}(v_1)$ hides the 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:

$$\mathcal{V}(v_\varepsilon) = H\left(H\Big(H\big(H(m_0) \,||\, \mathcal{V}(v_{001})\big) \,||\, H\big(H(m_2) \,||\, H(m_3)\big)\Big) \,||\, \mathcal{V}(v_1)\right).$$

3. Generalized Merkle Trees

In what follows we propose a generalization of Merkle trees that is aimed at reducing the set of subdocuments for which the authentication set $\mathrm{Auth}(M')$ exists. We call such requirements a "policy". Subdocuments without authentication set cannot be authenticated (in a standard way). Thus, the ability to enforce policy on a Merkle tree has a practical impact on content manipulation signature schemes, where some manipulations can be prohibited.

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

(1) for every leaf v_w there is a fragment m_i, such that m_i or $H(m_i)$ is assigned to the leaf (i.e. $\mathcal{V}(v_w) = m_i$ or $\mathcal{V}(v_w) = H(m_i)$);

(2) for every fragment m_i there exists at least one leaf v_w such that m_i or $H(m_i)$ is assigned to v_w;

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

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

Remark. Some applications need unambiguous encoding of GMT, i.e. values computed with hashing are distinguishable from values computed without hashing, values assigned to leaves are distinguishable from values assigned to internal nodes, and a concatenation of strings is uniquely parsable. These conditions can be easily met by a suitable encoding.

Informally, GMT differs from Merkle trees in the following way: the tree is not necessary complete; the values assigned to leaves are plain fragments or their hashes, they are assigned to leaves in any order (multiple assignments are allowed); and the values assigned to interior nodes are computed with or without hashing.

To simplify our presentation we employ a graphical representation of GMT. The nodes of GMT 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). Figure 1 shows an example of GMT, where the root value is computed as $\mathcal{V}(v_\varepsilon) = H\left(H(m_2) \,||\, \big(H(m_0 \,||\, H(m_1)\big) \,||\, m_2)\right)$.

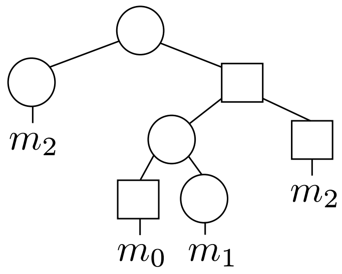

Figure 1. Example of GMT.

The authentication set for $M' \subseteq M$ in GMT is defined similarly to the authentication set on Merkle trees, taking into account possible multiple appearances of fragments in GMT. $\mathrm{Auth}(M')$ is the set of "circle" nodes such that the values of these nodes hide all occurrences of fragments from $M \setminus M'$. Moreover, for every fragment from M' there exists at least one occurrence in the tree that is not hidden (i.e., every fragment from M' can be authenticated).

Using Merkle tree for a document M, any $M' \subseteq M$ has the corresponding authentication set. This is no longer true for GMT. The structure of GMT prohibits the existence of $\mathrm{Auth}(M')$ for some M'.

Example. Let us take the GMT from Figure 1. Let $M' = \langle m_0, m_2 \rangle$, and $M'' = \langle m_1, m_2 \rangle$. It is easy to verify that the authentication set for M' exists, in fact $\mathrm{Auth}(M') = \{v_{101}\}$. On the other hand, the authentication set for M'' does not exist, since the value of v_{10}, i.e., $\mathcal{V}(v_{10}) = H\big(m_0 \,||\, H(m_1)\big)$ cannot be computed without the knowledge of m_0. In other words, the leaf v_{100} (the leaf for m_0 fragment) cannot be hidden by some internal node without simultaneously hiding the leaf v_{101} (the leaf for m_1 fragment).

Definition 2. Let T be a GMT for a document M consisting of n fragments. An n-ary Boolean function P is a policy enforced on T, if for any $M' \subseteq M$ the authentication set $\mathrm{Auth}(M')$ exists if and only if $P(x_{M'}) = 1$. The binary vector $x_{M'} = (x_0, \ldots, x_{n-1})$ is uniquely determined by M': $x_i = 1 \;\Leftrightarrow\; m_i \in M'$.

The policy enforced on GMT T will be denoted by P_T.

To simplify our presentation, policies will be expressed in terms of document's fragments.

Example. For the GMT from Figure 1 we get $P(0,0,0) = 1$, $P(0,0,1) = 1$, $P(1,0,1) = 1$, and $P(1,1,1) = 1$. The value of P is 0 for all other combinations of fragments.

Let $f(x_0, \ldots, x_{n-1})$ 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., $f^+(x_0, \ldots, x_{n-1}) = f(x_0, \ldots, x_{n-1}) \vee \neg(x_0 \vee \ldots \vee x_{n-1})$.

4. Computation of Authentication Set

The ability to compute the authentication set for a subdocument efficiently is a necessary requirement for practical cryptographic constructions based on GMT, see Section 6. Let T be a GMT for a document $M = \langle m_0, \ldots, m_{n-1} \rangle$. Let M' be a subdocument of M. The algorithm ComputeAuth computes $\mathrm{Auth}(M')$ for given T and M' in time $O(n)$, see Figure 2. Since the authentication set for some trees and subdocuments does not exist, the algorithm detects this situation as well.

The algorithm recursively computes the sets A_w and N_w (the function ComputeSets, see Figure 3). The set A_w contains precisely those nodes in the subtree with the root v_w that satisfy the following conditions:

(1) The values of nodes from A_w hide all fragments that absent from M' in the subtree with root v_w.

(2) The values of nodes from A_w hide the minimal set of fragments, i.e. the "hiding" nodes are as deep in the subtree as possible.

Notice that A_w is uniquely determined by T, M', and w. The set N_w contains all fragments not hidden by A_w in the subtree with the root v_w.

Having computed A_ε and N_ε, the following theorem shows the correctness of the algorithm ComputeAuth:

Theorem 1. *Let T be a GMT for document $M = \langle m_0, \ldots, m_{n-1} \rangle$, and let $M' \subseteq M$.*

(1) $\mathrm{Auth}(M')$ *exists if and only if* $N_\varepsilon = M'$*;*

(2) *If* $\mathrm{Auth}(M')$ *exists, then* $\mathrm{Auth}(M') = A_\varepsilon$*.*

Proof. (Sketch) Let us suppose that $\mathrm{Auth}(M')$ exists. The authentication set $\mathrm{Auth}(M')$ satisfies the conditions for A_w. Hence $\mathrm{Auth}(M') = A_w$. Further, each $m_i \in N_\varepsilon$ is not hidden by A_ε, thus $N_\varepsilon \subseteq M'$. Similarly, each $m_i \in M'$ is not hidden by $\mathrm{Auth}(M')$, thus $M' \subseteq N_\varepsilon$. This shows $N_\varepsilon = M'$.

Now, let us suppose that $N_\varepsilon \neq M'$. In case $N_\varepsilon \subsetneq M'$ the set A_ε, in order to hide fragments from $M \setminus M'$ was "forced" to hide more fragments. In case $N_\varepsilon \supsetneq M'$, the set A_ε was unable to hide some fragment from $M \setminus M'$. Hence, in both cases, $\mathrm{Auth}(M')$ does not exist. $\square$

A description of the function ComputeSets, see Figure 3, uses following notation. A fragment assigned (with or without hashing) to a leaf v will be denoted by frag(v). A root of a tree S will be denoted by root(S).

```
ComputeAuth(T, M′)

(A_ε, N_ε) ← ComputeSets(T)
if N_ε = M′ then return A_ε
else return "Auth(M′) does not exist"
```

FIGURE 2. Algorithm ComputeAuth.

```
ComputeSets(S)

if S is a leaf then
    if S is a "circle" node ∧ frag(S) ∉ M′
    then return ({S}, ∅)
    else return (∅, {frag(S)})
else
    (A_0, N_0) ← ComputeSets(left subtree of S)
    (A_1, N_1) ← ComputeSets(right subtree of S)
    if root(S) is "circle" node ∧ (N_0 ∪ N_1) ∖ M′ ≠ ∅
    then return ({root(S)}, ∅)
    else return (A_0 ∪ A_1, N_0 ∪ N_1)
```

FIGURE 3. Function ComputeSets.

5. Policy Enforcement

The relevance of GMT for cryptographic applications depends substantially on their expressive power. Therefore, two natural problems that we address in this section are of importance:

(1) What policies can be enforced on GMT?

(2) Are there any suitable (practical) policies that can be enforced on GMT?

Let T be a GMT. The policy enforced on T will be denoted by P_T. Having complete document M, the authentication set Auth(M) is empty, and the value of the root node can be computed. On the other hand, the node of GMT is a circle node, so $\mathrm{Auth}(\emptyset) = \{v_\varepsilon\}$. Thus, we get a basic property of policies enforced on GMT.

LEMMA 1. *Let T be a GMT. Then $P_T(1,\dots,1) = 1$, and $P(0,\dots,0) = 1$.*

An explicit characterization of policy enforced on some GMT T is a hard task. Certainly, we can construct the policy P_T interactively by running the algorithm ComputeAuth for every subdocument $M' \subseteq M$. This enumeration of P_T requires $O(n2^n)$ operations. A simpler situation arises for GMT without multiple assignments of fragments to leaf nodes, i.e., each fragment is assigned exactly once. For these GMT the enforced policy can be constructed as Boolean formula directly. Lemma 2 and Theorem 2 show such construction of policy, employing a recursive structure of binary tree – the policy is composed from policies enforced on the subtrees of GMT. Since subtrees can have root nodes computed without hashing, our analysis deals with such "square-rooted" trees as well. According to our graphical notation, we denote nodes computed with (without) hashing as circle (square) nodes. Lemma 2 analyzes policies enforced on single node trees.

LEMMA 2. *Let T be a single node GMT, with corresponding fragment x.*

(1) *Let T be a square node, i.e., $\mathcal{V}(T) = x$. Then $P_T = x$.*
(2) *Let T be a circle node, i.e., $\mathcal{V}(T) = H(x)$. Then $P_T = 1$.*

P r o o f. The lemma trivially follows from definition of policy enforced on GMT. □

Theorem 2 shows the construction of P_T from policies enforced on subtrees of T.

THEOREM 2. *Let T be a GMT without multiple occurrences of fragment assignments (i.e., each fragment is assigned exactly once). Let T_0 be a left subtree of T, and T_1 a right subtree of T.*

(1) *Let the root of T be a square node. Then $P_T = P_{T_0} P_{T_1}$.*
(2) *Let the root of T be a circle node. Then $P_T = \left[P_{T_0} P_{T_1}\right]^+$.*

P r o o f. It suffices to show that for arbitrary M' the constructed policy $P_T(x_{M'}) = 1$, if and only if there exists $\mathrm{Auth}(M')$. We prove it by induction on the height of the tree. It can be easily verified that it holds for all GMT of height $h = 1$ (there are only 4 such trees with a circle root, and 4 trees with a square root).

Let T be a tree of height $h + 1$. First, let the root of T be a square node. Since there are no multiple occurrences of fragments, the authentication set $\mathrm{Auth}(M')$ exists if and only if there exist $\mathrm{Auth}(M' \downarrow T_0)$ and $\mathrm{Auth}(M' \downarrow T_1)$

($M' \downarrow T_i$ denoting the subdocument restricted to fragments from the corresponding subtree). By induction hypothesis it is equivalent to $P_{T_0}(x_{M'} \downarrow T_0) = 1$ and $P_{T_1}(x_{M'} \downarrow T_1) = 1$ (similarly, $x_{M'} \downarrow T_i$ denotes the restriction to fragments from the corresponding subtree). Thus, $P_T = P_{T_0} \wedge P_{T_1}$.

In the second case, the root of T is a circle node. Then $\mathrm{Auth}(M')$, if it exists, is either $\mathrm{Auth}(M' \downarrow T_0) \cup \mathrm{Auth}(M' \downarrow T_1)$ or $\mathrm{Auth}(M') = \{v_\varepsilon\}$. Hence the existence of $\mathrm{Auth}(M')$ is equivalent to $\big(P_{T_0}(x_{M'} \downarrow T_0) = 1 \wedge P_{T_1}(x_{M'} \downarrow T_1) = 1\big)$ or $x_{M'} = (0, \ldots, 0)$, i.e. $P_T = \big[P_{T_0} P_{T_1}\big]^+ = 1$. □

A simple corollary for standard Merkle trees (a special case of GMT) follows:

Corollary 1. *Let T be a Merkle tree. Then $P_T = 1$.*

The rest of this section presents examples of various practical policies and the corresponding GMT that enable to enforce these policies. We introduce the advertisements policy, interview policy, majority policy, and Merkle policy.

The examples silently ignore the case of empty subdocument, i.e., the value of $P(0, \ldots, 0)$. According to Lemma 1, $P(0, \ldots, 0) = 1$, i.e., $\mathrm{Auth}(\emptyset)$ always exists. Hence, each policy P discussed in our examples should be expressed as P^+. We use Boolean functions (policies) in form P instead of P^+, because of better readability. Moreover, the existence of the authentication set for empty subdocument ($\mathrm{Auth}(\emptyset) = \{v_\varepsilon\}$) does not lead to violation of any intended policy.

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 M. Our intention is to allow only those subdocuments of M, that contain all adverts. For subdocuments with some missing adverts, it must be impossible to recompute the authentication data – the root value $\mathcal{V}(v_\varepsilon)$. In other words, the authentication sets do not exist for such subdocuments. The policy for this requirement (the advertisements policy) is

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

The Generalized Merkle tree for this policy is depicted in Figure 4. 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 recompute the root value $\mathcal{V}(v_\varepsilon) = H\big(\mathcal{V}(v_0) \,||\, a_1 \,||\, \ldots \,||\, a_t\big)$, one needs to know the values of all adverts. On the other hand, any data fragments can be hidden by values of suitably chosen authentication nodes, see Section 2. Thus, the advertisements policy is effectively enforced on this GMT.

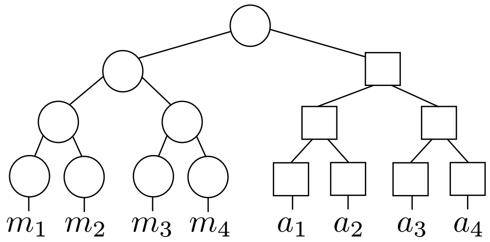

FIGURE 4. GMT for advertisements policy ($k = t = 4$).

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, etc.), and 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 subdocuments which contain the header and for every answer contained in the subdocument, the corresponding question must be contained as well. On the other hand, a question can be contained in a subdocument without corresponding answer. This interview policy is described by the following Boolean function:

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

The Generalized Merkle tree for interview policy is depicted in Figure 5. The extension for any k is straightforward. It is easy to verify that the interview policy is enforced on this GMT. The header must be included in any subdocument, since the root value directly depends on it (the header cannot be hidden): $\mathcal{V}(v_\varepsilon) = H(h \,||\, \ldots)$. When a 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., $H\big(q_i \,||\, H(a_i)\big)$, cannot be computed without q_i. On the other hand, q_i can be selected 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.

Majority policy

Let $M = \langle m_1, \ldots, m_n \rangle$ be a document. Our intention is to allow only those subdocuments that contain majority of fragments, i.e., more than $n/2$ fragments. In other words, unless the majority of fragments is not contained in a subdocument, the authentication set does not exist. A GMT for this policy has multiple assignments of fragments to leaf nodes, and its size grows exponentially (with n). The majority policy for $n = 4$ is expressed by the following Boolean function:

$$P(m_1, m_2, m_3, m_4) = m_1 m_2 m_3 \vee m_1 m_2 m_4 \vee m_1 m_3 m_4 \vee m_2 m_3 m_4.$$

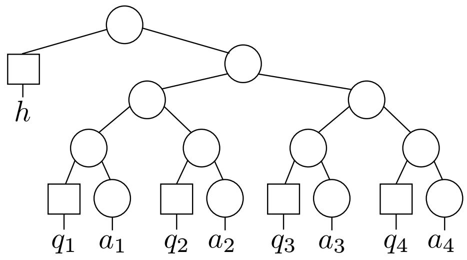

FIGURE 5. GMT for interview policy ($k = 4$).

The Generalized Merkle tree for this policy is depicted in Figure 6. If less than three fragments are present in a subdocument, we must hide at least two fragments. There are four subtrees of our GMT with values $\mathcal{V}(v_{00}) = H(m_1 \,||\, m_2 \,||\, m_3)$, ..., $\mathcal{V}(v_{11}) = H(m_2 \,||\, m_3 \,||\, m_4)$. Let us suppose (without loss of generality) that m_1, m_2 should be hidden. At least one fragment is present in each of four subtrees. Therefore, we are unable to compute $\mathcal{V}(v_{00})$, ..., $\mathcal{V}(v_{11})$ – the fragments in all these subtrees must be hidden. Hence the fragments contained cannot be authenticated, the authentication set does not exit.

On the other hand, it can be easily seen that the authentication set exists for subdocuments with three or four fragments.

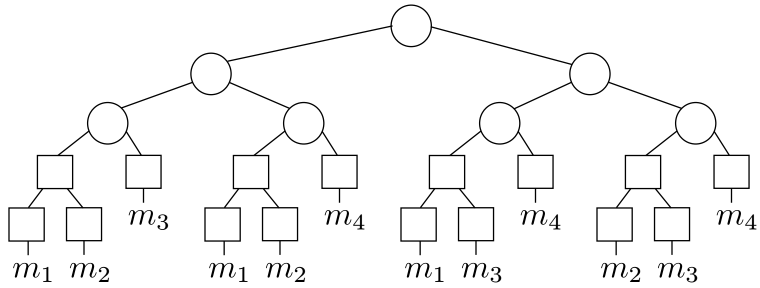

FIGURE 6. GMT for majority policy ($n = 4$).

Merkle policy

Merkle tree is a special case of Generalized Merkle tree. The policy enforced on Merkle tree is denoted as Merkle policy. For document $M = \langle m_0, \ldots, m_{n-1} \rangle$ this policy can be expressed as Boolean function $P(m_0, \ldots, m_{n-1}) = m_0 \vee \ldots \vee m_{n-1}$. The policy specifies that arbitrary subdocuments are permitted – for any $M' \subseteq M$ the $\mathrm{Auth}(M')$ exists.

6. Application – Context Extraction Signatures

We propose a content extraction signature scheme based on GMT. The scheme is a modification of scheme proposed by Steinfeld, Bull and Zheng [4]. The modification replaces the underlying Merkle tree with GMT in the scheme. In consequence, the policy defining which extractions are allowed is enforced in a stronger manner.

The scheme addresses the following scenario. The originator (some authority) signs a document M. The scheme allows any untrustworthy intermediary (e.g., a web server) 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. Certainly, the originator has some requirements on what extractions are allowed/prohibitted for M.

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 $(\mathsf{Pk}, \mathsf{Sk})$ denote a key pair of public and private keys of the originator for a standard signature scheme (e.g., RSA, DSA). Let $\mathrm{Sig}_{\mathsf{Sk}}(m)$ be a signature algorithm that outputs a signature σ of a message m, and let $\mathrm{Vrf}_{\mathsf{Pk}}(m, \sigma)$ be the corresponding signature verification algorithm.

Let $M = \langle M_0, \ldots, M_{n-1} \rangle$ be a document, and T be a GMT for M constructed according to the originator's policy. The originator commits to the document fragments: $m_i = \mathsf{Com}(M_i, r_i)$. The originator computes the root value of T: $\mathcal{V}(v_\varepsilon)$. The signature of M is

$$\sigma = \underbrace{\mathrm{Sig}_{\mathsf{Sk}}\big(\mathcal{V}(v_\varepsilon)\big)}_{\sigma_c}, r_0, \ldots, r_{n-1}.$$

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

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

Recall that $\mathrm{Aval}(M') = \big\{\mathcal{V}(v_w) \mid v_w \in \mathrm{Auth}(M')\big\}$ is the set of authentication nodes values for M'.

The verification of extracted signature consists 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 $\mathrm{Aval}(M')$. The final verification checks the signature of the root value:

$$\mathrm{Vrf}_{\mathsf{Pk}}\big(\mathcal{V}(v_\varepsilon), \sigma_c\big).$$

It can be easily seen that the intermediary cannot produce verifiable extracted signature for prohibited subdocuments (since the authentication sets do not exist for such subdocuments).

Original scheme [4] encodes originator policy as so called Content Extraction Access Structure (CEAS). To protect integrity and authenticity of CEAS, it must be a part signed data. Moreover, it must be transferred to verifier together with extracted signature. Modified scheme removes the usage of CEAS, and the need to encode the policy.

Including CEAS enables the verifier to verify whether the extraction policy was followed (by checking $I_{M'} \in \mathsf{CEAS}$). On the other hand, anyone can compute extracted signature created for arbitrary subdocument M' regardless CEAS (since $\mathrm{Auth}(M')$ can be constructed for any M' on Merkle tree). Although the check $I_{M'} \in \mathsf{CEAS}$ fails, the successful verification of extracted signature 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 $I_{M'} \notin \mathsf{CEAS}$. Our modified scheme with GMT achieves exactly this.

Remark. Suzuki *et al.* proposed a scheme for authentication of adaptive multimedia content based on Merkle trees [5]. Again, replacing Merkle tree with GMT allows better control of prohibited manipulation of a signed content.

REFERENCES

[1] JOHNSON, R.—MOLNAR, D.—SONG, D.—WAGNER, D.: *Homomorphic signatures scheme*. In: Proceedings of the RSA Security Conference, Cryptographers Track (B. Preneel, ed.), Lect. Notes Comput. Sci., Vol. 2271, Springer-Verlag, Berlin, 2003, pp. 244–262.

[2] KARPINSKI, M.—NEKRICH, Y.: A Note on Traversing Skew Merkle Trees, ECCC Report TR04-118, ISSN 1433-8092, 2004.

[3] MERKLE, R.: *A digital signature based on conventional encryption function*. In: Advances in Cryptology–CRYPTO '87 (C. Pomerance, ed.), Lect. Notes Comput. Sci., Vol. 293, Springer-Verlag, Berlin, 1988, pp. 369–378.

[4] STEINFELD, R.—BULL, L.—ZHENG, Y.: Content extraction signatures. In: Proceedings of the 4th International Conference on Information Security and Cryptology–ICISC '01 (K. Kwangjo, ed.), Lect. Notes Comput. Sci., Vol. 2288, Springer-Verlag, Berlin, 2001, pp. 285–304. (updated version: IACR ePrint 2002/016)

[5] SUZUKI, T.—RAMZAN, Z.—FUJIMOTO, H.—GENTRY, C.—NAKAYAMA, T.—JAIN, R.: *A system for end-to-end authentication of adaptive multimedia content*. In: Proceedings of the Communication and Multimedia Security–CMS '04 (Y. Sawaragi, ed.), Kluwer Academic Publishers, 2004.

[6] SZYDLO, M.: *Merkle Tree Traversal in Log Space and Time*. In: Advances in Cryptology–EUROCRYPT '04 (Ch. Cachin and J. Camenisch, eds.), Lect. Notes Comput. Sci., Vol. 3027, Springer-Verlag, Berlin, 2004, pp. 541–554.

Received October 27, 2005

Ľubica Staneková
Department of Mathematics
Slovak University of Technology
Radlinského 11
813 68 Bratislava
SLOVAKIA
E-mail: ls@math.sk

Martin Stanek
Department of Computer Science
Comenius University
Mlynská dolina
842 48 Bratislava
SLOVAKIA
E-mail: stanek@dcs.fmph.uniba.sk