The Provable Security of Graph-Based One-Time Signatures and Extensions to Algebraic Signature Schemes

Alejandro Hevia* and Daniele Micciancio**

Dept. of Computer Science & Engineering, University of California, San Diego,
9500 Gilman Drive, La Jolla, California 92093, USA.
{ahevia,daniele}@cs.ucsd.edu
www-cse.ucsd.edu/users/{ahevia,daniele}.

Abstract. Essentially all known one-time signature schemes can be described as special instances of a general scheme suggested by Bleichenbacher and Maurer based on "graphs of one-way functions". Bleichenbacher and Maurer thoroughly analyze graph based signatures from a combinatorial point of view, studying the graphs that result in the most efficient schemes (with respect to various efficiency measures, but focusing mostly on key generation time). However, they do not give a proof of security of their generic construction, and they leave open the problem of determining under what assumption security can be formally proved. In this paper we analyze graph based signatures from a security point of view and give sufficient conditions that allow to prove the security of the signature scheme in the standard complexity model (no random oracles). The techniques used to prove the security of graph based one-time signatures are then applied to the construction of a new class of algebraic signature schemes, i.e., schemes where signatures can be combined with a restricted set of operations.

1 Introduction

One-time signatures [Lam79] are digital signature schemes where the signer is restricted to sign a single document. They are interesting cryptographic primitives because they allow to solve many important cryptographic problems, and at the same time offer substantial efficiency advantages over regular digital signature schemes (cf. [RSA78,Sch90,GMR88,BM92]), especially with respect to signing, verification and key generation time. Applications of one time signatures include the design of regular signature schemes [Mer87,Mer90,BM92,DN94], on-line/off-line signatures [EGM96], digital signatures with forward security properties [BM99,AR00,MMM02], efficient broadcast authentication protocols [Per01] [Roh99], network routing protocols [HPT97], and more. The first one-time signature scheme was proposed by Lamport [Lam79] and (in an interactive setting) by

* Supported in part by NSF grant CCR-0093029 and Mideplan Scholarship.
** Supported in part by NSF Career Award CCR-0093029.

Rabin [Rab78]. The idea of the basic scheme of Lamport is very simple: given a one-way function f, one selects two random strings x_0, x_1 (which constitute the secret key), and publishes $f(x_0), f(x_1)$. Then, a single bit message $b \in \{0,1\}$ can be signed by revealing x_b. Verification is performed in the obvious way. Notice how the signing process is almost instantaneous, while verification only involves a single application of a one-way function. Key generation is almost as efficient, requiring only two applications of the one-way function.

Since Lamport's original proposal, many extensions and improvements have been suggested [MM82,Mer82,Mer87,Vau92,BC93,EGM96,BM94,BM96b,BM96a] [Per01]. The improvements usually involve iterating the application of the one-way function, or revealing multiple values as part of a signature. All these schemes (with the exception of Perrig's) can be described as special instances of a general scheme suggested by Bleichenbacher and Maurer [BM94,BM96b,BM96a], based on the use of "graphs of one-way functions". These are directed acyclic graphs or DAGs (see next section for a formal definition) with values associated to the vertices computed according to one-way functions associated to the edges (see Figure 1). Messages are signed by revealing the values for some of the vertices, and signatures verified using the publicly available one-way functions. As pointed out in [BM94,BM96b,BM96a] DAG-based one-time signatures schemes generalize and have potential advantages over schemes simply based on the iterated application of the one-way function (which correspond to graphs consisting of a collection of disjoint chains). Unfortunately, one-wayness does not seem a sufficiently strong assumption to guarantee the security of the graph based one time signature schemes. In fact, [BM94] and subsequent papers only study the combinatorial properties of the graphs, e.g., trying to maximize the size of the message space that can be signed using graphs with a predetermined number of vertices. The issue of determining sufficient security assumptions on the "one-way function" f, and proving the security of graph based signatures in the standard complexity model is left open in [BM94,BM96b,BM96a].

Our Contributions: In this paper we analyze the security of graph based signatures in order to put them on the firm grounds of the standard computational complexity security model. We show that under standard assumptions the security of graph based signatures can be formally proved. In order to achieve provable security, we adopt an approach in the definition of graph based signatures that is dual to the one used in [BM94]. Namely, instead of associating values to the nodes of a graph and functions to the edges, we propose to associate values to the edges and functions to the nodes (Figure 2 shows an example). Then, we prove that if the functions associated to the nodes are regular collision resistant (or simply universal one-way) hash functions and one-to-one pseudorandom generators, then the resulting one-time digital signature scheme is provably hard to break. These primitives can be built starting from any one-way permutation. The regularity and one-to-one properties can be relaxed assuming that the hash functions and pseudo-random generator only satisfy pseudorandomness and collision resistant properties.

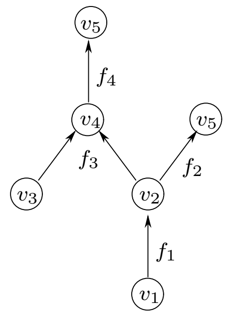

Fig. 1. DAG where values are associated to vertices and functions to edges (e.g. $v_2{=}f_1(v_1)$, $v_6{=}f_2(v_2)$, $v_4{=}f_3(v_3{,}v_2)$, $v_5{=}f_4(v_4)$).

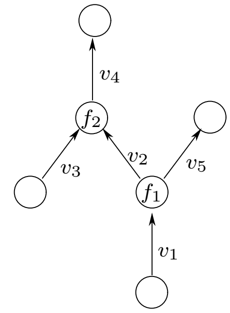

Fig. 2. DAG where values are associated to edges and functions to vertices (e.g. $(v_2{,}v_5){=}f_1(v_1){,}v_4{=}f_2(v_3{,}v_2)$).

An important byproduct of this work is the use of a hybrid argument in a novel way in our proof. Indeed, in order to prove the security of the signature scheme, our analysis involves telling two distributions apart. However, a direct hybrid argument cannot be used because the number of hybrid distributions may be exponential on the security parameter. We show that by carefully setting a total order relation on the hybrids, we can combine them into a small (polynomial) number and the proof goes through. To the best of our knowledge this is a novel use of hybrid argument and may be of independent interest.

Extensions: Graph-based one-time signatures can be extended to instantiate a new type of signature scheme referred as *algebraic signatures*, originally suggested by Rivest [MR02]. An algebraic signature scheme is a signature scheme in which computing signatures of unseen messages is allowed in a restricted way. Associated to each algebraic signature scheme there is a set of functions $\mathcal{O} = \{f_1, \ldots, f_t\}$ (where each function f_i maps messages into messages). The fundamental property of algebraic signature schemes is that given signatures $sig(m_1), \ldots, sig(m_r)$ anyone can compute signature $sig(f_i(m_1, \ldots, m_r))$. Clearly, algebraic signatures require the definition of a new notion of unforgeability. Namely, an algebraic signature scheme is secure if no adversary can efficiently compute signatures of messages that cannot be computed from $m_1, \ldots, m_r$ by applying the functions in $\mathcal{O}$. (See Section 6 for details). Micali and Rivest [MR02], and, recently, Bellare and Neven [BN02], presented constructions of *transitive signatures* which allow to sign edges in an undirected graph in such a way that computing signatures of the transitive closure of the signed edges does not require knowledge of the secret key. Similarly, Johnson et al. [JMSW02] studied several cases where the signing algorithm is homomorphic with respect to a binary operation f_i.

Building on graph-based one-time signature schemes we give explicit constructions for algebraic signatures on sets which support union and subset operations and also union and super-intersection operations[1]. We see graph-based algebraic signatures as an area that deserves further research, since it may lead to efficient and useful constructions.

2 Notation and basic definitions

In this section we review some definitions used throughout the paper. We start by recalling some standard definitions about cryptographic primitives and directed graphs.

2.1 Cryptographic Primitives

We first recall the standard definition of security of signature schemes under chosen-message attacks (cf. [GMR88]) adapted to the case of one-time signature schemes. Then, we recall the (also standard) definitions of security of collision-resistant one-way hash functions (cf. [BR97]) and pseudorandom generators (cf. [BM84,Yao82]).

One-Time Signature Scheme: Formally, a signature scheme consists of three algorithms $\Sigma = (\mathsf{KG}, \mathsf{Sig}, \mathsf{Vf})$. Given a security parameter $k \in \mathbf{N}$, the key generation algorithm $\mathsf{KG}(k)$ outputs a pair of public and private keys (pk, sk); Sig is the signing algorithm taking as input a key sk and a message m, and returning a signature σ; Vf is the verification algorithm taking as input a key pk, a message m and a signature σ, and returning a boolean decision. The signing algorithm may be randomized but the verification algorithm is usually deterministic. It is required that valid signatures are always accepted. A one-time signature scheme is secure against existential forgery in a one-chosen-message attack if no computationally bounded adversary (*forger*), after obtaining the signature of a single message of his choice, can output a (different) message and a corresponding valid signature, except with negligible probability.

Collision-Resistant Hash Functions: Let $\mathcal{H}$ be a family of functions. An individual element in $\mathcal{H}$ is function $H: R^2 \to R$, for some fix set R. The family $\mathcal{H}$ is said to be collision-resistant if, for H randomly chosen in $\mathcal{H}$, any computationally bounded adversary (*collision-finder*) can not find two different messages m and m' that map by H to the same value, except with negligible probability. Furthermore, we say $\mathcal{H}$ is *regular* if it satisfies $\Pr\left[H(X) = y \;:\; X \stackrel{R}{\leftarrow} R^2 \right] = \Pr\left[Y = y \;:\; Y \stackrel{R}{\leftarrow} R \right]$ for all $y \in R$, and all $H \in \mathcal{H}$.

Pseudorandom Generators: Let $G: R \to R^2$ be a deterministic function. G is a pseudorandom generator if it no computationally bounded adversary (*distinguisher*) can tell apart the output of $G(x)$ on a random input x from a truly random value on R^2 with non-negligible probability. Also, a pseudorandom

[1] The super-intersection of sets A and B, denoted $A \odot B$, is the collection of all sets S such that $A \cap B \subseteq S \subseteq A \cup B$.

generator G is *one-to-one* if there is no pair of distinct inputs $x, x' \in R$, that produce the same output on G.

2.2 Graphs

A *directed graph* is a pair (V, E) where V is a finite set of *vertices* and $E \subseteq V \times V$ is the set of *edges*. A *path* of length $\ell \geq 0$ from v_0 to v_ℓ in G is a sequence of vertices $p = (v_0, \ldots, v_\ell)$ such that $(v_{i-1}, v_i) \in E$ for all $i = 1, \ldots, \ell$. If such a path exists, we say that v_0 is a *predecessor* of v_ℓ and v_ℓ is a *successor* of v_0. The sets of predecessors and successors of v are denoted $\mathtt{Pred}(v)$ and $\mathtt{Succ}(v)$, respectively. A set of vertices S is *predecessor closed* if $\mathtt{Pred}(v) \subseteq S$ for all $v \in S$. Similarly, S is *successor closed* if $\mathtt{Succ}(v) \subseteq S$ for all $v \in S$. A *cycle* is a path $(v_0, \ldots, v_\ell)$ of length $\ell \geq 1$ such that $v_0 = v_\ell$. A *directed acyclic graph* (DAG) is a directed graph with no cycles.

The *indegree* of a vertex v is the number of edges $(v', v) \in E$ pointing to v, the *outdegree* is the number of edges $(v, v') \in E$ departing from v, and the *total degree* is the sum of the indegree and the outdegree. Vertices with indegree 0 are called *sources*, and vertices with outdegree 0 are called *sinks*. Vertices that are neither sources nor sinks are called *internal* vertices. For simplicity, in this paper we only considers DAGs with a single source $v_\perp$ with outdegree 1, a single sink $v_\top$ with indegree 1, and $n > 0$ internal nodes with total degree 3. For such graphs, there are only two kind of internal vertices: *expansion vertices* with indegree 1 and outdegree 2, and *compression vertices* with indegree 2 and outdegree 1. So, the sets of vertices of our graphs can be partitioned as $V = V_G \cup V_H \cup \{v_\perp, v_\top\}$, where V_G are the expansion vertices and V_H the compression vertices. We also fix a total order relation $(V_G, \leq)$ that extends the partial order defined over V_G by the predecessor relation.

An example of DAG is depicted in Figure 3. Vertex 0 is the source, vertex 11 the sink, $V_H = \{1, 2, 3, 4, 5\}$ are compression vertices, and $V_G = \{6, 7, 8, 9, 10\}$ are expansion vertices.

A *cut* in a graph (V, E) is a nontrivial partition $C = (S, \bar{S})$ of the vertices such that S is predecessor closed (or, equivalently, $\bar{S}$ is successor closed). The set of cuts in a graph (V, E) is denoted $\mathtt{Cuts}(V, E)$, and it forms a partial order where $(S, \bar{S}) \sqsubseteq (S', \bar{S}')$ if and only if $S \subseteq S'$ (or, equivalently, $\bar{S} \supseteq \bar{S}'$). Notice that since $(S, \bar{S})$ is nontrivial (i.e., both S and $\bar{S}$ are not empty), and $S, \bar{S}$ are predecessor and successor closed, it is always the case that $v_\perp \in S$ and $v_\top \in \bar{S}$. Therefore, a cut can be implicitly represented by a single set of vertices S with the convention that if $v_\perp \in S$ then (S) represents $(S, V \setminus S)$, while if $v_\top \in S$ then (S) represents $(V \setminus S, S)$. For any cut C, the component of C containing $v_\perp$ (resp. $v_\top$) is denoted $S(C)$ (resp. $\bar{S}(C)$).

An edge $e = (u, v)$ *crosses* a cut $C = (S, \bar{S})$ if $u \in S$ and $v \in \bar{S}$. The set of edges crossing C is denoted $\mathtt{Edges}(C) = E \cap (S \times \bar{S})$. We consider graphs where each edge is labeled with an element from some set R. The labels associated to the edges are not totally independent, but must satisfy certain constraints. Let $G\colon R \to R^2$ and $H\colon R^2 \to R$ be two arbitrary functions. (Later on, we will

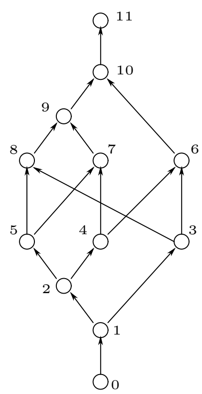

Fig. 3. Example of a DAG $\mathcal{G}$.

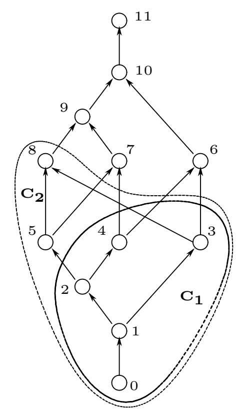

Fig. 4. Two cuts $\mathbf{C_1} \sqsubseteq \mathbf{C_2}$ in $\mathcal{G}$.

instantiate G with a pseudorandom generator and H with a collision resistant hash function.) A labeling is a partial function λ from E to R, i.e., a function $\lambda: T \to R$ where $T \subseteq E$. The domain T of the labeling is denoted $\mathrm{dom}(\lambda)$. We say that λ is consistent (with respect to functions G and H) if values are computed according to functions G and H, i.e.,

– for every expansion vertex with incoming edge $e_0 \in \mathrm{dom}(\lambda)$ and outgoing edges $e_1, e_2 \in \mathrm{dom}(\lambda)$, $G(\lambda(e_0)) = (\lambda(e_1), \lambda(e_2))$.
– for every compression vertex with incoming edges $e_0, e_1 \in \mathrm{dom}(\lambda)$ and outgoing edge $e_2 \in \mathrm{dom}(\lambda)$, $\lambda(e_2) = H(\lambda(e_0), \lambda(e_1))$.

We are interested in labeling functions defined over cuts. A *labeled cut* is a labeling function σ such that $\mathrm{dom}(\sigma)$ is the set of edges of a cut, i.e., $\mathrm{dom}(\sigma) = \mathtt{Edges}(C)$ for some $C \in \mathtt{Cuts}(V, E)$. If σ is a labeling with domain $\mathtt{Edges}(C)$ then we write $\sigma: C$. Similarly, we denote as $\{\sigma: C\}$ the set of all labellings with domain $\mathtt{Edges}(C)$. Notice that any function $\sigma: \mathtt{Edges}(C) \to R$ is consistent, i.e., the edges of a cut can be labeled independently. Any labeled cut $\sigma: C$ can be uniquely extended to a consistent labeling defined over all edges ending in $\bar{S}(C)$.

Proposition 1. *For any directed acyclic graph (V, E), cut $C \in \mathtt{Cuts}(V, E)$ and labeling σ: $\mathtt{Edges}(C) \to R$, there exists a unique labeling, denoted $[\sigma]$, such that*
(1) *$dom([\sigma]) = E \cap (V \times \bar{S}(C))$*
(2) *$[\sigma]$ is consistent, and*
(3) *$[\sigma](v) = \sigma(v)$ for all $v \in \mathtt{Edges}(C)$.*
Moreover, $[\sigma]$ can be efficiently computed from σ.

Notice that for any two cuts $C_1 \sqsubseteq C_2$, the set $\mathtt{Edges}(C_2)$ is contained in $V \times \bar{S}(C_1)$. Therefore, given a labeled cut $\sigma_1: C_1$ and a cut C_2 such that $C_1 \sqsubseteq C_2$, we can define a labeled cut $\sigma_2: C_2$ by restricting the domain of $[\sigma_1]$ to $\mathtt{Edges}(C_2)$.

Definition 1. *For any ordered pair of cuts $C_1 \sqsubseteq C_2$, we define a corresponding projection operation $\Pi_{C_2}^{C_1}$ (or, simply, Π_{C_2} when C_1 is clear from the context)*

Algorithm $\mathsf{KG}(1^k)$ $H \xleftarrow{R} \mathcal{H}$, $\sigma_\perp \xleftarrow{R} \{\sigma\colon\{v_\perp\}\}$ $\sigma_\top \leftarrow \Pi_{\{v_\top\}}(\sigma_\perp)$ $pk \leftarrow (H, \sigma_\top)$, $sk \leftarrow (H, \sigma_\perp)$ **return** (pk, sk)	**Algorithm** $\mathsf{Sig}(sk, m)$ **parse** sk as $(H, \sigma_\perp)$ $\sigma \leftarrow \Pi_{\mu(m)}(\sigma_\perp)$ **return** $\sigma : \mu(m)$	**Algorithm** $\mathsf{Vf}(pk, m, \sigma)$ **parse** pk as $(H, \sigma_\top)$ **if** $\Pi_{\{v_\top\}}(\sigma) = \sigma_\top$ **return** 1 **else return** 0

Fig. 5. Key Generation, Signing and Verification algorithms for GBOTS scheme.

that maps any labeled cut $\sigma_1\colon C_1$ to a corresponding labeled cut $\sigma_2\colon C_2$ obtained by first extending σ_1 to $[\sigma_1]$, and then restricting the domain of $[\sigma_1]$ to the set $\mathtt{Edges}(C_2)$.

Notice that if $C_1 = (S_1, \bar{S}_1)$ and $C_2 = (S_2, \bar{S}_2)$, then $\sigma_2 = \Pi_{C_2}(\sigma_1)$ can be computed from σ_1 with at most $|S_2 \setminus S_1|$ applications of functions G and H.

Example 1. Figure 4 depicts two example cuts $S(C_1) = \{0,1,2,3,4\}$ with $\mathtt{Edges}(C_1) = \{(2,5),(4,7),(4,6),(3,6),(3,8)\}$, and $S(C_2) = \{0,1,2,3,4,5,8\}$ with $\mathtt{Edges}(C_2) = \{(8,9),(5,7),(4,7),(4,6),(3,6)\}$. As a toy example, consider $R = \mathbb{Z}_{10}$, $H(x,y) \stackrel{\text{def}}{=} x+y$, and $G(x) \stackrel{\text{def}}{=} (x,x)$. If we choose $\{((2,5),3), ((4,7),9), ((4,6),5), ((3,6),2), ((3,8),8)\}$ as a labeled cut $\sigma{:}C_1$ in $\mathcal{G}$, then it is easy to check that the labeled cut defined by $\Pi_{C_1}^{C_2}(\sigma)$ (the consistent extension of C_1 onto C_2) is $\{((8,9),1), ((5,7),3), ((4,7),9), ((4,6),5), ((3,6),2)\}$.

3 The GBOTS construction

A graph based one-time signature (GBOTS) scheme is specified by a directed acyclic graph (V,E), a function $\mu : \mathcal{M} \to \mathtt{Cuts}(V,E)$ from a message space $\mathcal{M}$ to the set of cuts of the graph, a length doubling function $G : R \to R^2$ and a family $\mathcal{H}$ of length halving functions $H : R^2 \to R$. Function μ must satisfy the security property that if $m \neq m'$, then the cuts $\mu(m)$ and $\mu(m')$ are incomparable, i.e., neither $\mu(m) \sqsubseteq \mu(m')$ nor $\mu(m') \sqsubseteq \mu(m)$. In particular, function μ is injective. Examples of such functions are presented in [BM96b,BM96a].

The secret key of a GBOTS scheme consists of a labeled cut $\sigma_\perp\colon\{v_\perp\}$ and a hash function $H \in \mathcal{H}$, both chosen uniformly at random. The corresponding public key is given by function H and the labeled cut $\sigma_\top = \Pi_{\{v_\top\}}(\sigma_\perp)$. A signature for a message $m \in \mathcal{M}$ is a labeled cut $\sigma\colon\mu(m)$. Message m is signed using secret key $(H, \sigma_\perp)$ setting $\sigma = \Pi_{\mu(m)}(\sigma_\perp)$. A message signature pair $(m, \sigma\colon\mu(m))$ is verified using public key $(H, \sigma_\top)$ checking that $\Pi_{\{v_\top\}}(\sigma) = \sigma_\top$. A formal specification of the GBOTS scheme is given in Figure 5.

4 The Reduction

In this section we relate the security of GBOTS to the security of the underlying pseudorandom generator G and family of hash functions $\mathcal{H}$. Formally, we show how a forger adversary $\mathcal{F}$ that successfully attacks the one-time signature scheme, can be used to build efficient procedures to successfully attack G and $\mathcal{H}$ as follows: an inverter algorithm $\mathcal{I}_H$ that attempts to invert a randomly chosen function $H \in \mathcal{H}$; an inverter algorithm $\mathcal{I}_G$ that attempts to invert function G; a collision finder algorithm $\mathcal{C}_H$ that on input $H \in \mathcal{H}$ attempts to find a collision to H, and a distinguisher $\mathcal{D}_G$ that attempts to tell random strings and pseudorandom strings apart.

None of the adversaries $\mathcal{I}_G, \mathcal{I}_H, \mathcal{C}_H, \mathcal{D}_G$ is individually guaranteed to work, but we can bound the success probability of the forger $\mathcal{F}$ as a function of the combined success probabilities of $\mathcal{I}_G, \mathcal{I}_H, \mathcal{C}_H, \mathcal{D}_G$. So, if $G, \mathcal{H}$ are cryptographically secure, then the GBOTS scheme is secure. In the rest of this section we show how to build $\mathcal{I}_G, \mathcal{I}_H, \mathcal{C}_H, \mathcal{D}_G$ given black box access to the forger $\mathcal{F}$. The success probabilities of these adversaries are analyzed in the following section.

Adversaries $\mathcal{I}_G, \mathcal{I}_H, \mathcal{C}_H, \mathcal{D}_G$ all use the forger $\mathcal{F}$ in a specific way, common to all four of them. So, we describe this general procedure $\mathcal{A}$ first. This procedure takes as input a hash function H, a node v, and a labeling $\sigma_v\colon \mathtt{Pred}(v)$. The task is, given oracle access to the forger algorithm, compute a labeling $\sigma'_v\colon \mathtt{Succ}(v)$. In other words, $\mathcal{A}$ gets as input a labeling of the smallest cut containing v, and tries to output a labeling for the biggest cut not containing v (where biggest and smallest refer to the $\sqsubseteq$ ordering relation).

Procedure $\mathcal{A}(H, v, \sigma_v)$ operates as follows:

1. Compute $\sigma_\top = \Pi_{\{v_\top\}}(\sigma_v)$.
2. Run $\mathcal{F}$ on input $pk = (H, \sigma_\top)$.
3. Let $m \in \mathcal{M}$ be the message output by $\mathcal{F}$. If $v \notin \mu(m)$, then abort. Otherwise, compute $\sigma_m = \Pi_{\mu(m)}(\sigma_v)$ and continue to the next step.
4. Run $\mathcal{F}$ on input σ_m to get a forgery m', σ'. We assume, without loss of generality, that $\mathcal{F}$ always outputs a valid message-signature pair, i.e., $\Pi_{\{v_\top\}}(\sigma') = \sigma_\top$. If $\mathcal{F}$ cannot forge a signature, then it outputs (m, σ_m)
5. If $v \in \mu(m')$ then abort. Otherwise, compute and output $\sigma'_v = \Pi_{\mathtt{Succ}(v)}(\sigma')$.

A few remarks follow. First, for any vertex v, $\mathtt{Pred}(v) \sqsubseteq \{v_\top\}$, so the projection operation in step 1 can always be performed. This produces a pair $pk = (H, \sigma_\top)$ which is similar, but not necessarily identically distributed, to a public key. In step 3, if $v \in \mu(m)$, then $\mathtt{Pred}(v) \subseteq \mu(m)$ because cut $\mu(m)$ is closed. So, unless execution is aborted, $\mathtt{Pred}(v) \sqsubseteq \mu(m)$ and σ_m can be computed from σ_v. Similarly, in step 5, if execution does not abort, $v \notin \mu(m')$ and $\mu(m') \sqsubseteq \mathtt{Succ}(v)$. So, σ'_v can be computed from σ'. Therefore, $\mathcal{A}$ always either aborts or it succeeds, i.e., it outputs a cut $\sigma'_v\colon \mathtt{Succ}(v)$ such that $\Pi_{\{v_\top\}}(\sigma'_v) = \sigma_\top$. We use $\mathcal{A}$ to define $\mathcal{I}_G$, $\mathcal{I}_H$, $\mathcal{C}_H$ and $\mathcal{D}_G$.

4.1 Inverting *H*

Algorithm $\mathcal{I}_H$ on input a hash function H and target value $y \in R$, chooses one vertex $v \in V_H$ at random, and selects σ_v uniformly at random among all labeled cuts $\sigma\colon \mathtt{Pred}(v)$ such that $\sigma(e) = y$, where e is the only edge departing from v. Then algorithm $\mathcal{I}_H$ calls $\mathcal{A}(H, v, \sigma_v)$. If $\mathcal{A}$ aborts, also $\mathcal{I}_H$ aborts. Otherwise, let

σ'_v: $\mathtt{Succ}(v)$ be the signature output by $\mathcal{A}$. The output of $\mathcal{I}_H$ is $\sigma'_v(e_0); \sigma'_v(e_1)$, where e_0, e_1 are the edges pointing to v.

We remark that $\mathcal{I}_H$ may either abort, terminate successfully with a pre-image of y under H, or fail, i.e., terminate without aborting, but with an output value $x_0; x_1$ such that $H(x_0; x_1) \neq y$. The distinction between aborting execution and failure to invert will be used in the analysis.

4.2 Inverting *G*

The algorithm to invert G is similar to $\mathcal{I}_H$. $\mathcal{I}_G$ on input a target value $(x_1; x_2) \in R^2$, chooses $H \in \mathcal{H}$ uniformly at random, picks one vertex $v \in V_G$, and selects σ_v uniformly among all labeled cuts σ: $\mathtt{Pred}(v)$ such that $\sigma(e_1) = x_1$ and $\sigma(e_2) = x_2$, where e_1, e_2 are the edges departing from v. Then it calls $\mathcal{A}(H, v, \sigma_v)$. If $\mathcal{A}$ aborts, also $\mathcal{I}_G$ aborts. Otherwise, let σ'_v: $\mathtt{Succ}(v)$ be the signature output by $\mathcal{A}$. The output of $\mathcal{I}_G$ is $\sigma'_v(e_0)$ where e_0 is the edge pointing to v. As for $\mathcal{I}_H$, inverter $\mathcal{I}_G$ can either abort, terminate successfully, or fail.

4.3 Finding Collisions

In order to describe the collision finder algorithm we need the following lemma. The proof is simple and can be seen in the full version of this paper [HM02]. The proof uses the assumption that G is one-to-one.

Lemma 1. *For any cut $C \in \mathtt{Cuts}(V, E)$, and labellings $\sigma : C$ and $\sigma' : C$, if $\sigma \neq \sigma'$ and $\Pi_{\{v_\top\}}(\sigma) = \Pi_{\{v_\top\}}(\sigma')$, then there exists a compression node v not in C with incoming edges e_0, e_1 such that $([\sigma](e_0), [\sigma](e_1))$ and $([\sigma](e_0), [\sigma](e_1))$ form a collision, i.e., $H([\sigma](e_0), [\sigma](e_1)) = H([\sigma](e_0), [\sigma](e_1))$ and $[\sigma](e_i) \neq [\sigma'](e_i)$ for some $i \in \{0, 1\}$.*

The collision finder $\mathcal{C}_H$ takes as input a hash function H, and selects a vertex $v \in V_G \cup V_H$ uniformly at random. Notice that $v \in V_G$ and $v \in V_H$ happen with the same probability because V_G and V_H have the same size. The rest of the collision finder algorithm is similar to $\mathcal{I}_G$ or $\mathcal{I}_H$, depending on whether $v \in V_G$ or $v \in V_H$.

If $v \in V_G$, then $\mathcal{C}_H$ chooses $x \in R$ uniformly at random, computes $(y_1; y_2) = G(x)$, and picks σ_v uniformly at random among all labeled cuts σ: $\mathtt{Pred}(v)$ such that $\sigma(e_1) = y_1$ and $\sigma(e_2) = y_2$, where e_1, e_2 are the edges departing from v. Then it calls $\mathcal{A}(H, v, \sigma_v)$. If $\mathcal{A}$ aborts, also $\mathcal{C}_H$ aborts. Otherwise, let σ'_v: $\mathtt{Succ}(v)$ be the signature output by $\mathcal{A}$, and consider the cut $\mathtt{Succ}(v) \setminus \{v\}$. Notice that $\mathtt{Succ}(v) \sqsubseteq \mathtt{Succ}(v) \setminus \{v\}$ and $\mathtt{Pred}(v) \sqsubseteq \mathtt{Succ}(v) \setminus \{v\}$. Therefore, we can compute two labeling $\sigma = \Pi_{\mathtt{Succ}(v) \setminus \{v\}}(\sigma_v)$ and $\sigma' = \Pi_{\mathtt{Succ}(v) \setminus \{v\}}(\sigma'_v)$. If $\sigma \neq \sigma'$, then compute a collision from σ and σ' using Lemma 1.

If $v \in V_H$, then $\mathcal{C}_H$ chooses $x_0, x_1 \in R$ uniformly at random, compute $x_2 = H(x_0, x_1)$, and pick σ_v uniformly at random among all labeled cuts σ: $\mathtt{Pred}(v)$ such that $\sigma(e_2) = y_2$. It then call $\mathcal{A}(H, v, \sigma_v)$. If $\mathcal{A}$ aborts, also $\mathcal{I}_H$ aborts. Otherwise, let σ'_v: $\mathtt{Succ}(v)$ be the signature output by $\mathcal{A}$, and consider the cut $\mathtt{Succ}(v) \setminus$

$\{v\}$. As before, $\mathtt{Succ}(v) \sqsubseteq \mathtt{Succ}(v) \setminus \{v\}$ and $\mathtt{Pred}(v) \sqsubseteq \mathtt{Succ}(v) \setminus \{v\}$. Therefore, we can compute two labeling $\sigma = \Pi_{\mathtt{Succ}(v)\setminus\{v\}}(\sigma_v)$ and $\sigma' = \Pi_{\mathtt{Succ}(v)\setminus\{v\}}(\sigma'_v)$. If $\sigma \neq \sigma'$, then compute a collision from σ and σ' using Lemma 1.

4.4 Distinguishing G

Finally we describe a possible distinguisher for G. On input $x_1, x_2 \in R^2$, $\mathcal{D}_G$ picks a random vertex $v \in V$ and a hash function $H \in \mathcal{H}$. This time vertex v is not selected with uniform probability, but with probability proportional to $|V_G \cap (\mathtt{Pred}(v) \setminus \{v\})|$. Then $\mathcal{D}_G$ chooses a node $u \in V_G \cap (\mathtt{Pred}(v) \setminus \{v\})$ uniformly at random, and computes σ_v as follows. Let $\{\sigma \colon \cup_{u' \leq u} \mathtt{Pred}(u')\}$ denote the set of all labellings defined over the union of cuts $\mathtt{Pred}(u')$ for all expansion vertices $u' \leq u$ in the predecessor set of v but not including v; in other words, it denotes the union of cuts $\mathtt{Pred}(u')$ such that $u' \leq u$, and $u' \in V_G \cap (\mathtt{Pred}(v) \setminus \{v\})$. In this union, each labeling satisfies $\sigma(e_1); \sigma(e_2) = x_1; x_2$, where e_1, e_2 are the edges departing from u. Distinguisher $\mathcal{D}_G$ selects σ_u uniformly at random in $\{\sigma \colon \cup_{u' \leq u} \mathtt{Pred}(u')\}$, and computes $\sigma_v = \Pi_{\mathtt{Pred}(v)}(\sigma_u)$. Notice that for all u' predecessor of v, $\mathtt{Pred}(u') \subset \mathtt{Pred}(v)$, and the labeled cut σ_v can be computed from σ_u.

Procedure $\mathcal{A}$ is run on input H, v, σ_v. If $\mathcal{A}$ aborts then $\mathcal{D}_G$ outputs "random", while if $\mathcal{A}$ does not abort $\mathcal{D}_G$ outputs "pseudorandom".

5 Analysis

In this section we relate the success probability of the forger algorithm $\mathcal{F}$ to the success probability of attacks to G and H. The following result states that if G is a one-to-one pseudorandom generator and $\mathcal{H}$ is a regular collision-resistant hash function family then the GBOTS scheme is existentially secure under one-chosen-message attack.

Theorem 1. *Let (V, E) be a directed acyclic graph, G a one-to-one pseudorandom generator, and $\mathcal{H}$ a regular collision resistant family of hash functions, and consider the corresponding GBOTS scheme. Let $\mathcal{F}$ be a forger that succeeds with probability δ. Then $\delta \leq (\alpha\epsilon_D + \epsilon_C + \epsilon_G + \epsilon_H)n$ where $\alpha \leq n$ is the average number of V_G predecessors of a random vertex in the graph and $\epsilon_G, \epsilon_H, \epsilon_C, \epsilon_D$ are the success probabilities (or advantage) of adversaries $\mathcal{I}_G$, $\mathcal{I}_H$, $\mathcal{C}_H$, $\mathcal{D}_G$ as defined in the previous section.*

In order to prove the result, we first show that the success probability of the adversaries $\mathcal{I}_G$, $\mathcal{I}_H$ and $\mathcal{C}_H$ is tightly related to the aborting probability of procedure $\mathcal{A}$, when called on randomly chosen inputs. We make this statement more precise below. First, we need some notation.

A labeled cut σ is said to be *consistent with* $(v, y) \in V \times (R^2 \cup R)$ if one of two cases hold: **(a)** if $v \in V_G$ and $y = y_1; y_2 \in R^2$ then $\sigma(e_1) = y_1$ and $\sigma(e_2) = y_2$ where e_1 and e_2 are the edges departing from v, or **(b)** if $v \in V_H$ and $y \in R$ then $\sigma(e) = y$ where e is the only edge departing from v. The set of all labeled cuts σ consistent with (v, y) is denoted $\{\sigma : \texttt{Pred}(v)_y\}$. In particular, if either $v \in V_G$ and $y = G(x)$ for $x \in R$ chosen uniformly at random, or $v \in V_H$ and $y = H(x_1; x_2)$ for $x_1; x_2 \in R^2$ chosen uniformly at random, the set $\{\sigma : \texttt{Pred}(v)_y\}$ is denoted $\{\sigma\colon \texttt{Pred}(v)_{H/G(\cdot)}\}$.

Consider the following experiment. First, we choose a vertex $v \in V_H \cup V_G$, a hash function $H \in \mathcal{H}$ and a labeled cut $\sigma_v \in \{\sigma\colon \texttt{Pred}(v)_{H/G(\cdot)}\}$ uniformly at random. Then we call procedure $\mathcal{A}$ on input (H, v, σ_v). (For simplicity's sake, when clear from the context, we use $\mathcal{A}(\cdot)$ to denote $\mathcal{A}(H, v, \sigma_v)$). Let $\mathsf{NoAbort}$ denote the event that $\mathcal{A}$ does not abort in this experiment. The following lemma shows that the combined success probability of adversaries $\mathcal{I}_G$, $\mathcal{I}_H$ and $\mathcal{C}_H$ is equal to the probability of the event $\mathsf{NoAbort}$.

Lemma 2. *Let ϵ_H, ϵ_G and ϵ_C the advantages of adversaries $\mathcal{C}_H$, $\mathcal{I}_G$ and $\mathcal{I}_H$. Let* $\mathsf{NoAbort}$ *be the event as described above. Then* $\epsilon_H + \epsilon_G + \epsilon_C = \Pr[\,\mathsf{NoAbort}\,]$

Proof. We analyze the success probability of adversaries $\mathcal{I}_H, \mathcal{I}_G$ and $\mathcal{C}_H$ in turn. First, the success probability ϵ_H of adversary $\mathcal{I}_H$ is the probability that, for $x_1; x_2 \in R^2$ and $H \in \mathcal{H}$ uniformly chosen at random, $H(\mathcal{I}_H(H, H(x_1; x_2)))) = H(x_1; x_2)$, that is, that $\mathcal{I}_H$ returns a pre-image of $H(x_1; x_2)$ for a random domain point $x_1; x_2$. For $X \in \{H, G\}$, let $\Pr_X[\,E\,]$ denote the probability of event E when $H \in \mathcal{H}$, $v \in V_X$ and $\sigma_v \in \{\sigma\colon \texttt{Pred}(v)_{H/G(\cdot)}\}$ are chosen uniformly at random. Then

$$\begin{aligned}\epsilon_H &= \Pr_H[\,H(x') = H(x), x' \leftarrow \mathcal{A}(H, v, \sigma_v), x' \neq \mathsf{abort}\,]\\ &= (1 - \Pr_H[\,H(x') \neq H(x) \mid \mathcal{A}(\cdot) \neq \mathsf{abort}\,]) \cdot \Pr_H[\,\mathcal{A}(\cdot) \neq \mathsf{abort}\,]\end{aligned}$$

Similarly, for adversary $\mathcal{I}_G$ we have

$$\epsilon_G = (1 - \Pr_G[\,G(x') \neq G(x) \mid \mathcal{A}(\cdot) \neq \mathsf{abort}\,]) \cdot \Pr_G[\,\mathcal{A}(\cdot) \neq \mathsf{abort}\,]$$

Lastly, recall that Adversary $\mathcal{C}_H$ is successful if, after running $\mathcal{A}$ on a randomly chosen $v \in V_H \cup V_G$, either $G(x) \neq G(x')$ if $v \in V_G$ or $H(x) \neq H(x')$ if $v \in V_H$. Thus,

$$\begin{aligned}\epsilon_C = \frac{1}{2} \cdot (&\Pr_H[\,H(x') \neq H(x) \mid x' \leftarrow \mathcal{A}(\cdot), x' \neq \mathsf{abort}\,] \cdot \Pr_H[\,\mathcal{A}(\cdot) \neq \mathsf{abort}\,] +\\ &\Pr_G[\,G(x') \neq G(x) \mid \mathcal{A}(\cdot) \neq \mathsf{abort}\,] \cdot \Pr_G[\,\mathcal{A}(\cdot) \neq \mathsf{abort}\,])\end{aligned}$$

Combining the above results and using that $|V_H| = |V_G|$ the result follows.

As a second step toward proving Theorem 1, next lemma shows that the success probability of the distinguisher $\mathcal{D}_G$ is related to the difference between forger's success probability and the probability that procedure $\mathcal{A}$ does not abort (in the experiment described in the previous lemma).

Lemma 3. *Let ϵ_D and δ denote the advantage of distinguisher $\mathcal{D}_G$ and forger $\mathcal{F}$ respectively, and let α and* $\mathsf{NoAbort}$ *defined as before. Then*

$$\delta \leq n \cdot (\alpha \epsilon_D + \Pr[\,\mathsf{NoAbort}\,])\ .$$

The following notation will be useful in the proof. For any $v \in V$, let $W(v) = V_G \cap \mathtt{Pred}(v) \setminus \{v\}$ denote the set of all expansion vertices which are predecessors of v. Also, given a vertex $v \in V$ and a vertex $u \in W(v)$, let $\mathtt{Pred}_v(\leq u) = \cup_{u' \leq u,\, u' \in W(v)} \mathtt{Pred}(u')$ the cut formed by the union over $u' \leq u$ of all sets $\mathtt{Pred}(u') \subset \mathtt{Pred}(v)$. (Recall that $\leq$ is a total order relation over V_G) Also, let $\{\sigma\colon \mathtt{Pred}_u(\leq v)\}$ denote the set of all labeled cuts on $\mathtt{Pred}_v(\leq u)$; as before, for $y_1; y_2 \in R^2$, let $\sigma\colon \mathtt{Pred}_u(\leq v)_{y_1;y_2}$ denote the set of all labeled cuts compatible with $(u, y_1; y_2)$. (We stress that the compatibility is with respect to vertex u, that is, $\sigma(e_1); \sigma(e_2) = y_1; y_2$). As before, if $x \in R$ is uniformly distributed, the set $\{\sigma\colon \mathtt{Pred}_v(\leq u)_{G(x)}\}$ is denoted by $\{\sigma\colon \mathtt{Pred}_v(\leq u)_{G(\cdot)}\}$. Notice that in this extended definition, $\mathtt{Pred}_v(\leq u) \subset \mathtt{Pred}(v)$ and therefore a labeled cut for $\mathtt{Pred}(v)$ can be computed from any labeled cut in $\{\sigma\colon \mathtt{Pred}_u(\leq v)\}$.

Proof (Lemma 3). By definition, $\epsilon_D = p_1 - p_0$, where p_1 and p_0 denote the probability that $\mathcal{D}_G(y_1; y_2) = 1$ when $x \xleftarrow{R} R, y_1; y_2 \leftarrow G(x)$ and the probability that $\mathcal{D}_G(y_1; y_2) = 1$ when $y_1; y_2 \xleftarrow{R} R^2$, respectively. Consider the following two experiments, which we denote $\mathbf{Exp}_1$ and $\mathbf{Exp}_0$. In the first one, we choose $H \in \mathcal{H}$ uniformly at random, $v \in V_H \cup V_G$ with probability proportional to $|W(v)|$, $u \in W(v)$ and $\sigma_u \in \{\sigma\colon \mathtt{Pred}_v(\leq u)_{G(\cdot)}\}$ uniformly at random; we then compute σ_v as an extension of σ_u by $\sigma_v = \Pi_{\mathtt{Pred}(v)}(\sigma_u)$ and finally call $\mathcal{A}$ on input (H, v, σ_v). The second experiment, $\mathbf{Exp}_0$, is similar to the previous one, with the exception that σ_u is drawn at random from $\{\mathtt{Pred}_v(\leq u)_{y_1;y_2}\}$ for $y_1; y_2 \xleftarrow{R} R^2$. Let $q_1(v', u')$ and $q_0(v', u')$ denote the probability procedure $\mathcal{A}$ does not abort in $\mathbf{Exp}_1$ and $\mathbf{Exp}_0$ respectively, conditioned on the event that $v = v'$ and $u = u'$ are chosen in each experiment.

Let $\alpha = \frac{1}{n} \sum_{v \in V_H \cup V_G} |W(v)|$ be the average number of expansion vertices of a random vertex in the graph. We claim that,

$$p_1 = \frac{1}{n\alpha} \cdot \sum_{v \in V_H \cup V_G} \sum_{u \in W(v)} q_1(v, u) \;\; \text{and} \;\; p_0 = \frac{1}{n\alpha} \cdot \sum_{v \in V_H \cup V_G} \sum_{u \in W(v)} q_0(v, u) \quad (1)$$

and that for all $v \in V_H \cup V_G$, $u \in W(v) \cup \{v\}$

$$q_0(v, u^*) = q_1(v, u) \quad (2)$$

$$\sum_{v \in V_H \cup V_G} q_1(v, \overline{v}) \geq \delta \quad (3)$$

$$\sum_{v \in V_H \cup V_G} q_0(v, v^*) = n \cdot \Pr[\,\mathsf{NoAbort}\,] \quad (4)$$

where $w^* = \max_{w' < w}(w')$, denotes the biggest vertex in V_G smaller than $w \in V_G$ and $\overline{v} = \min_{v \in V_G}(v)$ is the "smallest" expansion vertex in V_G (where "biggest" and "smallest" refer to the $\leq$ ordering relation).

Before proving these claims, we use them to finish the proof of the lemma. Using equations (1-4), we have

$$\epsilon_D = \frac{1}{n\alpha} \sum_{v \in V_H \cup V_G} \{q_1(v, \overline{v}) - q_0(v, v^*)\} \geq \frac{1}{n\alpha} \cdot (\delta - n \cdot \Pr[\,\mathsf{NoAbort}\,])$$

which gives the desired result.

We now justify the claimed equations (1-4) by analyzing each them in turn. To justify the first part of (1), notice that by definition of p_1 and standard conditioning we have

$$p_1 = \Pr\Big[\mathcal{A}(H, v, \sigma_v) \neq \mathsf{abort} \;:\; v \xleftarrow{W} V_G \cup V_H, u \xleftarrow{R} W(v), H \xleftarrow{R} \mathcal{H}, x \xleftarrow{R} R,$$
$$\sigma_u \xleftarrow{R} \mathtt{Pred}_v(\leq u)_{G(x)}\Big] \;=\; \sum_{v \in V_H \cup V_G} \sum_{u \in W(v)} q_1(v, u) \cdot \Pr[\, u \mid v\,] \cdot \Pr[\, v\,]$$
$$= \sum_{v \in V_H \cup V_G} \frac{\Pr[\, v\,]}{|W(v)|} \sum_{u \in W(v)} q_1(v, u)$$

where $v \xleftarrow{W} V_G \cup V_H$ means vertex v is drawn from set $V_G \cup V_H$ with probability proportional to $|W(v)|$. Since, for all $v \in V$, $\Pr[\, v\,] = |W(v)|/(n\alpha)$ Equation (1) holds. The second part of (1) follows from a similar argument.

We justify Equation (2) as follows. Fix $v \in V_H \cup V_G$ and $u \in W(v) \cup \{v\}$. Consider experiment $\mathbf{Exp}_0$, and assume v and $u^* \in W(v)$ are the vertices chosen. First of all, notice that $\mathtt{Pred}_v(\leq u^*) \subset \mathtt{Pred}_v(\leq u)$ because $u^* \leq u$, and thus, σ_u can be computed from σ_{u^*}. Second, assume $v \in V_G$. Since the labeled cut $\sigma_{u^*} \in \sigma\colon \mathtt{Pred}_v(\leq u^*)$ is chosen uniformly at random, there is no other expansion node in any path from $u' \leq u^*$ and u, and H is regular, the induced labeled cut $\sigma_u = \Pi_{\mathtt{Pred}(u)}(\sigma_{u^*}) \in \sigma\colon \mathtt{Pred}_v(\leq u)$ is such that $\sigma_u(e_1); \sigma_u(e_2) = G(x)$ for some $x \in R$ uniformly distributed (e_1 and e_2 are edges leaving vertex u). The same argument when $v \in V_H$ boils down to $\sigma_u(e) = H(x_1; x_2)$ for uniformly distributed $x_1; x_2 \in R^2$ and e the only leaving edge of u. Thus, $\sigma_u \in \sigma\colon \mathtt{Pred}_v(\leq u)_R$, and $q_0(v, u^*) = q_1(v, u)$.

To justify Equation (3) we notice that when distinguisher $\mathcal{D}_G$ chooses $u = \overline{v}$, the distribution of the public key and signature so computed by $\mathcal{A}$ from σ_u follows the same distribution than the forger expects in the one-chosen-message attack and, thus, the output of the forger is independent of the choice of v.

$$\sum_{v \in V_H \cup V_G} q_1(v, \overline{v}) =$$
$$\sum_{v \in V_H \cup V_G} \Pr[\, \mathcal{F}(m, \sigma_m) = (m', \sigma'), m \neq m', v \in \mu(m), v \notin \mu(m')\,] \;\geq\; \delta$$

where the last inequality follows from that, for any $m, m' \in \mathcal{M}$, if $m \neq m'$ there always exists $v \in V_H \cup V_G$ such that $v \in \mu(m)$ but $v \notin \mu(m')$, otherwise m and m' would be comparable.

It remains to prove Equation (4). This follows from $q_0(v, v^*) = q_1(v, v) = \Pr[\, \mathsf{NoAbort} \mid v\,]$ and from vertex $v \in V_H \cup V_G$ being chosen uniformly at random in the experiment that defines the event $\mathsf{NoAbort}$. This concludes the proof of the lemma.

Proof (Theorem 1). Immediate from Lemma 2 and Lemma 3.

6 Extensions

In this section we consider extensions of the basic security results presented in the previous sections. The first one concerns relaxing the security assumptions about the underlying primitives G, H. The second applies the ideas in our proof of security to build provably secure signature schemes with special algebraic properties.

Universal one-way hash functions: The collision-resistance requirement on the hash function family $\mathcal{H}$ can be relaxed to *universal one-wayness* as defined by Naor and Yung [NY89]. Recall that universal one-way hash function (UOWHF) families are such that it is hard to find a colliding pair $x \neq x'$ such that $H(x) = H(x')$ but the adversary must select x before H is given to it. We modify our GBOTS construction, so that for each compression vertex v a different randomly chosen function $H_v \in \mathcal{H}$ is used. The security argument in this case is modified as follows. In order to compute $\sigma_\top = \Pi_{\{v_\top\}}(\sigma_v)$, algorithm $\mathcal{A}(H, v, \sigma)$ picks a hash function $H_v \in \mathcal{H}$ uniformly at random anew to compute the label of each edge leaving a compression vertex with the exception of the edge corresponding to v, for which H is used. Thus, adversary $\mathcal{I}_H$ needs only to pick ahead a random value $x \in R^2$ and, once given a target hash function H, to use procedure $\mathcal{A}$ to invert $H(x)$. Similarly, for $\mathcal{C}_H$ it suffices to guess the compression vertex where the collision given by Lemma 1 will be found, and use the target hash function H there. Adversaries $\mathcal{I}_H$ and $\mathcal{C}_H$ remain the same. The remaining security argument does not differ substantially from the one presented in Section 5. We point out that regular universal one-way hash functions and one-to-one pseudorandom generators can be constructed from any one-way permutation [NY89,CMR98].

Mapping messages to edges (or vertices): In this paper, we associate values to edges in the graph and functions to vertices. This approach can be seen as dual to the one used in [BM94], which associates values to vertices and function to edges. Both approaches are essentially equivalent from a syntax viewpoint and in terms of the class of schemes they yield. From a foundational viewpoint, we believe that the approach presented here is conceptually simpler.

Graph based algebraic signature schemes: Algebraic signature schemes are signature schemes in which signatures for (certain) new messages can be produced by combining signatures with a restricted set of operations. Since these operations do not require knowledge of the secret key, algebraic signatures are not signature schemes in the standard interpretation of the term, but they are a new cryptographic primitive. They are useful in contexts where possession of signatures of certain messages automatically entitles possession of signatures of new messages, such as in credential systems. Credentials may be implemented as signed documents which specify capabilities (or attributions) to be granted to the credential holder. Thus, if implemented with the appropriate algebraic signature, the possession of one or more credentials (signatures) will automatically enable the computation of the entitled credentials without the involvement of the original signer. Algebraic signatures were originally suggested by Rivest [MR02].

Informally, an algebraic signature scheme consists of three algorithms $\mathcal{AS} = (\mathsf{KG}, \mathsf{Sig}, \mathsf{Vf})$ and a two set of operations $\mathcal{O} = \{f_1, f_2, \ldots, f_q\}$ and $\mathcal{S} = \{g_1, g_2, \ldots, g_s\}$, where each f_i (resp. g_i) is a function that takes one or more messages (resp. signatures) as inputs and produces one message (resp. signature) as output. KG, Sig, and Vf are as in any digital signature scheme (see Section 2.1). We require that if $\delta_1, \ldots, \delta_t$ are valid signatures for $m_1, \ldots, m_t$ then $g_i(\delta_1, \ldots, \delta_t)$ is a valid signature for $f_i(m_1, \ldots, m_t)$ for all appropriate f_i, g_i. Notice that signatures so generated are subject to existential forgery under chosen message attacks, so a new definition of security is required. Let $\mathsf{span}(\mathcal{O}, \{m_1, \ldots, m_t\})$ be the set of all messages computable from $\{m_1, \ldots, m_t\}$ by applying functions in $\mathcal{O}$ on them. The security of algebraic signatures is defined in terms of unforgeability against chosen-message attacks, where by convention, the forger is deemed successful only if it outputs a signature of a message m not in the set $\mathsf{span}(\mathcal{O}, \{m_1, \ldots, m_t\})$.

Graph-based one-time signatures can be used to build very efficient algebraic signatures. Indeed, for practical functions f_i, it is possible to build graphs such that f_i is embedded in the order relation $\sqsubseteq$. That is, if $f_i(m_1, m_2) = m_3$, then there exists a labeling $\sigma\colon \mu(m_3)$ which can be computed from labellings $\sigma_1\colon \mu(m_1)$ and $\sigma_2\colon \mu(m_2)$ and it is consistent with them.

Notice that the proof of security of Section 4 and Section 5 can be easily modified to prove that our (graph-based) algebraic signature scheme $\mathcal{AS}$ is secure. Indeed, the only technical difference is that the forger $\mathcal{F}$ can request multiple signatures $\sigma_m\colon \mu(m)$. This can be easily factored in by modifying Procedure $\mathcal{A}$ so each signature σ_m is computed from σ_v (or $\mathcal{A}$ aborts, if not possible). Since the forger $\mathcal{F}$ must output (m', σ') for m' not in $\mathsf{span}(\mathcal{O}, \cup_m \mu(m))$, there must exist $v \in \cup_m \mu(m)$ so $v \notin \mu(m')$ and the argument goes through. The rest of the proof is identical and, in particular, adversaries $\mathcal{I}_G$, $\mathcal{I}_H$, $\mathcal{C}_H$, $\mathcal{D}_G$ remain the same, given black-box access to $\mathcal{A}$.

Concrete constructions of Algebraic Signature Schemes: In this section we sketch concrete graph constructions that yield algebraic signature schemes with respect to (a) union and subset operations, and (b) union and super-intersection operations. (Recall that the super-intersection of sets A and B, denoted $A \odot B$, is the collection of all sets S such that $A \cap B \subseteq S \subseteq A \cup B$.)

Let $\mathcal{M}$ be the set of all subset of n elements, where we denote such elements as $t_0, \ldots, t_{n-1}$. Consider the graph shown in Figure 6. (Although the figure shows vertices v_i having indegree and outdegree 1, and the vertices $v'_\perp$ and $v'_\top$ having outdegree and indegree n, respectively, it is easy to cast this graph as one with the properties considered in this work. Indeed, it suffices to replace each vertex v_i with a small subgraph of 2 compression and 2 expansion vertices, and to connect each v_i to both $v'_\perp$ and $v'_\top$ by simple tree construction).

We map every set S into the set of vertices $\mu(S) = C$ defined as follows: vertices $v_\perp$ and $v'_\perp$ are in C, and vertices v_i are in C if and only if $t_i \notin S$. Notice C is a valid cut for any set S. Given a labeled cut $\sigma\colon \mu(S)$ the labeling for any $C' = \mu(S')$ such that $S' \subseteq S$ can be computed by projecting $\sigma\colon \mu(S)$ on C'. The union operation is defined similarly, since given labeled cuts $\sigma_1\colon \mu(S_1)$ and $\sigma_2\colon \mu(S_2)$ a consistent labeled cut for $\mu(S_1 \cup S_2)$ can be computed.

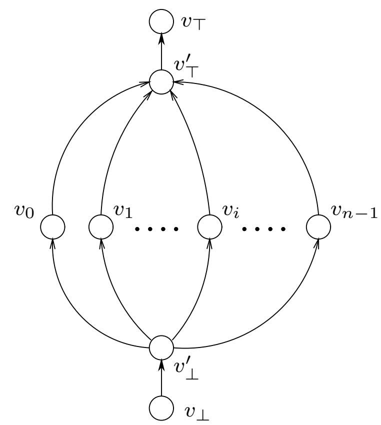

Fig. 6. DAG for algebraic scheme with operations $\{\cup, \text{subset}\}$.

A algebraic signature scheme for the $\{\cup, \odot\}$ operations can be build by using two graphs $\mathcal{G}_1$ and $\mathcal{G}_2$ each one like the one described above. In this case, given a set S, we define the cut on the first graph by using the above shown rule, while for the second case we "invert" the condition, and we include the corresponding vertices only if $t_i \in S$. It is an easy exercise to verify that such mapping allows the computation of labeled cuts corresponding to the union and super-intersection of two sets S_1 and S_2, given labeled cuts $\sigma_1 \colon \mu(S_1)$ and $\sigma_2 \colon \mu(S_2)$.

7 Conclusions

In this paper, we analyze graph based signatures from a security viewpoint and give sufficient conditions, namely the existence of one-way permutations, under which the signature scheme is secure in the standard complexity model (no random oracles). Additionally, we present a security proof which uses a new hybrid argument where the number of hybrid distributions may be exponential. We believe this technique is of independent interest. We also propose a new paradigm for the construction of algebraic signature schemes, which are new useful primitives for applications where controlled "forgeability" of signatures is needed, as in credential systems.

7.1 Acknowledgments

We want to thank Bogdan Warinschi, and Yee Hea Ann for helpful comments and discussions.

References

[AR00] M. Abdalla and L. Reyzin. A new forward-secure digital signature scheme. In ASIACRYPT'2000, LNCS 1976, pages 116–129. Springer-Verlag, 2000.

[BC93] J. N. E. Bos and D. Chaum. Provable unforgeable signatures. In CRYPTO'92, LNCS 740, pages 1–14. Springer-Verlag, 1993.

[BKR94] M. Bellare, J. Kilian, and P. Rogaway. The security of the cipher block chaining message authentication code. In CRYPTO'94, LNCS 839. Springer-Verlag, 1994.

[BM84] M. Blum and S. Micali. How to generate cryptographically strong sequences of pseudorandom bits. In *Siam Journal of Computing*, 13(4):850–864, 1984.

[BM92] M. Bellare and S. Micali. How to sign given any trapdoor function. In *Journal of Cryptology*, 39(1):214–233, 1992.

[BM94] D. Bleichenbacher and U. M. Maurer. Directed acyclic graphs, one-way functions and digital signatures. In CRYPTO'94, LNCS 839, pages 75–82. Springer-Verlag, 1994.

[BM96a] D. Bleichenbacher and U. M. Maurer. On the efficiency of one-time digital signatures. In ASIACRYPT'96, LNCS 1163, pages 145–158. Springer-Verlag, 1996.

[BM96b] D. Bleichenbacher and U. M. Maurer. Optimal tree-based one-time digital signature schemes. In STACS'96, LNCS 1046, pages 363–374. Springer-Verlag, 1996.

[BM99] M. Bellare and S. Miner. A forward-secure digital signature scheme. In CRYPTO'99, LNCS 1666, pages 431–448. Springer-Verlag, 1999.

[BN02] M. Bellare and G. Neven. Transitive Signatures based on Factoring and RSA In ASIA-CRYPT'02, (these proceedings).

[BR97] M. Bellare and P. Rogaway. Collision-resistant hashing: Towards making UOWHFs practical. In CRYPTO'97, LNCS 1294, pages 470–484. Springer-Verlag, 1997.

[CLR92] T. H. Cormen, C. E. Leiserson, and R. L. Rivest. In *Introduction to algorithms.* MIT Press and McGraw-Hill Book Company, 6th ed., 1992.

[CMR98] R. Canetti, D. Micciancio, and O. Reingold. Perfectly one-way probabilistic hash functions. STOC'98, pages 131–140. ACM, 1998.

[DN94] C. Dwork and M. Naor. An efficient existentially unforgeable signature scheme and its applications. In CRYPTO'94, LNCS 839, pages 234–246. Springer-Verlag, 1994.

[EGM96] S. Even, O. Goldreich, and S. Micali. On-line/off-line digital signatures. In *Journal of Cryptology*, 9(1):35–67, 1996.

[GMR88] S. Goldwasser, S. Micali, and R. Rivest. A digital signature scheme secure against adaptive chosen-message attacks. In *Siam Journal of Computing*, 17(2):281–308, 1988.

[HPT97] R. Hauser, A. Przygienda, and G. Tsudik. Reducing the cost of security in link state routing. In *Symposium on Network and Distributed Systems Security (NDSS '97)*, pages 93–99, Internet Society, 1997.

[HM02] A. Hevia and D. Micciancio. The provable security of Graph-Based One-Time Signatures and extensions to algebraic signature schemes. Full version of this paper, available via `http://www-cse.ucsd.edu/users/ahevia`.

[JMSW02] R. Johnson, D. Molnar, D. Song, and D. Wagner. Homomorphic signature schemes. In CT-RSA '2002, LNCS 2271, pages 244–262. Springer-Verlag, 2002.

[Lam79] L. Lamport. Constructing digital signatures from a one way function. Technical Report CSL-98, SRI International, 1979.

[Mer82] R. C. Merkle. In *Secrecy, Authentication, and Public Key Systems*, vol. 18 of *Computer science. Systems programming.* UMI Research Press, 1982.

[Mer87] R. C. Merkle. A digital signature based on a conventional encryption function. In CRYPTO'87, LNCS 293, pages 369–378. Springer-Verlag, 1987.

[Mer90] R. C. Merkle. A digital signature based on a conventional encryption function. In CRYPTO'89, LNCS 435, pages 428–446. Springer-Verlag, 1990.

[MM82] C. H. Meyer and S. M. Matyas. In *Cryptography: A New Dimension in Computer Data Security*. John Wiley and Sons, New York, 1982.

[MMM02] T. Malkin, D. Micciancio, and S. Miner. Efficient generic forward-secure signatures with an unbounded number of time periods. In EUROCRYPT'2002, LNCS 2332, pages 400–417. Springer-Verlag, 2002.

[MR02] S. Micali and R. L. Rivest. Transitive signature schemes. In CT-RSA '2002, LNCS 2271, pages 236–243. Springer-Verlag, 2002.

[NY89] M. Naor and M. Yung. Universal one-way hash functions and their cryptographic applications. STOC'89, pages 33–43. ACM, 1989

[Per01] A. Perrig. The BiBa one-time signature scheme and broadcast authentication protocol. In *Proceedings of the 8th ACM conference on Computer and Communications Security*, pages 28–37. ACM, 2001.

[Rab78] M. O. Rabin. Digitalized signatures. In R. A. DeMillo, D. P. Dobkin, A. K. Jones, and R. J. Lipton, editors, *Foundations of Secure Computation*, pages 155–168. Academic Press, 1978.

[Roh99] P. Rohatgi. A compact and fast hybrid signature scheme for multicast packet authentication. In *Proceedings of the 6th ACM conference on Computer and communications security*, pages 93–100, ACM, 1999.

[RSA78] R. L. Rivest, A. Shamir, and L. Adleman. A method for obtaining digital signature and public-key cryptosystems. In *Communications of the ACM*, 21(2):120–126, 1978.

[Sch90] C. Schnorr. Efficient identification and signatures for smartcards. In CRYPTO'89, LNCS 435, pages 239–252. Springer-Verlag, 1990.

[Vau92] S. Vaudenay. One-time identification with low memory. In *Eurocode 92*, CISM Courses and Lectures, no. 339, pages 217–228, Springer-Verlag, 1992.

[Yao82] A. Yao. Theory and applications of trapdoor functions. FOCS'82, pages 80–91. IEEE, 1982.