How to Authenticate Graphs Without Leaking

Ashish Kundu, Elisa Bertino
Department of Computer Science and CERIAS, Purdue University.
West Lafayette, Indiana, USA
{ashishk,bertino}@cs.purdue.edu

ABSTRACT

Secure data sharing in multi-party environments requires that both authenticity and confidentiality of the data be assured. Digital signature schemes are commonly employed for authentication of data. However, no such technique exists for directed graphs, even though such graphs are one of the most widely used data organization structures. Existing schemes for DAGs are authenticity-preserving but *not* confidentiality-preserving, and lead to leakage of sensitive information during authentication.

In this paper, we propose two schemes on how to *authenticate* DAGs and directed cyclic graphs *without leaking*, which are the first such schemes in the literature. It is based on the structure of the graph as defined by depth-first graph traversals and aggregate signatures. Graphs are structurally different from trees in that they have four types of edges: tree, forward, cross, and back-edges in a depth-first traversal. The fact that an edge is a forward, cross or a back-edge conveys information that is sensitive in several contexts. Moreover, back-edges pose a more difficult problem than the one posed by forward, and cross-edges primarily because back-edges add bidirectional properties to graphs. We prove that the proposed technique is *both* authenticity-preserving and non-leaking. While providing such strong security properties, our scheme is also efficient, as supported by the performance results.

1. INTRODUCTION

Authentication of data is an important problem, and has been widely investigated [9, 24], especially in the context of databases [22, 27, 28, 29, 30]. Authentication is a stronger security requirement than integrity assurance [19]. An authentication scheme is used to verify (1) the integrity of data, and (2) the fact that whether the received data object O is indeed sent by the claimed sender from the claimed data source. For example, for a message M, a signed cryptographic hash of the message M is generally used to verify the authenticity of the source as well as the integrity of the message. Specific authentication requirements and techniques depend on the structure according to which data is organized [1, 21, 26]. Since one of the most widely used data organization structures is the graph structure, the development of such techniques specific to graph structures is crucial.

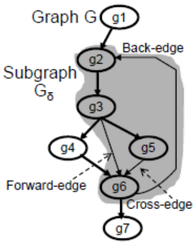
Graph-structured data widely occurs in various practical fields such as GIS (Geographical Information Systems), ontology, healthcare, biology, molecular science and military. Graph data structures are either explicitly stored as part of the data such as in biological databases and XML graphs, or extracted through techniques such as data mining [18, 32] (and link mining [11]). In what follows, "graph" refers to a directed acyclic graph (DAG) or a directed graph with cycles.

When addressing the problem of authentication of graph structures, it is important to note that a graph consists of nodes and edges; each node contains some contents, and the edge between two nodes represents some relationship between the contents of these nodes. Integrity of such relationships is referred to as *structural authenticity*, while the authenticity of the contents is referred to as *content authenticity*. An authenticity scheme for graph structures must thus preserve both structural and content authenticity. In many cases, such as healthcare, biological and military scenarios, an additional requirement is to maintain the confidentiality of the content and structural information [1, 21, 33]. By confidentiality we mean that: (i) a prover, the entity that verifies data authenticity (also referred to as a *receiver* or *user*)[1], receives only the nodes and structural information that it is allowed to access according to the stated access control policies; (ii) a prover should not receive nor should be able to infer any information about the content and presence of nodes and structural relationships that it is not authorized to access. We refer to such information as *extraneous information*. In case of sensitive information being shared, the process of authentication should not lead to information leakage. Information leakage leads to confidentiality breaches and violation of privacy policies. Because of the increasing number of applications requiring the ability to be able to store, manage and share (query/search/publish) graph-structured data, authentication schemes with confidentiality guarantees must not only be devised but should also be efficient in terms of implementation at each party involved.

Digital signature schemes are commonly used for authentication of data. However no such technique exists for graphs in the literature, even though there are authentication schemes for directed acyclic graphs (DAGs) and trees, which do not contain cycles and are a restricted form of graphs. The Merkle hash technique [26] is the most widely adopted technique for authentication of trees, which is however authenticity-preserving (binding) but not confidentiality-preserving (hiding) [7]. Martel et al. [25] proposed a technique for DAGs; however since it uses the Merkle hash technique, it is binding but not hiding. In our earlier work [21], we proposed a leakage-free integrity assurance technique for trees. In this paper, we de-

EDBT 2010, March 22–26, 2010, Lausanne, Switzerland.
Copyright 2010 ACM 978-1-60558-945-9/10/0003 ...$10.00

[1] A prover is typically a software running on behalf of the user to verify data authenticity.



Node	Post-order	Pre-order
g1	7	1
g2	6	2
g3	5	3
g4	3	4
g5	4	7
g6	2	5
g7	1	6

Figure 1: A graph with depth-first tree in bold.

Table 1: Information leakages via edge-types.

Type of $e(x, y)$	*Associated information leakages*
Forward-edge	(i) in-degree of $y \geq 2$. (ii) at least one more edge e' incident on y. (iii) e' is a tree-edge. (iv) at least one more node w, such that $x \ldots w \ldots y$ is a simple path. (v) source graph is larger than received graph.
Cross-edge	(i), (ii), (v).
Back-edge	(a) at least one simple path from y to x. (b) at least one cycle in the graph, (c) one cycle is between x and y; (d) (v).

velop a leakage-free authentication scheme for DAGs and graphs with cycles. However, since graphs are more complex structures than trees and graphs with cycles cannot be topologically sorted, the problem addressed in this paper is harder to solve than the one addressed in [21].

Graphs are structurally different from trees. Edges in a graph can be organized into four different types based on a specific depth first traversal of the graph: tree-edges, forward-edges, cross-edges and back-edges (related to cycles), while trees have only one type of edges: tree-edges [20]. Depth-first traversal numbers are used to determine the type of an edge for that specific traversal Such numbers are assigned to a node in the order in which they are visited in a specific traversal; for example, post-order and pre-order numbers are assigned to the nodes in post-order and pre-order traversals, respectively. (For more details, the reader is referred to [20]). The various types of edges in a graph are defined below using the notion of traversal numbers.

DEFINITION 1.1. *Let τ be the depth-first tree (DFT) of a directed graph $\mathcal{G}(V, E)$. Let x, $y \in V$, and $e(x, y) \in E$. Let o_x and q_x refer to post-order number and pre-order number of node x, respectively. With respect to the DFT τ, $e(x, y)$ is a (1) tree-edge, iff $o_x > o_y$, and $q_x < q_y$; (2) forward-edge, iff there is a path from x to y consisting of more than one tree-edges, $o_x > o_y$, and $q_x < q_y$; (3) cross-edge, iff $o_x > o_y$, and $q_x > q_y$; (4) back-edge, iff $o_x < o_y$, and $q_x > q_y$.*

An example of depth-first tree, types of edges are given in Figure 1 with the post- and pre-order numbers for each node being given in the table in the figure. An authenticity-preserving and confidentiality-preserving scheme for graph data must not convey the knowledge of whether a given edge is a forward-edge (edge $e(g_3, g_6)$), a cross-edge (edge $e(g_5, g_6)$) or a back-edge (edge $e(g_6, g_2)$), unless the user is authorized to access a corresponding tree-edge(s) or the associated cycle, respectively. The information leakages due to the knowledge about the type of the edge are listed in Table 1. Such leakages are described in detail in the context of health information in Section 2. Whether an edge is a back or a cross-edge can be determined using both post-order and pre-order numbers [20]. Therefore, the structural signature scheme for trees [21] cannot solve the problem for graphs.

The problem that the paper addresses is as follows: The trusted owner Alice of a data item organized as a connected directed graph $\mathcal{G}$ wants to digitally sign $\mathcal{G}$ *once* so that it can be queried or accessed many times. A user Bob should be able to verify the authenticity of the content and structure of a connected subgraph G_δ (of $\mathcal{G}$) that Bob is authorized to access. Any extraneous information, *i.e.* information about a node or a structural relation (edge) which is in $\mathcal{G}$ but not in G_δ should *not* be revealed to Bob.

In this paper, we develop two such provably leakage-free authentication schemes, one for DAGs and another for directed graphs with cycles. In each of these schemes, the signature of a graph (refers to a DAG or a directed graph with cycles), called "structural signature of a graph", is based on the structure of the graph as defined by graph traversals and aggregate signatures based on bilinear maps [6]. For DAGs that contain tree-, forward- and cross-edges, we define the notion of forward-structural signature (α-structural signature) or just forward signature, which is provably binding and hiding. For directed graphs with cycles, in order to handle back-edges, we introduce the notion of back-structural signature (β-structural signature), which is defined for all the nodes that are origins of back-edges and/or reachable from back-edges. Intuitively, such a technique splits a graph into a forward-DAG containing only tree-edges and cross-edges, and a number of backward-DAGs containing the back-edges, the nodes that are reachable from back-edges over simple paths[2] and the edges in such simple paths. The β-structural signatures and the α-structural signatures are used together in our scheme for directed graphs with cycles.

In addition to formally defining the techniques, we prove that they protect against violations of content and structural authenticity, and information leakages. While providing such strong security guarantees, our schemes are also efficient. For DAGs, it incurs linear cost ($O(n)$, n is the number of nodes in the DAG) for computation, storage and distribution of verification objects for arbitrary DAGs. For graphs with cycles, the cost is $O(n * d)$, where d is the maximum number of back-edges that are incident on any node in the graph. Our scheme is efficient as supported by performance results for DAGs as large as two million nodes. Moreover, our performance analyis shows that the time to sign not only DAGs but also graphs with cycles is linear with respect to the number of nodes, even for very large DAGs and graphs.

Outline of the paper. Section 2 introduces a running example showing how and what sensitive information can be leaked during authentication of graphs. The security model is presented in Section 3. Section 4 gives a brief overview of aggregate signatures, and summarizes the "structural signatures for trees". The signatures for DAGs and graphs with cycles are presented in Sections 5 and 6, respectively. Security-related lemmas for the proposed schemes are stated in the subsequent section, following which, in Section 8, the performance results are discussed. Related work is discussed in Section 9. Section 10 concludes the paper.

2. RUNNING EXAMPLE

Our example is in the area of XML data management. XML is widely used to represent data both in tree and graph forms; the "IDREF" attribute or a user-defined attribute is used to refer to another element (node) in the XML document, which leads to a non-

[2] A simple path among two nodes is one in which each node in the path appears only once [8].

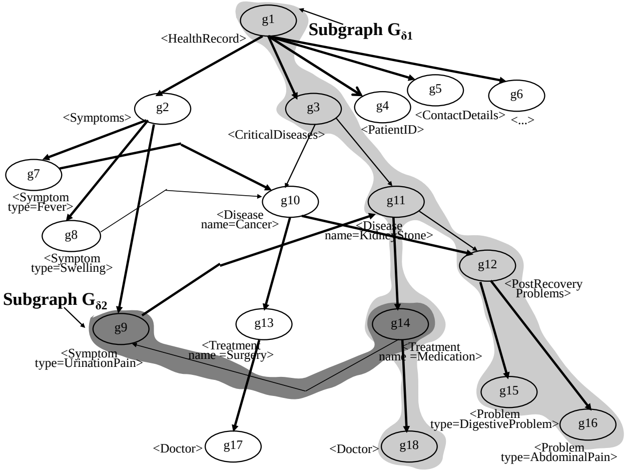


Figure 2: A health-record graph; tree-edges in bold.

tree edge (either a back-edge, cross-edge, or a forward-edge). The data object in Figure 2 is a graph-based representation of XML-based health-care record of a patient, which typically contains sensitive information. The hospital database, which can be accessed remotely, stores all such patient health records.

Consider the following scenario. An insurance manager handling the insurance claim specifically towards the treatment of the disease *KidneyStone* has access to subgraph $G_{\delta 1}$. As a result of querying the database, the manager receives $G_{\delta 1}$ and a set of authenticity verification items. This portion includes the nodes g_1, g_3, g_{11}, g_{12}, g_{14}, g_{15}, g_{16} and g_{18}. One of the cross-edges received is $(e(g_{11}, g_{12}))$. By knowing that $e(g_{11}, g_{12})$ is a cross-edge, the manager also learns that there is a tree-edge (to which the manager has no access) to the node representing *PostRecoveryProblems*, which implies that the patient has also suffered from another disease.

By knowledge of the schema, the insurance manager determines that an edge to such a node must be from a Disease node, which is a child of the *CriticalDiseases* node. Thus the manager infers that the patient is definitely suffering from some other critical disease. If the hospital specializes in some specific critical disease(s), which is mostly the case in reality, the manager can further infer which (possible) disease the patient is suffering from. Moreover, the manager can definitely infer that such a disease was cured (and thus diagnosed) before the *KidneyStone* disease; this is because the node of *PostRecoveryProblems* already exists. Each of these inferences leads to disclosure of information that is sensitive for the patient. Leakages inferred from the knowledge that $e(g_{11}, g_{12})$ is a cross-edge are listed in Table 3.

We now consider another scenario, which also leads to leakage of some sensitive information by the knowledge that an edge is a back-edge. A pharmacist has access to $G_{\delta 2}$ from the record in Figure 2. She has access to $G_{\delta 2}$, because she provides this medication (specified by node g_{14}) to the patient. The pharmacist receives $G_{\delta 2}$ and the corresponding verification items from the remote database to verify authenticity. If she somehow learns that the $e(g_{14}, g_9)$ is a back-edge from the received information, she would learn that there is a path from g_9 to g_{14} in the health-record. The schema of *HealthRecord* specifies that a symptom node is related to a critical disease from which a patient suffers from, that is there is an edge from the symptom node to a disease node. The disease node further has an edge to the *Medication* node. The only path that can exist from g_9 to g_{14} is through a disease node. It is thus apparent that the patient suffers from a critical disease related to the symptom *UrinationPain*. The symptom name further leads to the possible name of the disease. If the hospital specializes in a limited number of critical diseases, the pharmacist can determine the name of the disease. The treatment type may easily lead to determine the seriousness of the illness. Table 4.

Table 2: Health-record: XML Elements

Element	*Semantics*
HealthRecord	Root of the XML document
Symptoms	Lists the symptoms of a patient
CriticalDiseases	Critical diseases the patient suffers/suffered from.
Symptom	Specifies a symptom. Attributes. *type* refers to the type a symptom. *idref* refers to the related disease.
Disease	Information about a specific critical disease; Attributes: *idref* refers to PostRecoveryProblems.
Treatment	Type of treatment administered on the patient. Attributes: *name* specifies the treatment. *idref* refers to the symptom that a medication may affect.
Doctor	Name of the doctor who administered the treatment.
PostRecovery-Problems	Post-recovery problems that a patient goes through. The node is created only once for the first recovery. For any later disease and recoveries from it, this node is referred to from that Disease node.
Problem	A post-recovery problem. Attribute: *type* specifies the type of the problem.
PatientID	Identifier of the patient. Attributes: *id* and *name*
ContactDetails	Contact details of the patient.

3. SECURITY MODEL

Data Model. A weakly[3] connected directed graph $\mathcal{G}(V, E)$ is a data object, where V and E are the sets of vertices and edges in $\mathcal{G}$. A node x represents an atomic unit of the data referred to as C_x. The fact that x precedes one of its siblings y in a DAG is denoted by $x \prec y$. A subgraph $G'(V', E')$ that is shared with a user as a result of a query on the graph $\mathcal{G}(V, E)$ is a weakly connected graph; $G'(V', E') \subseteq \mathcal{G}(V, E)$.

Distribution Model. We assume an untrusted third-party distribution model. The *trusted owner* Alice ($\mathcal{A}$) of a data object organized as a graph $\mathcal{G}$ signs it. After signing $\mathcal{G}$, $\mathcal{A}$ may delegate the job of publishing $\mathcal{G}$ or processing queries over $\mathcal{G}$ to *third-party distributors* ($\mathcal{D}$), each of whom does not have signature authority and maybe untrusted. A set of verification items or authentication units (IVs) for a subgraph G' denoted by $\mathcal{VO}(G')$ is sent to the user alongwith G'.

Threat: Data tampering attack. An attacker tampers the content, the structural order and/or the type of structural relation (edge) between two or more nodes of a graph.

Threat: Inference attack. A user, who has access to the subgraph

[3] A directed graph that is not disconnected is a weakly connected directed graph [8].

Table 3: Leakage via cross-edge $e(g_{11}, g_{12})$, subgraph: $G_{\delta 1}$.

Distinct leakages	*Related sensitive information leaked*
In-degree of $g_{12} \geq 2$	The patient has suffered from at least one more disease.
There is at least one more path from the node *CriticalDiseases* to g_{12}	(1) This disease is a critical disease. (2) The patient has already recovered from this disease.
The graph is larger than $G_{\delta 1}$	(1) The patient has been treated by another doctor. (2) She has been associated with this hospital earlier. (3) HealthRecord contains more disease-related data.

Table 4: Leakage via back-edge $e(g_{14}, g_9)$, subgraph: $G_{\delta 2}$.

Distinct leakages	*Related sensitive information leaked*
(a) There is at least one cycle in the graph; (b) so there is at least one path from g_9 to g_{14}	(1) Symptom *UrinationPain* has led to a disease. This disease is being treated with medication. (2) Since the treatment is medication, with high probability, this disease is not in a serious stage.
The graph is larger than $G_{\delta 2}$	HealthRecord contains more disease-related data.

G' of a graph $\mathcal{G}$, attempts to infer sensitive information from the signature and IVs sent to it with G'.

Security Definitions. We assume a probabilistic polynomial adversary [19] throughout the paper. A user $\mathcal{B}$ can authenticate the subgraph $G'(V', E')$ that it receives. Authenticity of a graph is defined in Definition 3.1. Leakage-free requirements entail that the user cannot infer any information that is extraneous (Definition 3.2) to G'. Definition 3.3 gives the formal definition of leakage-free authentication.

DEFINITION 3.1 (AUTHENTICATION OF GRAPHS). *A graph $G'(V', E')$ is authenticated as a subgraph of $\mathcal{G}(V, E)$ if and only if (1) no node or edge has been dropped or added to $G'(V', E')$ in an unauthorized manner, and (2) none of the following entities in the graph has been modified: (i) the content C_x of any node $x \in V'$, (ii) any edge $e(x, y) \in E'$, and (iii) (For DAGs) any structural order $x \prec y$, $x, y \in V'$.*

DEFINITION 3.2 (EXTRANEOUS INFORMATION). *Extraneous information in a graph $\mathcal{G}(V, E)$ with respect to its subgraph $G'(V', E')$ comprises of the nodes and edges that are in $\mathcal{G}$ but not in G'.*

DEFINITION 3.3 (LEAKAGE-FREE SIGNING OF GRAPHS). *Let $G'(V', E')$ be a subgraph of graph $\mathcal{G}(V, E)$. The content of each node x $C_x \in \{0, 1\}^*$. A leakage-free signature scheme (S) for graphs is defined as follows:*

1. *A key generation algorithm* Gen *takes as input a security parameter 1^k and outputs a pair of keys* ($\underline{\texttt{pk}}$, $\underline{\texttt{sk}}$)*, where* $\underline{\texttt{pk}}$ *and* $\underline{\texttt{sk}}$ *are called as public keys and private keys respectively. We assume for convention that each of these keys have length k, and k can be determined from* $\underline{\texttt{pk}}$ *and* $\underline{\texttt{sk}}$.
2. *The signing algorithm* Sign *takes as input a private key* $\underline{\texttt{sk}}$ *and a graph $\mathcal{G}(V, E)$, where the content C_x of each node $x, \in V$ is such that $C_x \in \{0, 1\}^*$. It outputs a* set of signature items *(or integrity verifiers) $\mathcal{VO}(\mathcal{G}(V, E))$, computed as $\mathcal{VO}(\mathcal{G}(V, E)) \leftarrow \texttt{Sign}_{\underline{\texttt{sk}}}(\mathcal{G}(V, E))$.*
3. *The deterministic verification algorithm* Vrfy *takes as input a public key* $\underline{\texttt{pk}}$*, a graph $G'(V', E')$, and a set of signature items $\mathcal{VO}(G'(V', E'))$. It outputs a bit b, with $b = 1$ meaning* valid *and $b = 0$ meaning* invalid*. We write this as $b \leftarrow \texttt{Vrfy}_{\underline{\texttt{pk}}}(\mathcal{VO}(G'(V', E')), G'(V', E'))$. It is required that for every k, every* ($\underline{\texttt{pk}}$, $\underline{\texttt{sk}}$) *output by* $\texttt{Gen}(1^k)$*, every graph $\mathcal{G}(V, E)$, and every subgraph $G'(V', E') \subseteq \mathcal{G}(V, E)$, it holds that $\texttt{Vrfy}_{\underline{\texttt{pk}}}(\mathcal{VO}(G'(V', E')), G'(V', E')) = 1$.*
4. *It is required that for a probabilistic polynomial distinguisher $\mathcal{B}$ who receives $G'(V', E')$ and the associated $\mathcal{VO}(G'(V', E'))$, the success probability of inferring that there exists at least a node $y \in (V - V')$, or there exists at least an edge $e \in (E - E')$, is negligible*[4].

4. BACKGROUND

In this section, we give an overview of the structural signatures for trees as a background. Table 4 summarizes the common acronyms and notations used in the paper.

4.1 Summary of Structural Signatures

The structural signature scheme proposed by Kundu and Bertino [21] is used for the integrity assurance (a weaker security notion than authentication) of trees without leakage. The scheme is based on the observation that - "post-order and pre-order sequences of the vertices of a tree uniquely represent the tree". Since traversal numbers are not secure, a cryptographically secure notion of traversal numbers is used for structural positions of nodes. Such secure counterparts of traversal numbers are called as *randomized traversal numbers*. RPON (RRON) is randomized post-order number (resp., randomized pre-order number) and PON (RON) is post-order number (resp., pre-order number).

The properties of traversal numbers with respect to tree-edges, forward-edges, cross-edges, and back-edges is stated by the Definition 1.1. It shows that it is not possible to use the structural signature scheme for trees for authentication of graphs, as such scheme would leak information about the type of edges.

4.2 Aggregate Signatures and Bilinear Maps

Let $\mathbb{G}_1 = \langle \texttt{P} \rangle$ be an additively-written group of prime order p, and let $\mathbb{G}_2$ be a multiplicatively written group of the same prime order p. A mapping $e : \mathbb{G}_1 \times \mathbb{G}_1 \rightarrow \mathbb{G}_2$ is a bilinear map if (i) $e(aX, bY) = e(X, Y)^{ab}$ for all $X, Y \in \mathbb{G}_1$ and $a, b \in \mathbb{Z}_p^*$; and (ii) $\mathbb{G}_2 = \langle e(\texttt{P}, \texttt{P}) \rangle$. The mapping e is efficiently computable, but given only P, aP, and X (but not a) it is computationally infeasible to compute aX (i.e., the Computational Diffie-Hellman problem is difficult in $\mathbb{G}_1$). This difficulty is what enables the signature and aggregate signature schemes based on bilinear pairings.

In this paper, we use the aggregate signature scheme by Boneh at al. [6]. In such scheme, the signer's secret key is $\underline{\texttt{sk}} \in Z_p^*$, $\texttt{Q} = \underline{\texttt{sk}}\texttt{P} \in \mathbb{G}_1$ is public, and the signature for a message m is $\underline{\texttt{sk}}M$ with $M = \mathcal{H}(\texttt{Q}, \texttt{m}) \in \mathbb{G}_1$, where $\mathcal{H}$ is a cryptographic one-way hash function; for convenience, we henceforth omit mention of

[4] A function $\epsilon(k)$ is negligible in cryptography if for every polynomial $p(.)$, there exists an N such that for all integers $k > N$ it holds that $\epsilon(k) < \frac{1}{p(k)}$ ([19]:Definition 3.4).

Table 5: Acronyms and Notations.

Notation/Acro.	*Meaning*
PON, o_x	Post-Order Number, PON of x.
RPON, p_x	Randomized PON, RPON of x.
RON, q_x	Pre-Order Number, RON of x.
RRON, r_x	Randomized RON, RRON of x.
$\mathcal{G}(V, E)$	A weakly connected directed graph with set of vertices V and set of edges E.
$G_\delta(V_\delta, E_\delta)$	A weakly connected subgraph with set of vertices V_δ and set of edges E_δ.
$e(x, y)$	A directed edge from node x to y.
C_x	Content of x.
$\mathcal{H}$	Cryptographically secure hash function. $\mathcal{H}$:{0, 1}* → {0, 1}^{k}
α	Related to *forward*, the direction of forward-, cross- and tree-edges.
β	Related to *backward*, the direction of back-edges. Used in the context of β-nodes and β-*reachable* nodes.
f, χ, τ	Related to forward- cross- and tree-edges.
q_x^χ, r_x^χ	χ-RON of x, χ-RRON of x.
$o_x^{\beta:u\rightarrow v}$, $p_x^{\beta:u\rightarrow v}$, $q_x^{\beta:u\rightarrow v}$, $r_x^{\beta:u\rightarrow v}$	β-PON, β-RPON, β-RON, β-RRON, resp. of x with respect to a *not-β-covered* back-edge $e(u, v)$.
θ_x^τ, θ_x^χ	Structural position of τ-node or χ-node x.
$\theta_x^{\beta:u\rightarrow v}$,$\Psi_x^{\beta:u\rightarrow v}$	Structural position/signature resp., of a β-node or β-*reachable* node x, with respect to a *not-β-covered* back-edge $e(u, v)$.
Ψ_x^α	Forward structural signature of x. (same as α-signature of x).
$\Psi_\mathcal{G}^\alpha$	Structural signature of DAG $\mathcal{G}$. Also referred to as α-signature of $\mathcal{G}$.
$\Psi_\mathcal{G}$	Structural signature of graph $\mathcal{G}$.
$\overset{\cup}{\leftarrow}$	Union of the sets referred to by the left and right operands followed by the assignment of the result to the left op.

the $M = \mathcal{H}(\mathtt{Q}, \mathtt{m})$ and simply say "message M". In the aggregate signatures, given the public P and Q, and given k message-signature pairs $M_i, \Psi_i = \underline{\mathtt{sk}}M_i$, $1 \leq i \leq k$, the signature is verified by checking that the following equality holds: $e(\mathtt{Q}, \sum_{\mathtt{i=1}}^{\mathtt{k}} \mathtt{M_i}) = \mathtt{e}(\mathtt{P}, \sum_{\mathtt{i=1}}^{\mathtt{k}} \Psi_\mathtt{i})$.

5. DAGS

In this section, we develop structural signatures for DAGs that makes use of bilinear maps. We chose bilinear maps primarily because to the best of our knowledge, there is no other mechanism that is as secure as the aggregate signatures based on bilinear maps. One possible option is the scheme for batch verification of RSA signatures [15, 16, 2], which is however not known to be secure (Section 9).

Consider a user, who is authorized to access one or more cross-edges incident upon a node x, but not the associated tree-edge(s). For example with reference to Figure 2, the user has access only to $e(g_{11}, g_{12})$, which is a cross-edge. One way to share a forward or cross-edge with the user, without leaking it to the user the type of the edge (such as the fact that "$e(g_{11}, g_{12})$ is a cross-edge") is to convey to the user that it is a tree-edge. By concealing the original type of an edge (such as forward or cross-edge) and conveying to the user that any edge it receives is of type tree-edge, unless the user also receives the associated tree-edge(s), we can prevent leakages associated with cross-edges. Note that the structural signature makes use of the notion of traversal numbers, and that post-order and pre-order numbers allows one to detect whether an edge is a cross-edge or not (Definition 1.1). Moreover, such numbers cannot be used to differentiate a forward-edge from a tree-edge.

We thus need to define a different notion of traversal numbers so that a cross-edge would be verified by Bob as a tree-edge. In that context, we refer to the end-point of cross-edge(s) as a χ-node. By Definition 1.1, it is the pre-order number (and not the post-order number) of a χ-node that violates the behavior of pre-order numbers that is exhibited in case of tree-edges. We first define a variant of the pre-order numbers denoted by χ-pre-order numbers (in short, χ-RONs) (Definition 5.2) specifically for χ-nodes. Using a randomized notion of such variants of pre-order numbers, denoted by χ-randomized pre-order numbers (in short, χ-RRONs), we define a structural position of a χ-node that satisfies the constraints of a tree-edge in terms of traversal numbers (Definition 1.1). Such a notion of structural position is then used to compute the aggregate signature for the DAG. Since the definitions of χ-RONs and χ-RRONs are specific to cross-edges and χ-nodes only, tree-edges are always conveyed as they are.

Given that the tree-edges, forward-edges, and cross-edges have the same direction, we refer to the signatures for DAGs as *forward-signatures* (denoted by α-signatures). A node that is not a χ-node (in a DAG or in a graph) such as g_3 in Figure 2, is referred to as a tree-node (in short, τ-node).

DEFINITION 5.1 (χ-NODE). *A node x in a connected directed acyclic graph $\mathcal{G}(V, E)$ is a χ-node, iff there exists an edge $e(w, x)$ in $\mathcal{G}$ such that $e(w, x)$ is a cross-edge.*

DEFINITION 5.2 (χ-RONS). *Let x be a χ-node in a connected directed acyclic graph $\mathcal{G}(V, E)$. Let $e(w, x)$ and $e(x, y)$ be two edges in $\mathcal{G}$. The χ-pre-order numbers of x and w denoted by q_x^χ and q_w^χ are defined such that they satisfy the following conditions:(1) $q_x^\chi > q_x^\tau$; (2) if w is a χ-node, $q_x^\chi > q_w^\chi$, else $q_x^\chi > q_w^\tau$; (3) $q_x^\chi < q_y^\tau$; (4) $q_u^\chi < q_x^\chi < q_v^\chi$, where u, x, and v are siblings and $u \prec x \prec v$.*

Since a DAG can be topologically-ordered, the properties of a χ-pre-order number can be satisfied. For a χ-node x, its χ-randomized pre-order number (χ-RRON) is the randomized version of q_x^χ and is denoted by r_x^χ. It is defined in the same way RPON is defined for PON and RRON for RON. Randomized traversal numbers can be easily computed by either of the three techniques mentioned in [21].

In our running example (Figure 2), g_{10} and g_{12} are χ-nodes; $r_{g_{10}}^\chi$ and $r_{g_{12}}^\chi$ are the χ-RRONs of g_{10} and g_{12}. The notions of τ-structural and χ-structural position (τ-position and χ-position in short, respectively) of a τ-node and a χ-node, respectively are defined by the following definitions.

DEFINITION 5.3 (τ-STRUCTURAL POSITION). *Let x be a node in a connected DAG $\mathcal{G}(V, E)$. Its τ-structural position, denoted by θ_x^τ, is defined as the pair of its RPON p_x^τ and RRON r_x^τ, that is, θ_x^τ = (p_x^τ, r_x^τ).*

DEFINITION 5.4 (χ-STRUCTURAL POSITION). *Let x be a χ-node in a connected DAG $\mathcal{G}(V, E)$. Its χ-structural position, denoted by θ_x^χ, is defined as the pair of its RPON p_x^τ and χ-RRON r_x^χ, that is, θ_x^χ = (p_x^τ, r_x^χ).*

In our running example (Figure 2), the τ-structural position of node g_3 is θ_x^τ = ($p_{g_3}^\tau$, $r_{g_3}^\tau$); for χ-node g_{10}, its χ-structural position is θ_x^χ = ($p_{g_{10}}^\tau$, $r_{g_{10}}^\chi$).

```
1. Execute a depth-first traversal of $\mathcal{G}(V,E)$.
2. For each node $x \in V$, compute its post-order number $o^{\tau}_{x}$ and pre-order number $q^{\tau}_{x}$.
3. If $e(w,x) \in E$ is such that ($o^{\tau}_{w} > o^{\tau}_{x}$) and ($q^{\tau}_{w} > q^{\tau}_{x}$), then mark $e(w,x)$ as a $\chi$-edge and $x$ as a $\chi$-node.
4. For each $\chi$-node, compute its $\chi$-RON $q^{\chi}_{\cdot}$
5. For all nodes in $V$, transform the traversal numbers into randomized traversal numbers, that preserve the order.
6. For a $\chi$-node $x$, $\theta^{\chi}_{x} \leftarrow (p^{\tau}_{x}, r^{\chi}_{x})$.
7. For each $x$ that is not a $\chi$-node, assign ($p^{\tau}_{x}$, $r^{\tau}_{x}$) to $x$ as its structural position $\theta^{\tau}_{x}$.
8. For each node $x$ in $V$, compute its authentication unit $\xi_x$:
    (a) If $x$ is a cross-node, $\theta_x \leftarrow \theta^{\chi}_{x}$; Else $\theta_x \leftarrow \theta^{\tau}_{x}$.
    (b) $\xi_x \leftarrow \mathcal{H}(\theta_x \| C_x)$.
9. Choose a secure random $\omega$; Let $\omega_{\mathcal{G}} \leftarrow \mathcal{H}(\omega)$.
10. Compute the signature of the DAG as in equation 1.
```

Figure 3: Algorithm to sign a DAG.

5.1 Signing a DAG

The signature of a DAG is the salted aggregate signature of all the nodes in the DAG. The salt used is a random ω, which can be treated to be associated with a dummy node. The algorithm to sign a DAG is described in Figure 5.1.

DEFINITION 5.5 (AUTHENTICATION UNIT OF A NODE). *Let x be a node in a connected DAG $\mathcal{G}(V,E)$. Let $\mathcal{H}$ denote a one-way cryptographic hash function. The authentication unit ξ_x of x is defined as follows: $\xi_x = \mathcal{H}(\theta_x \| C_x)$, where: if x is a χ-node, θ_x is equal to θ^{χ}_{x}, else it is equal to θ^{τ}_{x}.*

DEFINITION 5.6 (SIGNATURE OF A DAG). *Let ω be a cryptographically secure random. Let $\mathcal{H}$ denote a one-way cryptographic hash function. Let $\underline{\mathtt{sk}}$ be the private key and $\mathtt{P}$ be defined as in an aggregate signature framework (Section 4.2). Let $\omega_{\mathcal{G}}$ be defined as $\mathcal{H}(\omega)$. The structural signature of a graph $\mathcal{G}(V,E)$, denoted by $\Psi_{\mathcal{G}(V,E)}$, is defined as follows:*

$$\Psi_{\mathcal{G}(V,E)} \leftarrow e(\mathtt{P}, \underline{\mathtt{sk}}(\omega_{\mathcal{G}} + \sum_{\mathtt{x} \in \mathtt{V}} \xi_{\mathtt{x}})). \qquad (1)$$

5.2 Distribution

Let $G'(V',E')$ be an arbitrary weakly connected subgraph of the weakly connected DAG $\mathcal{G}(V,E)$. $\mathcal{D}$ sends the following $O(1)$ number of authentication items to $\mathcal{B}$: (1) $G'(V',E')$, and (2) the set of verification units $\mathcal{VO}(G'(V',E'))$ consisting of the signature of the DAG $\Psi_{\mathcal{G}}$, and authentication unit of the subgraph $\xi_{G'}$, which is computed as follows: $\xi_{G'} \leftarrow \omega_{\mathcal{G}} + \sum_{y \in (V-V')} \xi_y$.

5.3 Authentication

A user $\mathcal{B}$ receives (1) a subgraph $G''(V'',E'')$, (2) the signature of the DAG $\Psi_{\mathcal{G}}$, and (3) the authentication unit ξ of the subgraph from the distributor $\mathcal{D}$. The verification process goes as follows.

Verification of Contents. $\mathcal{B}$ evaluates Eq. 2 and then Eq. 3.

$$\Omega \leftarrow \sum_{y \in V''} \mathcal{H}(\theta_y \parallel C_y). \qquad (2)$$

$$e(\mathtt{Q}, \Omega + \xi) \stackrel{?}{=} \Psi_{\mathcal{G}}. \qquad (3)$$

Verification of Structural Relations

1. Execute a depth-first traversal of G'. Let the structural position of each node x be $\theta_x = (p_x, r_x)$.
2. Let x be an immediate ancestor of z; if ($p_x \leq p_z$) or ($r_x \geq r_z$), then this relationship between x and z is incorrect.
3. For ordered-DAGs, let y be the right sibling of z; if ($p_z \geq p_y$) or ($r_z \geq r_y$), then the left-right order among the siblings y and z is incorrect.

Example: Suppose that the back-edge $e(g_{14}, g_9)$ did not exist in our running example (Figure 2), thus turning this graph into a DAG. In such a DAG, the cross-edges are $e(g_{11}, g_{12})$, $e(g_8, g_{10})$, and $e(g_3, g_{10})$. Consider the cross-edge $e(g_{11}, g_{12})$ and the cross-node g_{12}. The α-structural signatures for this DAG are computed such that r^{χ}_{g12} is larger than r^{χ}_{g11} and p^{τ}_{g12} is smaller than p^{τ}_{g11}, which conveys that $e(g_{11}, g_{12})$ is a tree-edge and conceals the fact that it is a cross-edge.

6. GRAPHS WITH CYCLES

In this section, we build on the solution for DAGs and present the general solution for graphs with cycles that handles all the four types of edges.

A back-edge (such as $e(g_{14}, g_9)$ in the graph in Figure 2) should be presented to the user as a tree-edge unless the user has access to an cycle associated with the back-edge also. Following Definition 1.1, both the post-order and pre-order numbers of a node (e.g., g_{14}), which is the origin of a back-edge (denoted by β-nodes), violate the behavior of the respective numbers that is exhibited in case of tree-edges. In order to handle back-edges, we define a notion of β-post-order number (β-PON) and β-pre-order number (β-RON) for each node that either is a β-node or is reachable from a β-node over a simple path. Nodes that are reachable from a β-node x over a simple path over a β-edge $e(x,w)$, also need to be considered for the following reason. The authentication unit of such nodes must be such that when they are presented to an authenticity prover alongwith the related back-edge $e(x,w)$, they should not leak the fact that $e(x,w)$ is a back-edge; rather they should be consistent with the information presented to the prover that $e(x,w)$ is a tree-edge. Examples of β-*reachable* nodes from the β-node g_{14} in Figure 2 are g_9, g_{11}, g_{12}, g_{15} and g_{16}.

We define the notion of β-nodes like the notion of χ-nodes. β-nodes are nodes that are origins of back-edges (Definition 6.1). We then define the notion of β-*reachable* nodes (Definition 6.2). In order to define such signatures, we define a variant of PON and RON for such nodes (Definition 6.4). Such variants satisfy the property of tree-edges in terms of traversal numbers.

DEFINITION 6.1 (β-NODE). *A node x in a graph $\mathcal{G}(V,E)$ is a β-node, iff there exists an edge $e(x,w)$ in $\mathcal{G}$ such that $e(x,w)$ is a back-edge.*

DEFINITION 6.2 (β-REACHABLE). *Let node x be a β-node in a graph $\mathcal{G}(V,E)$ and let $e(x,w)$ be a back-edge. A node y is said to be β-*reachable *from x (over $e(x,w)$) in $\mathcal{G}$ iff either y is w or there exists a simple path $sp(x,y)$ from x to y in $\mathcal{G}$ such that $sp(x,y) = x \rightarrow w \ldots \rightarrow y$.*

Unlike the case of cross-edges (Section 5), in the case of back-edges, a β-node or a β-*reachable* node has the following position(s) and authentication unit(s): (1) β-structural; and (2) χ-structural, if it is a χ-node, τ-structural, otherwise. A β-node, or a node that is β-*reachable*, may have multiple β-structural positions (β-positions

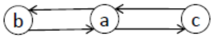


Figure 4: Illustration of not-β-covered edges.

in short) and authentication units. A given node may be reachable from multiple back-edges over a simple-path. We only need to consider a minimal set of such edges, which are not covered by other back-edges in a simple-path. Such back-edges are called *not-β-covered* (Definition 6.3). The number of such positions and authentication units for a node is the same as the number of *not-β-covered* back-edges (Definition 6.3) from which it is reachable over a simple path. However, a user receives *one* position per node. For example, if a user is not permitted to access a cycle, but the related back-edge, β-structural position and signature are sent to the user. However, if the user has access to a cycle, conveying the information that one of the edges in the cycle is a back-edge does not leak any information. In what follows, $x \rightarrow y$ denotes the edge $e(x, y)$ and $y \ldots \rightarrow z$ denotes the simple path $sp(y, z)$ from y to z.

In a graph, a node maybe *β-reachable* from many back-edges. The question is do we need to consider only one of them or "some" of them in order to minimize the number of β-positions. Consider the graph in Figure 4 with a as the root of the depth-first tree; edges $e(a, b)$ and $e(a, c)$ are the tree-edges. Edges $e(b, a)$ and $e(c, a)$ are back-edges. Both b and c are β-nodes. a is *β-reachable* from both b and c. The question is would there be one or two β-positions for a. If a user has access to all nodes and only the back-edges, then how would both of them be proven as tree-edges. To this end, let us first define β-covered and not-β-covered edges.

DEFINITION 6.3 (β-COVERED, NOT-β-COVERED). *A back-edge $e(u, v)$ is said to be "β-covered" by another back-edge $e(x, y)$ in graph $\mathcal{G}(V, E)$ iff there is a simple path $sp(x, v)$ in $\mathcal{G}$ such that $sp(x, v) = x \rightarrow y \ldots \rightarrow u \rightarrow v$. A back-edge $e(u, v)$ is said to be "not-β-covered" iff there exists no such back-edge $e(x, y)$ in $\mathcal{G}$.*

β-structural positions are assigned to nodes as follows: for each node x that is β-reachable from a not-β-covered back-edge $e(u, v)$, a β-position is assigned to x for each such $e(u, v)$. In order to define the constituents of a β-position, we define β-PONs and β-RONs next.

DEFINITION 6.4 (β-PONS, β-RONS). *Let $e(x, w)$ and $e(y, z)$ be back-edges in graph $\mathcal{G}(V, E)$. Let $e(x, w)$ be not-β-covered and $e(y, z)$ be β-covered by $e(x, w)$. The β-post-order (β-pre-order) numbers of x, w, y and z with respect to $e(x, w)$, denoted by $o_x^{\beta:x \rightarrow w}$, $o_w^{\beta:x \rightarrow w}$, $o_y^{\beta:x \rightarrow w}$, and $o_z^{\beta:x \rightarrow w}$, respectively ($q_x^{\beta:x \rightarrow w}$, $q_w^{\beta:x \rightarrow w}$, $q_y^{\beta:x \rightarrow w}$, and $q_z^{\beta:x \rightarrow w}$, respectively) satisfy the following conditions:*
(1) $o_x^{\beta:x \rightarrow w} = o_x^{\tau}; q_x^{\beta:x \rightarrow w} = q_x^{\chi}$, if x is a χ-node, $q_x^{\beta:x \rightarrow w} = q_x^{\tau}$; else
(2) $o_w^{\beta:x \rightarrow w} < o_x^{\beta:x \rightarrow w}$; $q_w^{\beta:x \rightarrow w} > q_x^{\beta:x \rightarrow w}$;
(3) $o_z^{\beta:x \rightarrow w} < o_y^{\beta:x \rightarrow w} < o_w^{\beta:x \rightarrow w}$; $q_z^{\beta:x \rightarrow w} > q_y^{\beta:x \rightarrow w} > q_w^{\beta:x \rightarrow w}$.

For a β-node x, the randomized versions of its β-post-order number with respect to $e(x, w)$ $o_x^{\beta:x \rightarrow w}$ and β-pre-order number $q_x^{\beta:x \rightarrow w}$ are denoted by $p_x^{\beta:x \rightarrow w}$ and $r_x^{\beta:x \rightarrow w}$, respectively and are defined in the same manner as RPON is defined for PON and RRON for RON. In order to define α-signatures for non-acyclic graphs, the χ-pre-order-numbers for χ-nodes are defined below with an additional constraint related to back-edges, but without changing their properties as defined in Definition 5.2.

DEFINITION 6.5 (χ-RONS). *Let x be a χ-node in a connected directed graph $\mathcal{G}(V, E)$. Let $e(w, x)$ and $e(x, y)$ be edges in $\mathcal{G}$ "such that neither of them is a back-edge"[5]. The χ-pre-order number of x and w denoted by q_x^{χ} and q_w^{χ}, respectively, satisfy the following conditions:(1) $q_x^{\chi} > q_x^{\tau}$; (2) if w is a χ-node, $q_x^{\chi} > q_w^{\chi}$, else $q_x^{\chi} > q_w^{\tau}$; and (3) $q_x^{\chi} < q_y^{\tau}$.*

DEFINITION 6.6 (β-STRUCTURAL POSITION). *Let x be a node, which is* β-reachable *from y over the not-β-covered back-edge $e(y, z)$ in a graph $\mathcal{G}(V, E)$. The β-structural position of x with respect to $e(y, z)$ denoted by $\theta_x^{\beta:y \rightarrow z}$, is defined as the pair of its β-RPON $p_x^{\beta:y \rightarrow z}$ and β-RRON $r_x^{\beta:y \rightarrow z}$, that is, $\theta_x^{\beta:e(y,z)} = (p_x^{\beta:y \rightarrow z}, r_x^{\beta:y \rightarrow z})$. Similarly, the β-structural position of y with respect to $e(y, z)$ denoted by $\theta_y^{\beta:y \rightarrow z} = (p_y^{\beta:y \rightarrow z}, r_y^{\beta:y \rightarrow z})$.*

If a node is a β-node, it is also a τ-node or a χ-node. However if a node is a χ-node it is not a τ-node and vice versa. The authentication units associated with β-nodes, *β-reachable* nodes and back-edges are called backward signatures (β-authentication unit in short). Each node has a α-authentication unit. Moreover, if a node x is *β-reachable* from a not-β-covered back-edge $e(y, z)$ in $\mathcal{G}$, it has a β-authentication unit $\Psi_x^{\beta:y \rightarrow z}$.

DEFINITION 6.7 (β-AUTHENTICATION UNIT OF A NODE). *Let x be a node and $e(y, z)$ be a not-β-covered edge in a weakly connected directed graph $\mathcal{G}(V, E)$ such that x is* β-reachable *from y over $e(y, z)$. The β-structural authentication unit of x, denoted by $\xi_x^{\beta:y \rightarrow z}$, is defined as $\xi_x^{\beta:y \rightarrow z} = \mathcal{H}(\theta_x^{\beta:y \rightarrow z} \| C_x)$. The β-structural signature of the β-node y, denoted by $\xi_y^{\beta:y \rightarrow z}$, is defined as $\xi_y^{\beta:y \rightarrow z} = \mathcal{H}(\theta_y^{\beta:y \rightarrow z} \| C_y)$.*

DEFINITION 6.8 (SIGNATURE OF A GRAPH). *Let ω be a cryptographically secure random. Let $\mathcal{H}$ denote a one-way cryptographic hash function. Let $\underline{\mathtt{sk}}$ be the private key and $\mathtt{P}$ be defined as in an aggregate signature framework (Section 4.2). Let $\omega_{\mathcal{G}}$ be defined as $\mathcal{H}(\omega)$. Let S_x be the set of not-β-covered edges from each of which x is β-reachable. The structural signature of a graph $\mathcal{G}(V, E)$, denoted by $\Psi_{\mathcal{G}(V,E)}$, is defined as follows:*

$$\Psi_{\mathcal{G}(V,E)} \leftarrow e(\mathtt{P}, \underline{\mathtt{sk}}(\omega_{\mathcal{G}} + \sum_{\mathtt{x} \in \mathtt{V}} \xi_{\mathtt{x}}^{\alpha} + \sum_{\mathtt{x} \in \mathtt{V}} \sum_{\mathtt{e(y,z)} \in \mathtt{S_x}} \xi_{\mathtt{x}}^{\beta:\mathtt{y} \rightarrow \mathtt{z}})). \quad (4)$$

6.1 Signing a Graph

The steps that the trusted owner Alice uses to sign and share a graph $\mathcal{G}(V, E)$ with Bob are presented in Figure 6.1. How the set of not-β-covered back-edges are determined is stated by Lemma 6.9.

LEMMA 6.9. *In a graph $\mathcal{G}$, the β-node u that has the lowest o_u^{τ} among all β-nodes is such that a back-edge $e(u, v)$ in $\mathcal{G}$ is* not-β-covered *by any other back-edge.*

PROOF. Let w be a node and $e(w, x)$ be a back-edge in $\mathcal{G}$ such that $e(u, v)$ is β-covered by $e(w, x)$. In such a case, by the property of post-order numbers, the depth-first traversal would assign a post-order number o_w^{τ} to w such that $o_w^{\tau} < o_u^{\tau}$, which is a contradiction. □

6.2 Distribution

In this section, we show how to provide an optimal distribution technique that sends only $O(1)$ authentication units to the user. $\mathcal{D}$ sends the following items to $\mathcal{B}$: $G'(V', E')$, the signature of the graph $\Psi_{\mathcal{G}}$, and an authentication unit of the subgraph $\xi_{G'}$, which is computed as in the algorithm given in Figure 6.2. The set of verification units $\mathcal{VO}(G'(V', E')) = \{\Psi_{\mathcal{G}}, \xi_{G'}\}$.

[5] Difference from Definition 5.2 is "such that neither of them is a back-edge".

```
1. Forward pass on graph $\mathcal{G}(V, E)$.
    (a) Execute a depth-first manner traversal on $\mathcal{G}(V, E)$.
    (b) For each node $x \in V$, compute $o^{\tau}_{x}$ and $q^{\tau}_{x}$.
    (c) If $e(w, x) \in E$, and
        i. If $((o^{\tau}_{w} > o^{\tau}_{x})$ and $(q^{\tau}_{w} > q^{\tau}_{x})$, then mark $e(w, x)$ as a cross-edge and $x$ as a $\chi$-node.
        ii. If $(o^{\tau}_{w} < o^{\tau}_{x})$ and $(q^{\tau}_{w} > q^{\tau}_{x})$, then mark $e(w, x)$ as a back-edge and $w$ as a $\beta$-node.
    (d) For each $\chi$-node, compute its $q^{\chi}_{.}$
2. Backward pass on graph $\mathcal{G}(V, E)$.
    (a) Let $V^{\beta}$ be the set of all back-nodes in $\mathcal{G}$.
    (b) Let $y \in V^{\beta}$ such that $o^{\tau}_{y} < o^{\tau}_{u}, \forall\, u \in (V^{\beta} - \{y\})$
    (c) For each back-edge $e(y, z)$ from $y$, execute a depth-first traversal of the graph from $z$.
    (d) For each node $w$ that is $\beta$-reachable from $y$ over $e(y, z)$,
        i. If $w$ is not visited earlier from $y$, compute $o^{\beta:y \rightarrow z}_{w}$ and $q^{\beta:y \rightarrow z}_{w}$.
        ii. Else compute $o^{\beta:y \rightarrow z}_{w}$ and $q^{\beta:y \rightarrow z}_{w}$, such that $o^{\beta:y \rightarrow z}_{w}$ is less than and $q^{\beta:y \rightarrow z}_{w}$ is larger than the respective values computed in the previous visit.
        iii. if $w$ is a back-node, $V^{\beta} \leftarrow V^{\beta} - \{\text{w}\}$.
    (e) Goto step $(a)$ until $V^{\beta}$ is empty.
3. For each node $x \in V$, transform the traversal numbers into randomized traversal numbers, so that they preserve the order.
4. For each node $x \in V$ that is a $\chi$-node, assign $(p^{\tau}_{x}, r^{\chi}_{x})$ to $x$ as its structural position $\theta^{\chi}_{x}$.
5. For each node $x \in V$ that is not a $\chi$-node, assign $(p^{\tau}_{x}, r^{\tau}_{x})$ to $x$ as its structural position $\theta^{\tau}_{x}$.
6. For each node $x \in V$, and for each not-$\beta$-covered back-edge $e(y, z)$ such that $x$ is $\beta$-reachable from $y$ over $e(y, z)$, $\theta^{\beta:y \rightarrow z}_{x} \leftarrow (p^{\beta:y \rightarrow z}_{x}, r^{\beta:y \rightarrow z}_{x})$.
7. For each node $y \in V$ such that $e(y, z)$ is a not-$\beta$-covered back-edge, $\theta^{\beta:y \rightarrow z}_{y} \leftarrow (p^{\beta:y \rightarrow z}_{y}, r^{\beta:y \rightarrow z}_{y})$.
8. For each node $x \in V$:
    (a) If $x$ is a $\chi$-node, $\theta_{x} \leftarrow \theta^{\chi}_{x}$ else $\theta_{x} \leftarrow \theta^{\tau}_{x}$.
    (b) Compute the $\alpha$-signature $\xi^{\alpha}_{x} \leftarrow \mathcal{H}(\theta_{x} \| C_{x})$.
    (c) If $x$ is $\beta$-reachable from a not-$\beta$-covered back-edge $e(y, z)$, compute the $\beta$-signature $\xi^{\beta:y \rightarrow z}_{x} \leftarrow \mathcal{H}(\theta^{\beta:y \rightarrow z}_{x} \| C_{x})$.
    (d) If $x$ is such that $e(x, w)$ is a not-$\beta$-covered back-edge, compute the $\beta$-signature $\xi^{\beta:x \rightarrow w}_{x} \leftarrow \mathcal{H}(\theta^{\beta:x \rightarrow w}_{x} \| C_{x})$.
9. Choose a secure random $\omega$; Let $\omega_{\mathcal{G}} \leftarrow \mathcal{H}(\omega)$.
10. Compute the signature of the graph $\mathcal{G}(V, E)$ as in Eq. 4.
```

Figure 5: Algorithm to sign a graph.

6.3 Authentication

A user $\mathcal{B}$ receives (1) a subgraph $G''(V'', E'')$, (2) the signature of the graph $\Psi_{\mathcal{G}}$, and (3) the authentication unit of the subgraph - ξ from the distributor $\mathcal{D}$. The user verifies the authenticity of the contents as well as the structural position of the nodes in G'' by using the aggregate signature $\Psi_{G'(V',E')}$. The process for verification of contents is same as the process described in Section 5.3. $\mathcal{B}$ evaluates Equation 2, and then Equation 3.

Verification for Structural Relations

1. Execute a depth-first traversal of G''. Let the structural position of each node x be denoted by $\theta_x = (p_x, r_x)$.

```
1. $\vartheta \leftarrow \emptyset$.
2. Let $\Gamma$ be a set: $\Gamma \leftarrow \{\langle x, \theta_x, \xi_x \rangle | \forall x \in V, \theta_x$ is a structural position of $x$, and $\xi_x$ refers to the authentication unit of $x$ associated with $\theta_x\}$.
3. If no edge in $G'$ is a back-edge in $\mathcal{G}$, For each node $x$ in $G'$,
    (a) If $x$ is a cross-node, $\theta_x \leftarrow \theta^{\chi}_{x}$. Else $\theta_x \leftarrow \theta^{\tau}_{x}$.
    (b) $\vartheta \overset{\cup}{\leftarrow} \{\langle x, \theta_x, \xi^{\alpha}_{x} \rangle\}$.
   Else proceed to the next step.
4. For each $e(x, y)$ in $G'$ that is a not-$\beta$-covered back-edge in $\mathcal{G}$,
    (a) For each node $z$ in $G'$ such that $z$ is either $x$ or is $\beta$-reachable from $x$ over $e(x, y)$ in $G'$, $\vartheta \overset{\cup}{\leftarrow} \{\langle z, \theta^{\beta:x \rightarrow y}_{z}, \xi^{\beta:x \rightarrow y}_{z} \rangle\}$. Flag $z$ as visited.
   Flag $e(x, y)$ as visited.
5. Let $E^{\beta}_{0} \leftarrow \{e(x, y) | e(x, y)$ is in $G'$ and is not visited, $e(x, y)$ is a back-edge in $\mathcal{G}$ and is not-$\beta$-covered in $G'\}$.
6. For each $e(x, y) \in E^{\beta}_{0}$,
    (a) Let be the back-edge $e(u, v)$ in $\mathcal{G}$ but not in $G'$ such that $e(x, y)$ is $\beta$-covered by $e(u, v)$ in $\mathcal{G}$.
    (b) For each node $z$ such that $z$ is in $G'$, $z$ is not visited, $z$ is either $x$ or $\beta$-reachable from $x$ over $e(x, y)$ in $G'$, $\vartheta \overset{\cup}{\leftarrow} \{\langle z, \theta^{\beta:u \rightarrow v}_{z}, \xi^{\beta:u \rightarrow v}_{z} \rangle\}$. Flag $z$ as visited.
7. For each node $w$ that is not visited and has a simple path $w \ldots \rightarrow x$ in $G'$, If $w$ is $\beta$-reachable from $u$ over $e(u, v)$, and $e(u, v)$ is in $\mathcal{G}$, but not in $G'$, $\vartheta \overset{\cup}{\leftarrow} \{\langle w, \theta^{\beta:u \rightarrow v}_{w}, \xi^{\beta:u \rightarrow v}_{w} \rangle\}$. Flag $w$ as visited.
8. For each node $w$ in $G'$ that is not visited,
   If $w$ is a $\chi$-node, $\vartheta \overset{\cup}{\leftarrow} \{\langle w, \theta^{\chi}_{w}, \xi^{\alpha}_{w} \rangle\}$. Else $\vartheta \overset{\cup}{\leftarrow} \{\langle w, \theta^{\tau}_{w}, \xi^{\alpha}_{w} \rangle\}$.
9. Compute $\xi_{G'(V',E')}$ as the aggregate signature of the nodes in $V'$: $\xi_{G'} = \sum_{\langle x, \theta_x, \xi_x \rangle \in \Gamma - \vartheta} \xi_x$.
```

Figure 6: Algorithm to distribute a subgraph.

2. For edge $e(x, z)$, if $(p_x < p_z)$ **and** $(r_x > r_z)$, then if x is not a β-node in G'', then $e(x, z)$ is not a back-edge (there is no cycle in G'' involving x and z), so the parent-child relationship between x and z is incorrect.

3. Else if $(p_x \leq p_z)$ **or** $(r_x \geq r_z)$, then parent-child relationship between x and z is incorrect.

Using the depth-first traversal carried out during authentication of structural relations, it is easy to determine if x is a β-node or $e(x, z)$ is a β-edge in $G''(V'', E'')$ (Definition 1.1). We just need to initialize a pair of new pre-order and post-order numbers of each node to -1. These numbers are different than the ones that have been received as signature items. For each edge $e(x, z)$, if the post-order number of z is already computed (so that its value is some number greater than -1), then $e(x, z)$ is not a back-edge, which implies that x is not a β-node. There is no ordering among siblings (of a DFT) in a non-acyclic graph (cannot be topologically-ordered).

7. SECURITY ANALYSIS

This section states the lemmas about the security of the proposed schemes in terms of authenticity and non-leakage properties. First of all, the proposed scheme supports *non-repudiation* primarily because the aggregate signature scheme supports non-repudiation. Moreover, the proposed schemes for DAGs and graphs with cycles are "existentially unforgeable under adaptive chosen-message attack". In what follows, we provide proof sketches for the security of the proposed schemes.

LEMMA 7.1 (AUTHENTICATION (DAGS)). *Given that $\mathcal{H}$ is a cryptographic hash function and that aggregate signatures are secure, any authenticity violation of a graph can be detected by using the structural signature scheme for DAGs.*

PROOF SKETCH. Let $\xi_{G'(V',E')}$ and $\Psi_{G'(V',E')}$ be the authentication units received by a $\mathcal{B}$, who receives the $G'(V', E')$, subgraph of $\mathcal{G}(V, E)$. Suppose that the authenticity verification scheme in Section 5.3 authenticates a graph, $G''(V'', E'')$ different from G_δ, to $\mathcal{G}$, using $\xi_{G'(V',E')}$ and $\Psi_{G'(V',E')}$. By Definitions of χ-RONs, τ-PONs and τ-RONs, the relationship between the nodes in G'' must be identical to that of G'(Section 5.3), otherwise the randomized traversal numbers are not secure, which is a contradiction. Contents of G'' would be authenticated iff Eq. 3 is satisfied for G'', which implies that $\mathcal{H}$ incurred a collision, and the aggregate signature scheme and $\mathcal{H}$ are not secure, which is a contradiction, under the random oracle hypothesis [19] and hardness of computational Diffie-Hellman problem [6]. Thus the lemma is proven. □

LEMMA 7.2 (NON-LEAKAGE (DAGS)). *Given that $\mathcal{H}$ is a cryptographic hash function and that aggregate signatures are secure, the structural signature of a DAG do not lead to any leakage of any extraneous information.*

PROOF SKETCH. A user receiving a sub-DAG $G'(V', E')$ of DAG $\mathcal{G}$ should not be able to infer any extraneous information of $\mathcal{G}$ from (1) the signature $\Psi_{\mathcal{G}}$, (2) set of verification units $\mathcal{VO}(G')$, and (3) the τ- and χ-structural positions of the nodes in G'. Due to the properties of Bilinear maps and aggregate signatures, (1) and (2) do not leak any information (i.e., the probability of leakage is negligible) [6]. We now would prove that (3) does not leak. (Forward-edges:) The τ-structural positions of nodes in a forward-edge are identical to the rules of tree-edge (Definition 1.1); therefore an edge $e(x, y)$ that is a forward-edge in $\mathcal{G}$ but not a forward-edge in G' would be verified by the user as a tree-edge. (Cross-edges:) By Definitions 5.2 and 5.4, a χ-node x has a structural position such that r_x^χ is larger than the r_w^τ for each w that has a cross-edge incident on x. By Definition 1.1, the user would only know that $e(w, x)$ is a tree-edge and thus cannot learn whether it is a cross-edge (Definition 1.1). Thus the lemma is proved. If the signature of the DAG and the authentication unit of the subgraph leak extraneous information, then $\mathcal{H}$ incurred a collision, and the aggregate signature scheme and $\mathcal{H}$ are not secure, which is a contradiction, under the random oracle hypothesis [19] and hardness of computational Diffie-Hellman problem [6]. Thus the lemma is proven. □

LEMMA 7.3 (AUTHENTICATION (GRAPHS WITH CYCLES)). *Given that $\mathcal{H}$ is a cryptographic hash function and that aggregate signatures are secure, any authenticity violation of a graph can be detected by using the structural signature scheme for connected directed graphs.*

PROOF SKETCH. Let $\mathcal{G}(V, E)$ be a directed connected graph and x be a node in it.

Content authenticity: Any compromise of the content C_x or the structural position (either τ, χ, or β) of a node x in $\mathcal{G}$ would invalidate the structural authentication unit ξ_x, which is a hash of a message that contains C_x and structural position of x as defined by Definitions 5.4, and 6.6, unless the hash function $\mathcal{H}$ encounters a collision, which contradicts our assumption and the random oracle hypothesis. Moreover, if a node/edge is added to or dropped from the received subgraph $G'(V', E')$, it would invalidate the received authentication unit of the subgraph $\xi_{G'(V',E')}$. If $\xi_{G'(V',E')}$ does not get invalidated, then under the random oracle hypothesis, the $\mathcal{H}$-function is not secure or the aggregate signature scheme is not secure, which is a contradiction.

Structural authenticity: The signature of a node in a graph is a forward-signature, if it is not reachable from a back-edge. Such a signature is with respect to the DFT obtained from a forward traversal of the graph. Any unauthorized re-ordering between two or more nodes (violation of structural integrity) in such DFT can be detected using the randomized traversal numbers [21]. If a node is a β-node or is reachable from a back-edge, then such a node also belongs to the DFT obtained from a depth-first traversal carried out from β-node(s) over back-edges. Any re-ordering would be detected here as well.

Suppose x belongs to another tree $\mathcal{G}'$, but claimed to belong to $\mathcal{G}$. By the arguments similar to the one in the proof of Lemma 7.1, it is not possible. □

LEMMA 7.4 (NON-LEAKAGE (GRAPHS WITH CYCLES)). *Given that $\mathcal{H}$ is a cryptographically secure hash function and aggregate signatures are secure, the structural signatures for connected directed graphs do not lead to any leakage of any extraneous information.*

PROOF SKETCH. A user receiving a subgraph $G'(V', E')$ of graph $\mathcal{G}$ should not be able to infer any extraneous information of $\mathcal{G}$ from (1) the signature $\Psi_{\mathcal{G}}$, (2) set of verification units $\mathcal{VO}(G')$, and (3) the τ-, χ-, and β-structural positions of the nodes in G'. Due to the properties of Bilinear maps and aggregate signatures, (1) and (2) do not leak any information (i.e., the probability of leakage is negligible) [6]. We now would prove that there is no leakage due to (3).

In order to be confidentiality-preserving, an authentication scheme must not leak any extraneous information about (a) cross-edges, (b) forward-edges, (c) back-edges. Following Lemmas 7.2, the authentication scheme for connected directed graphs does not leak any information about (a) and (b). The scheme does not leak any information via the back-edges or β-structural positions, which is proven as follows.

Let $e(x, w)$ be a *not-β-covered* back-edge in graph $\mathcal{G}(V, E)$ (e.g., $e(g_{14}, g_9)$ in Figure 2). By Definitions 6.4 and 5.4, the β-node x has a β-structural position such that $p_x^{\beta:}$ and $r_x^{\beta:}$ are larger and smaller than the $p_w^{\beta:x\rightarrow w}$ and $r_w^{\beta:x\rightarrow w}$, respectively. By Definition 1.1, the user would only verify that $e(x, w)$ is a tree-edge and cannot learn whether it is a back-edge.

The following cases arise when a user is authorized to access a connected subgraph that includes a back-edge $e(x, w)$, which is *not-β-covered* in the subgraph; the subgraph may also include: (I) Nodes reachable from w, (II) Nodes reachable from x and (III) Nodes from which x is reachable.

Case I: Nodes reachable from w: The user is authorized to access the back-edge $e(x, w)$ and the edge $e(w, y)$ (e.g., $e(g_9, g_{11})$ in Figure 2). By Definition 6.2, y is *β-reachable* from x. Further by Definitions 6.4 and 5.4, $p_x^{\beta:x\rightarrow w}$ and $r_x^{\beta:x\rightarrow w}$ are larger and smaller than the $p_y^{\beta:x\rightarrow w}$ and $r_y^{\beta:x\rightarrow w}$, respectively. By Definition 1.1, the user would only verify that an edge $e(w, y)$ is a tree-edge and cannot learn from it whether $e(x, w)$ is a back-edge.

Case II: Nodes reachable from x: The user is authorized to access the back-edge $e(x, w)$ and the edge $e(x, y)$ (e.g., $e(g_{14}, g_{18})$ in Figure 2).

- y is not *β-reachable* and x is not *β-reachable*: By sending τ-position and τ-signatures for both x and y and β-position and β-signatures for w, the edges would be verified as tree-edges. This is because, $p_x^\tau = p_x^{\beta:x\rightarrow w}$ and $r_x^\tau = r_x^{\beta:x\rightarrow w}$ (Definition 6.4) and thus the τ-signature for x are same as its β-signature.

- y is not *β-reachable* and x is *β-reachable*: Not possible by the definition of the notion of *β-reachable*.

- y is *β-reachable* and x is not *β-reachable*: By sending τ-position and τ-signatures for y and β-position and β-signature for x and w, both the edges $e(x, w)$ and $e(y, x)$ are verified by the user as tree-edges, thus hiding the original type of $e(x, w)$. This is because, since x is not *β-reachable*, by Definition 6.4, $p_x^\tau = p_x^{\beta:x\rightarrow w}$ and $r_x^\tau = r_x^{\beta:x\rightarrow w}$. Thus and by the properties of post-order and pre-order numbers, since $e(y, x)$ is a tree-edge in $\mathcal{G}$, $p_y^\tau > p_x^{\beta:x\rightarrow w}$ and $r_y^\tau < r_x^{\beta:x\rightarrow w}$.

- y is *β-reachable* and x is *β-reachable*: By sending β-positions and β-signatures for both y, x and w, the edges would be verified as tree-edges. This is because, by Definition 6.4, $p_y^{\beta:x\rightarrow w} > p_x^{\beta:x\rightarrow w} > p_w^{\beta:x\rightarrow w}$ and $r_y^{\beta:x\rightarrow w} < r_x^{\beta:x\rightarrow w} < r_w^{\beta:x\rightarrow w}$.

Case III: Nodes from which x is reachable: The user is authorized to access the back-edge $e(x, w)$ and the edge $e(y, x)$ (e.g., $e(g_{11}, g_{14})$ in Figure 2). Four cases arise:

- y is not *β-reachable* and x is not *β-reachable*: Proof is identical to this condition in Case-II.

- y is not *β-reachable* and x is *β-reachable*: By sending τ-positions and τ-signatures for y and β-positions and β-signatures for x and w, the edges would be verified as tree-edges. This is because, for an

x that is *β-reachable* and a β-node, $p_x^{\beta:x\to w} < p_x^\tau$ and $r_x^{\beta:x\to w} > r_x^\tau$. Since $e(y,x)$ is an edge in the graph, $p_x^\tau < p_y^\tau$ and $r_x^\tau > r_y^\tau$.

- y is *β-reachable* and x is not *β-reachable*: Proof is identical to this condition in Case-II.
- y is *β-reachable* and x is *β-reachable*: Proof is identical to this condition in Case-II.

The above arguments can be easily extended to multiple *not-β-covered* back-edges in the graph. Consideration of one *not-β-covered* back-edge takes care of all other back-edges that are it covers. So the argument also extends to multiple *β-covered* and *not-β-covered* back-edges.

Suppose that a user Bob has access to G', a subgraph in $\mathcal{G}$. Bob receives the structural signature of $\mathcal{G}$, the node signatures of G' and their structural positions. Any leakage would be a direct leakage through these information or an inference from them.

Direct leakage: Clearly (as per Definitions 5.6, 5.5, 6.8 and 6.7, and the protocols specified in Sections 5.2 and 6.2) Bob does not need the authentication unit as it is of any node u that is in $\mathcal{G}$ but not in G'. He therefore does not need to know any of the structural relationships and structural ordering that exist in $\mathcal{G}$, but not in G'. Therefore none of (3), (4), (5) and (6) is directly leaked to Bob; he does not learn any extra information from the integrity verification process.

Indirect leakage through the signature of the graph and authentication unit for G': Under the Random Oracle Hypothesis and the hard-ness of Computational Diffie-Hellman problem, the structural signature of the tree reveals neither (3) the signature of u, nor (4) the existence of u. Similarly, the structural signature of a node leaks neither (3) nor (4). Therefore (5) - the structural relations (edges or paths) and (6) - the structural order among nodes in G' and u are also not revealed by the signatures.

In addition, the structural positions of the nodes in G' do not reveal any information [21]: because the probability of inference (and leakage) about (2) - the existence of node u between two immediate siblings from such randoms is negligible[6] ($\frac{1}{2^k}$) [19], where k is the security criteria - the number of bits as the output of hash function $\mathcal{H}$. Therefore structural positions of nodes cannot be used to determine the structural signature of u. Since (3) and (4) cannot be inferred from the RPON's and RRON's, (5) and (6) also cannot be inferred from the structural position of a node. □

8. COMPLEXITY AND PERFORMANCE RESULTS

In the following sections, we analyze the complexity and performance of the proposed schemes.

8.1 Complexity

Signature Generation Complexity. The pre-order and post-order numbers can be generated by a single traversal of the graph $\mathcal{G}(V,E)$. The traversal complexity is thus $O(|V| + |E|)$. In the signing scheme of graphs (Section 6.1), a graph is traversed once forward, a number of times backward, and once for computing the signatures in the end. The number of backward traversals is the number of *not-β-covered* back-edges d, which is in $[0, |V| - 1]$ (0 in case of DAGs and $(|V| - 1)$ in case of complete graphs). In case of signing DAGs, the scheme requires only two traversals (no backward traversal is required here). Therefore, the complexity for signing a DAG is $O(|V| + |E|)$. In case of graphs with cycles, the scheme requires d number of backward traversals, where d is the number of not-β-covered back-edges.

The number of authentication items that need to be *computed*, and *stored* is $(|V|*(2 + 2d))$ as explained below by assuming that the sizes of the secure random number and the cryptographic hash are identical. In the worst case (when the graph is a complete graph), each node is a β-node; so each node has one forward and d backward positions. Each structural position involves two secure random numbers. Therefore, the *storage complexity* of structural signatures is $O(|V|$*d$)$.

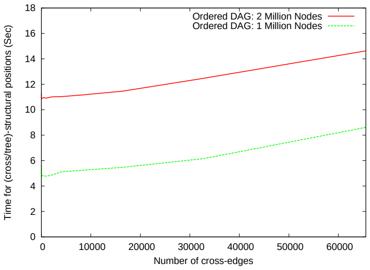


Figure 7: Average time in seconds to compute the χ- and τ-structural positions Vs. the number of cross-edges in each ordered-DAG.

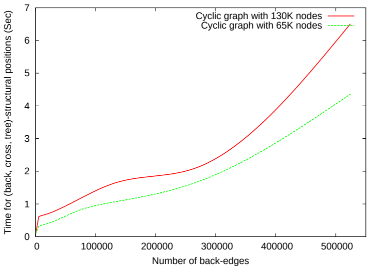


Figure 8: Average time in seconds to compute the β-,χ- and τ-structural positions Vs. the number of back-edges in a graph with cycles.

Distribution Complexity. Let the signed subgraph that needs to be shared with a user be $G'(V', E')$. In the worst case ($G'(V', E')$ is a complete graph), the distributor has to send $(2 * |V'|)$ number of signature items as follows. Each node in such a subgraph is a β-node; thus the β-structural position (two authentication items) of each node is sent to the user. Therefore the distribution complexity of structural signatures is $O(|V'|)$.

Integrity Verification Complexity. The procedure for the verification of content integrity incurs a cost linear in terms of the size of the subgraph $G'(V', E')$ received, that is, $O(|V'| + |E'|)$. It accounts for one hashing for each node. The cost of verification of structural integrity is also linear in terms of the size of the subgraph received, that is, $O(|V'| + |E'|)$ as the cost of comparison of randomized traversal numbers is constant.

8.2 Performance Results

We implemented the two schemes in Java 1.6 and JCA 6.0 (Java Cryptography Architecture) APIs. The experiments were carried out on a desktop with the following specification: 64-bit Linux (Ubuntu 8.10) on Intel Core 2 Duo CPU with 4GB RAM. For signing using aggregate signatures and bilinear maps, we used the C implementation of bilinear maps[7]. Further, we used Java Native Interface (JNI) to integrate this C implementation with our Java implementation.

The performance is measured as: time to compute the χ-, τ-structural positions and β-structural positions (for graphs) versus the number of cross-edges, back-edges in the DAG and the graphs. The other performance metric that we have evaluated the schemes against is time taken for signing and verification.

Time to Compute Structural Positions: The process of computing appropriate structural positions of each node in either a DAG or a

[6] In cryptography, a technique that leads to only negligible leakage is provably non-leaking [19].

[7] http://crypto.stanford.edu/pbc

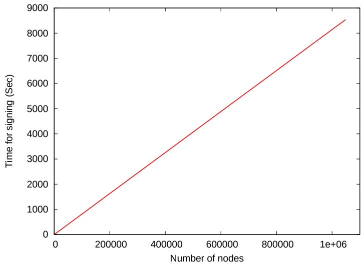


Figure 9: Average time to sign the nodes Vs. the number of nodes (authentication units) in the graph.

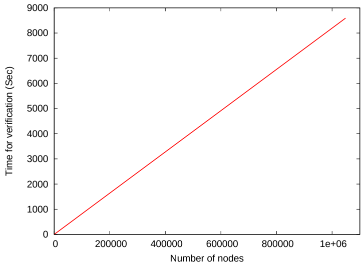


Figure 10: Average time to verify the nodes Vs. the number of nodes (authentication units) in the graph.

graph is part of the one-time process of signing. Figure 7 shows that the time for computation of χ- and τ-structural positions for DAGs is very efficient (linear). For DAGs of size about 2 million nodes, it takes only 14 seconds. Figure 8 shows that the time to compute all the structural positions for all the nodes in a graph with respect to the number of back-edges in the graph is also highly efficient - for more than 0.5 million back-edges, it takes 6.5 seconds.

Time to Sign: As evident from the plot in Figure 9, the time taken to sign graphs of size as many as one million nodes using aggregate signatures is linear with respect to the number of nodes, which corroborates our complexity analysis. It is evident that the performance is better for graphs with cycles than what we expected in the complexity analysis (Section 8.1). Note that aggregate signatures based on bilinear mapping and elliptic curves are inherently expensive.

Time to Verify: From Figure 10, it is evident that the time taken to verify the contents and authentication units of such graphs against the received signature and verification units is *linear* with respect to the number of nodes in the graph.

9. RELATED WORK

To the best of our knowledge, this is the first solution to the problem of "authentication of graphs *without leaking*". Two works closest to the proposed solution are as follows. The first one is by Kundu and Bertino [21], in which, the problem of integrity assurance of trees without leaking is addressed by use of "structural signatures"; however, it can not be trivially applied to graphs, which are much more complex structures than trees and graphs with cycles cannot be topologically sorted.

The second work is by Martel et al. [25], who proposed an authentication technique for data structures - directed acyclic graphs referred to as "Search DAGs" in third party distribution frameworks. However their technique uses Merkle hash technique [26], which leaks information during authentication [7, 21]. Moreover the "Search DAGs" technique only covers DAGs, not the general directed graphs. Querying, management [10] and mining [11] of graph-structured data as well as privacy-preserving graph publishing and mining techniques [23, 34] have recently emerged as important topics in both academia and industry. Confidentiality and authenticity are important requirements in secure data management and publishing [4, 5].

The authentication scheme for completeness of query results proposed by Pang et al [28] leaks even in the simple case of "greater than" predicates. In their scheme, if a_3, a_4, and a_5 satisfy a the "greater than" predicate among elements $a_1, a_2, \ldots, a_{10}$, then the user uses extra information *about* a_2 and of a_6 in order to authenticate the results; such extra information lead to the leakage about the existence and structural position of these two elements in the data object, which as we have seen earlier could be sensitive information or could lead to inference of sensitive information. Goodrich and Tamassia [12] have used skip lists and one-way accumulators for authentication of dictionaries. Goodrich et al [14, 13] proposed a framework for authenticated graph searching and query processing. Some of the other notable authentication schemes in the literature are by Li et al [22], Mouratidis et al [27], Pang and Mouratidis [29], and by Pang and Tan [30]. Even though each of these techniques provides authenticity, yet it leaks (primarily because, they rely on Merkle hash technique, which inherently leaks). Moreover, none of these works (including [28]) address the problem of authentication of graphs.

Batch verification of RSA signatures [3, 15] can be thought to support secure signature aggregation. However, such a scheme is insecure (Hwang et al [17] first showed the forgery attack that can be carried out on it) and it cannot be used to carry out secure signature aggregation. Later in another paper [16] Hwang et al proposed a new scheme overcoming the weakness of Harn's scheme. However, Bao et al [2] have shown that even this scheme is not secure. The scheme proposed in [2] has also not been proven to be secure. Given that there are so many assumptions essential to build a strong RSA scheme and so many attacks are possible on RSA [31], and the fact that batch verification of RSA signatures have not yet shown to be secure, an authentication scheme using RSA signatures would not be secure. Rather the aggregate signatures by Boneh et al [6] based on bilinear maps, which is proven to be secure, is what should be used for signature aggregation and/or authentication purposes.

10. CONCLUSIONS AND FUTURE WORK

The problem of how to verify authenticity of data without leakage is an important problem in secure data publishing [7, 1, 21]. Since authentication is a much stronger security requirement, integrity assurance schemes such as the one proposed in [21] cannot be directly used to address this problem. In this paper, we proposed two schemes in order to address this problem - one for DAGs and another for graphs with cycles so that such data can be published and distributed while assuring both authenticity and confidentiality. The proposed schemes are based on structure of graphs and aggregate signatures.

Our schemes prevent leakage of information while facilitating authentication of content as well as the structure of graphs. The security of such schemes are based on the security of cryptographic hash functions (random oracles) and aggregate signatures (Computational Diffie-Hellman problem). Our schemes minimize the number of of authentication units that a user must receive in order to be able to carry out authenticity verification - a constant number ($O(1)$). In terms of complexity, the schemes for DAGs and graphs with cycles incur $O(n)$ and $O(n{*}d)$, respectively, for time and storage (n is the number of nodes in the graph). Performance of our schemes on large DAGs and on graphs as large as 0.5-million back-edges nodes show that our scheme is efficient. Further, experimental results show that our technique for graphs with cycles performs linearly - better in actual runtime than the theoretical complexity analysis.

The proposed scheme has a number of applications such as in graph databases of biological and healthcare data, XML graphs, and in authentication and completeness verification of query results. In future, we plan to apply our scheme to some of these areas.

Acknowledgement. The work reported in this paper has been partially supported by the MURI award FA9550-08-1-0265 from the Air Force Office of Scientific Research.

11. REFERENCES

[1] M. Atallah, Y. Cho, and A. Kundu. Efficient data authentication in an environment of untrusted third-party distributors. In *ICDE*, 2008.

[2] F. Bao, C.-C. Lee, and M.-S. Hwang. Cryptanalysis and improvement on batch verifying multiple rsa digital signatures. *Applied Mathematics and Computation*, 172(2), 2006.

[3] M. Bellare, J. A. Garay, and T. Rabin. Fast batch verification for modular exponentiation and digital signatures. In *EUROCRYPT*, 1998.

[4] E. Bertino, L. Khan, R. Sandhu, and B. Thuraisingham. Secure knowledge management: confidentiality, trust, and privacy. *IEEE SMC-A*, May '06.

[5] M. Bishop. *Computer Security: Art and Science*. Addison-Wesley, 2002.

[6] D. Boneh, C. Gentry, H. Shacham, and B. Lynn. Aggregate and verifiably encrypted signatures from bilinear maps. In *Eurocrypt*, 2003.

[7] A. Buldas and S. Laur. Knowledge-binding commitments with applications in time-stamping. In *Public Key Cryptography*, pages 150–165, 2007.

[8] T. H. Cormen, C. E. Leiserson, R. L. Rivest, and C. Stein. *Introduction to Algorithms*. MIT Press, 2001.

[9] P. Devanbu, M. Gertz, C. Martel, and S. G. Stubblebine. Authentic data publication over the internet. *Journal of Computer Security*, 11, 2003.

[10] B. A. Eckman and P. G. Brown. Graph data management for molecular and cell biology. *IBM Journal of Research and Development*, 50(6), 2006.

[11] L. Getoor and C. P. Diehl. Link mining: a survey. *SIGKDD Explor. Newsl.*, 7(2), 2005.

[12] M. Goodrich and R. Tamassia. Efficient authenticated dictionaries with skip lists and commutative hashing, 2000.

[13] M. T. Goodrich, R. Tamassia, and N. Triandopoulos. Efficient authenticated data structures for graph connectivity and geometric search problems. In *Algorithmica*, volume Online, 2009.

[14] M. T. Goodrich, R. Tamassia, N. Triandopoulos, and R. Cohen. Authenticated data structures for graph and geometric searching. In *LNCS*, volume 2612, 2003.

[15] L. Harn. Batch verifying multiple rsa digital signatures. *Electronics Letters*, 34(12), 1998.

[16] M.-S. Hwang, C.-C. Lee, and Y.-L. Tang. Two simple batch verifying multiple digital signatures. *Proceedings of Information and Communications Security*, 2001.

[17] M.-S. Hwang, I.-C. Lin, and K.-F. Hwang. Cryptanalysis of the batch verifying multiple rsa digital signatures. *Informatica*, 11(1), 2000.

[18] R. Jin, Y. Xiang, N. Ruan, and H. Wang. Efficiently answering reachability queries on very large directed graphs. In *SIGMOD*, 2008.

[19] J. Katz and Y. Lindell. *Introduction to Modern Cryptography: Principles and Protocols*. Chapman & Hall/CRC, 1 edition, 2007.

[20] D. E. Knuth. *The Art of Computer Programming*, volume 1. Pearson Education Asia, third edition, 2002.

[21] A. Kundu and E. Bertino. Structural signatures for tree data structures. In *VLDB*, 2008.

[22] F. Li, M. Hadjieleftheriou, G. Kollios, and L. Reyzin. Dynamic authenticated index structures for outsourced databases. In *SIGMOD*, pages 121–132, New York, NY, USA, 2006. ACM.

[23] K. Liu and E. Terzi. Towards identity anonymization on graphs. In *SIGMOD*, 2008.

[24] C. Martel, G. Nuckolls, P. Devanbu, M. Gertz, A. Kwong, and S. G. Stubblebine. A general model for authenticated data structures. *Algorithmica*, 39(1), 2004.

[25] C. Martel, G. Nuckolls, P. Devanbu, M. Gertz, A. Kwong, and S. G. Stubblebine. A general model for authenticated data structures. *Algorithmica*, 39(1), 2004.

[26] R. C. Merkle. A certified digital signature. In *CRYPTO*, 1989.

[27] K. Mouratidis, D. Sacharidis, and H. Pang. Partially materialized digest scheme: an efficient verification method for outsourced databases. *The VLDB Journal*, 18(1):363–381, 2009.

[28] H. Pang, A. Jain, K. Ramamritham, and K.-L. Tan. Verifying completeness of relational query results in data publishing. In *SIGMOD*, 2005.

[29] H. Pang and K. Mouratidis. Authenticating the query results of text search engines. *PVLDB*, 1(1), 2008.

[30] H. Pang and K. Tan. Authenticating query results in edge computing. In *ICDE*, 2004.

[31] D. L. e. P.B. Garrett. Public-key cryptography (baltimore 2003). In *Proc. Symp. Applied Math., AMS*, volume 62, 2005.

[32] H. Tong, C. Faloutsos, and Y. Koren. Fast direction-aware proximity for graph mining. In *KDD*, 2007.

[33] H. Wang and L. V. S. Lakshmanan. Efficient secure query evaluation over encrypted XML databases. In *VLDB*, 2006.

[34] E. Zheleva and L. Getoor. Preserving the privacy of sensitive relationships in graph data. In *PinKDD*, 2007.