Securing Provenance-based Audits

Rocío Aldeco-Pérez
raap06r@ecs.soton.ac.uk
Luc Moreau
l.moreau@ecs.soton.ac.uk

School of Electronics and Computer Science, University of Southampton,
Southampton SO17 1BJ, UK

Abstract. Given the significant increase of on-line services that require personal information from users, the risk that such information is misused has become an important concern. In such a context, information accountability is desirable since it allows users (and society in general) to decide, by means of audits, whether information is used appropriately. To ensure information accountability, information flow should be made transparent. It has been argued that data provenance can be used as the mechanism to underpin such a transparency. Under these conditions, an audit's quality depends on the quality of the captured provenance information. Thereby, the *integrity of provenance information* emerges as a decisive issue in the quality of a provenance-based audit. The aim of this paper is to secure provenance-based audits by the inclusion of cryptographic elements in the communication between the involved entities as well as in the provenance representation. This paper also presents a formalisation and an automatic verification of a set of security properties that increase the level of trust in provenance-based audit results.

1 Introduction

In recent years, an increasing number of on-line services have appeared on the Web, e.g. social networks, governmental sites, on-line selling sites. Most of them offer personalised services that require private personal information from their users. By disclosing personal information, users get access to a wide range of new functionalities, such as recommendations or customisation. But at the same time, they face the risk that their information is misused.

Within this context, it is desirable to allow users to verify whether their information was misused or not. In order to achieve this, information usage should be made transparent so it can be determined later whether the use of such information is appropriate [1]. In other words, the transparency of information usage enables information accountability, a property according to which users can inspect such information usage through a process we refer to as audit.

Weitzner *et al.* have recognised that provenance, which consists of causal dependencies between data and events explaining what contributed to a result in a specific state [2], can be used as a mechanism to achieve information accountability [1]. Thus, if provenance of data is available, processing becomes transparent

since the provenance of data can be audited to decide whether information was used in a proper way.

In order to support such a vision, systems should be made *provenance-aware* [3] by describing all the steps and data derivations involved in their execution, in the form of *process documentation* [4]. Information related to the use of a specific piece of data can be obtained from process documentation by means of a provenance query [5], resulting in a provenance graph, which can be analysed to decide whether information was used appropriately [6].

Against this background, the *integrity of the captured process documentation and the provenance graph derived from it* becomes a vital issue in guaranteeing the quality of a provenance-based audit. Therefore, we address this problem by developing a framework that secures the communication between the entities that are part of a provenance-aware system as well as the provenance query result representation. Specifically, we secure the process documentation created by entities and the result of provenance queries by including cryptographic elements in both.

The contributions of this paper are: *(i)* A secure provenance-aware communication protocol that addresses the integrity of the information exchanged between entities, *(ii)* A specially designed provenance graph that allows us to check the integrity of its content and, *(iii)* An automatic verification of the integrity of a Secure Provenance-based Auditing Architecture, which increases the level of trust in the audit results generated by it.

The remainder of this paper is structured as follows. In Section 2, an overview of the provenance model this work relies upon is presented. In Section 3, to address the integrity property in the communication between entities, the secure communication formalisation, which is related to the Provenance-based Auditing Architecture [6], is presented. In Section 4, to address the integrity property in a provenance graph, we presented the Secured Provenance Graph and an algorithm that checks its integrity. In Section 5, the formal verification of the integrity property of one protocol related to the Provenance-based Auditing Architecture is presented and explained. Finally, Section 6 discusses some related work and Section 7 offers some concluding remarks.

2 Provenance Model Overview

In this section, we present a brief overview of the provenance model and concepts that we use in this paper [7]. We assume that applications capture extra information describing what occurred during their execution. Such extra information is referred to as *process documentation*, which is recorded in a storage component called *Provenance Store*, and queried to obtain the provenance of some data. Process documentation consists of a set of *assertions*, created by the applications' components. These assertions contain a description of the data exchanged by such components and *relationships* expressing causal dependencies between them. A *provenance graph*, which is a view of past execution in which its nodes are data and its edges are labelled with causal relationships' names

[8], can be obtained by querying the Provenance Store. If a provenance graph is later analysed during an audit, it is possible to answer questions regarding past executions of applications. One important assumption of this model is that all participants are not malicious and send provenance information faithfully [3].

The information flow of an auditable provenance-aware system consists of four stages. *(1) Recording* of process documentation in which components make assertions related to the actions they perform and record them in a Provenance Store. *(2) Storage* of process documentation in which assertions are persistently stored in a Provenance Store. *(3) Querying* of process documentation in which process documentation is queried to obtain a provenance graph. *(4) Analysis* of a provenance graph to answer questions regarding the execution of the entities within the system the result of which is an audit report. Requirements such as processing of data is compatible with the purpose for which it was captured and only information to be processed was captured can be checked in the analysis stage. These requirements are not presented in this paper, however, initial work related to that analysis can be found in [6]. In order to guarantee a correct audit report, it is necessary to ensure during all these stages the integrity of the information in which such an analysis is based.

To do that, we create two mechanisms that guarantee the integrity of assertions. One is used in the recording and storage stages, whereas the other is used in the querying and analysis stages. The reason for having two separated mechanisms is to maintain the independence between the creation of distributed assertions and the querying of them in a centralised repository.

In Section 3, we discuss the mechanism used to secure the recording and storage stages. In Section 4, we explain the mechanism used to secure the querying and analysis stages.

3 Securing the Recording and Storage Stage

In this section, we discuss how the assertions created by the entities of a provenance-aware system can be secured. The assertions that are recorded during the recording stage are created from the information exchanged between the participating entities, i.e. during their communication. If this information is maliciously altered then, the quality of the audit can be compromised.

In order to address this problem, we need to secure the messages exchanged between entities and also the assertions that they are sending to the Provenance Store. To achieve that, we add some cryptographic components to both messages and assertions.

To exemplify this process, we formalise a secure communication protocol between the entities that are part of a provenance-based auditing system, specifically we secure the Provenance-based Auditing Architecture presented in [6]. This formalisation process relies on UML sequence diagrams that model a security protocol enabling the involved entities to apply security functions to the transferred data and, thus, protect it. To this end, we use the UML extension

UMLsec, which offers a cryptographic notation for secure systems development [9].

In this formalisation, we assume that entities establish communication by using the TLS (Transport Layer Security) protocol [10], which allows them to verify each others' identities and create a session key used to encrypt/decrypt exchanged messages. We also assume that entities' public and private keys are created and interchanged.

The sequence diagram presented in Section 3.2 models four basic security characteristics: *confidentiality*, *authentication*, *non-repudiation* and *integrity*. Data integrity is the state that exists when computerized data is the same as that in the source documents and has not been exposed to accidental or malicious alteration or destruction [11]. If data integrity is not supported by auditable systems, the quality of an audit report will be affected. Due to the the importance of this property, we only focus on the verification of it; the remaining characteristics can be verified using a similar technique.

3.1 Provenance-based Auditing Architecture

The Provenance-based Auditing Architecture, which is presented in [6] and briefly explained in this section, is depicted in Figure 1. This architecture uses provenance to audit the correct use of private information to later make accountable the involved entities for any information misuse. The architecture is inspired by the roles introduced in the Data Protection Act [12], which places restrictions on how organisations can use personal information that they request from individuals. It contains the actors Data Controller (`DC`), who is the individual or organisation that decides the purpose for which, and the manner in which, personal information is to be processed; the Data Subject (`DS`), who is an individual whose information is held by DC, and the Data Processor, who is an individual or organisation that processes personal information on behalf of DC. In order to make this architecture provenance-aware the Provenance Store (`PS`) component is introduced. This component represents a provenance repository in which provenance information is maintained. Finally, to be able to perform audits, the Auditor actor is introduced. This actor represents an internal or external entity that assesses the use of Data Subject's private information.

Communication's architecture can be structured in three protocols. The *Data Request protocol* represents a request for personal information issued by a Data Controller to a Data Subject. The *Task Request protocol* represents a request for delegating a task issued by a Data Controller to a Data Processor. The *Query Request protocol* represents the querying of the assertions stored in the Provenance Store issued by an Auditor to a Provenance Store. The Data Request and the Task Request protocols model the recording and storage stage. The Query Request protocol models the querying and analysis stage. As Data Request and Task Request are similar protocols [6], we focus on the Data Request protocol in the next section. The query request protocol is introduced in Section 4.

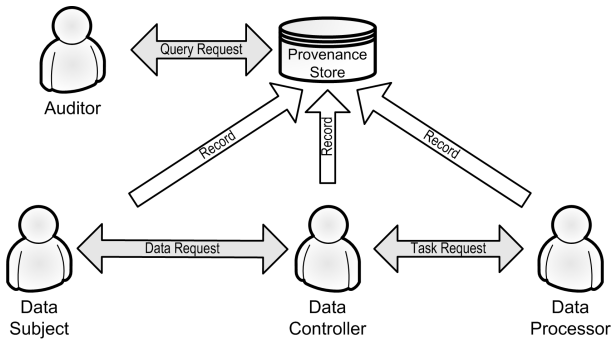

Fig. 1. Provenance-based Auditing Architecture

3.2 Data Request Protocol Formalisation

This section presents and explains the Data Request sequence diagram, which is used in the formalisation of the `Data Request` protocol.

Data Request Protocol The `Data Request` protocol represents the process in which the Data Controller establishes communication with the Data Subject to request personal information. The process is the following: `DC` requests some personal information from `DS` for a given purpose that indicates the way in which this personal information can be used, `DS` authenticates the identity of `DC` and after a successful authentication, `DS` responds with the requested information. Finally, when `DC` receives such information, `DC` acknowledges it reception. At the same time, both actors (`DS` and `DC`) record in the Provenance Store the assertions related to such a process.

Messages In the Data Request sequence diagram, which is displayed in Figure 2, the messages interchanged between `DS` and `DC` are marked with `Mi`. These messages use the notation showed in Table 1, Equation (1). To provide confidentiality, non-repudiation and integrity, these messages are symmetrically encrypted and signed. These messages also contain a unique identifier id_i and a hash-value h_i related to their corresponding assertions, which are used to create the relationship and the hash-value of the next assertions.

To make this communication protocol provenance-aware, the entities should record assertions related to the messages they send. Then, the sender of a message generates an assertion related to it indicating the relationship with the previous message. In the sequence diagram, the messages marked with `Ai` are assertions recorded by actors in the Provenance Store. These messages use the notation showed in Table 1, Equation (2) and (3), in which id_i is a unique assertion identifier of the cause and id_{i-1} of the effect of an optional relationships rel in the Provenance Store. These identifiers are created locally by the entities. d is the data contained in the message to which this assertion is related to. To

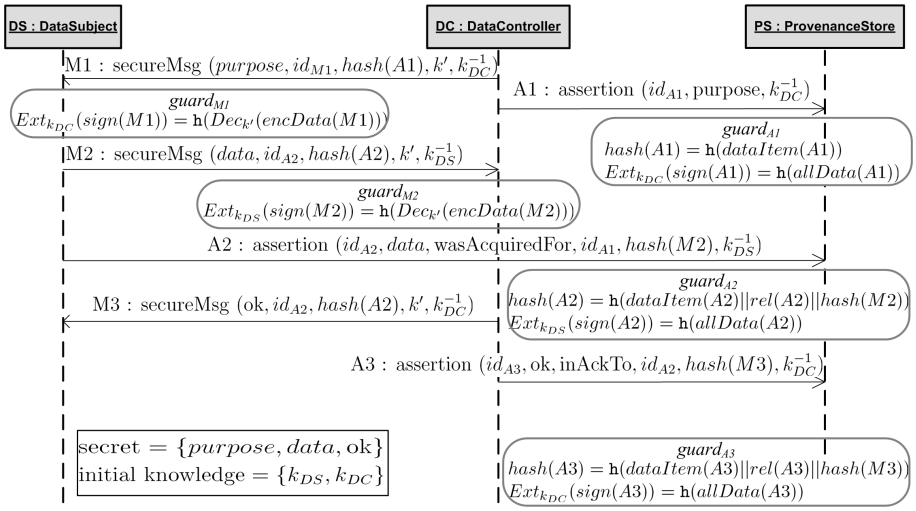

Fig. 2. Data Request UMLsec Sequence Diagram

provide integrity of the information asserted by entities, a hash-value h is introduced. This hash-value is created by the sender to protect the content of the assertion and its relationship with the previous message. For that reason, this hash-value includes the hash-value of the previous assertion, which is identified by the corresponding *id*. To provide non-repudiation, assertions also contain a signature, which is computed by the sender. If an assertion related to the first message of a protocol is created, this assertion does not contain a relationship and a hash-value then, the Equation (2) is used.

Cryptographic Elements In this protocol, a hash-value h is computed by a hash function $\mathtt{h}$, and represented as $h = \mathtt{h}(d)$. The concatenation operation is represented by $||$. For the signature, the public and private keys of an actor A are represented by k_A and k_A^{-1} respectively. The signature s is computed by $s = Sign_{k_A^{-1}}(h)$ and the verification of s by $Ext_{k_A}(s) = h$. In this digital signature scheme, a hash-value of data is signed, so after verifying the signature the hash-value should be verified too. The encryption of a piece of data d is computed by $x = \{d\}_{k'}$ and the decryption by $Dec_{k'}(x) = d$, where k' is a symmetric key. This symmetric key is created during the execution of the TLS protocol and it is used to encrypt the information that is considered private. During the TLS execution the entities also check their identities. Due to space restriction, messages related to the TLS protocol are not presented. However, TLS formalisation can be found in [9].

Guards When this protocol is executed, the guards, which are shown in the sequence diagram of Figure 2 in rounded rectangles and are identified by the

names $\texttt{guard}_{Mi}$ and $\texttt{guard}_{Ai}$, are used to verify the content of message Mi and assertion Ai, respectively. The Data Request protocol proceeds by exchanging six messages between DS, DC and PS. Message M1 contains purpose, which is symmetrically encrypted using the session key and signed using the private key of DC. With this message DC requests personal information from DS indicating the purpose from which this information is captured. When M1 is received, it is later verified and decrypted, as $\texttt{guard}_{M1}$ shows. In response, DS sends the encrypted personal data requested (data) in message M2, which is also signed by DS. When DC receives M2, the signature is verified and the data is decrypted, as $\texttt{guard}_{M2}$ shows. Then, DC sends an acknowledgement to the reception of the data to DS in M3, which is also verified and decrypted (even its corresponding guard is not presented to avoid cluttering the diagram).

Table 1. Auxiliar Functions

$secureMsg(d, id_i, h_i, k', k_A^{-1}) = \left\langle \{id_i \| d \| h_i\}_{k'}, sign_{k_A^{-1}}(\texttt{h}(d \| id_i \| h_i)) \right\rangle$ (1)
If $Mi = secureMsg(d, id_i, h_i, k', k_A^{-1})$ then
$encData(Mi) = \{id_i \| d \| h_i\}_{k'}, sign(Mi) = sign_{k_A^{-1}}(\texttt{h}(d \| id_i \| h_i)), hash(Mi) = h_i$
$assertion(id_i, d, k_A^{-1}) = \left\langle id_i, d, \texttt{h}(d), sign_{k_A^{-1}}(\texttt{h}(id_i \| d)) \right\rangle$ (2)
If $Ai = assertion(id_i, d, k_A^{-1})$ then
$cause(Ai) = id_i, dataItem(Ai) = d, hash(Ai) = \texttt{h}(d),$
$sign(Ai) = sign_{k_A^{-1}}(\texttt{h}(id_i \| d)), allData(Ai) = id_i \| d$
$assertion(id_i, d, rel, id_{i-1}, h_i, k_A^{-1}) =$ (3)
$\left\langle id_i, d, rel, id_{i-1}, \texttt{h}(d \| rel \| h_i), sign_{k_A^{-1}}(\texttt{h}(id_i \| d \| rel \| id_{i-1} \| h_i)) \right\rangle$
If $Ai = assertion(id_i, d, rel, id_{i-1}, h_i, k_A^{-1})$ then
$cause(Ai) = id_i, dataItem(Ai) = d, rel(Ai) = rel,$
$effect(Ai) = id_{i-1}, hash(Ai) = \texttt{h}(d \| rel \| h_i),$
$sign(Ai) = sign_{k_A^{-1}}(\texttt{h}(id_i \| d \| rel \| id_{i-1} \| h_i)), allData(Ai) = id_i \| d \| rel \| id_{i-1} \| h_i$

Turning to assertions, A1 creates an assertion related to the first message of the process, then, it does not create a relationship. When A1 is received by PS the hash-value and the signature contained in it are checked, as $\texttt{guard}_{A1}$ shows. A2 creates a relationship indicating that data contained in M2 *was Acquired For* the purpose contained in M1. Again, when this assertion is received by the PS, its hash-value and signature are checked according to $\texttt{guard}_{A2}$. Finally, A3 records a relationship indicating that M3 was sent in acknowledgement to (*in Ack To*) M2. Similarly, this assertion is checked according to $\texttt{guard}_{A3}$. If any of the

guards related to the assertions does not check, it means that the integrity of the asserted information was compromised. Then, the protocol terminates in a failed state and the appropriated measures should be carried out. After each guard is successfully checked, the corresponding assertion is stored in the Provenance Store.

3.3 Storage Stage

After a successful execution of the protocol, the assertions are stored in the Provenance Store; we are then able to check the integrity of its complete content by checking each of the hash-values and signatures of the assertions. That guarantees that the assertions were not modified during their exchange or during their storage. This checking can be used to frequently inspect the integrity of the stored information and take the necessary measures if a problem is found. This mechanism also prevents internal attacks, such as attacks from the Provenance Store administrator that can maliciously modify the stored assertions, as the assertions' hash-values were created by the architecture entities.

Another important issue is the maliciously insertion of assertions. This can occur in three different ways: insert a malicious message in the communication that creates a malicious assertion, an entity creates a malicious assertion to record it in the Provenance Store, or a malicious assertion is inserted directly to the Provenance Store. To prevent the first one, we rely on nounce numbers included in the interchanged messages as part of the TLS protocol [10]. This technique prevents the insertion of malicious messages, and consequently, the creation of assertions related to them. To prevent the second one, we assume that all the entities creating assertions are properly authenticated, so we can trust in the assertions created by them. In the last one, we assume that the Provenance Store is properly protected and just entities with the right credentials can record assertions.

So far, we have secured the assertions created by the entities of our architecture. However, as our architecture can contain various entities that interchange information at different times, new relationships can be created continuously. For example, suppose that an entity A produced a result r that is later reused by an entity B. When A produced r, it was not aware that r would be reused by another entity. Therefore, A did not create any relationship related to that reusing process. When B reuses r, it creates a relationship indicating the way in which r is reused by B. If we obtain the complete provenance graph of r, we will get two relationships: one created by A, indicating how r was produced, and one by B, indicating how r was reused. During the querying process, both relationships are linked by the Provenance Store to the item r. However, as such a link is created at the querying stage, the mechanism explained in Section 3 does not secure it. For that reason, we create a different mechanism to protect the integrity of provenance graphs. This mechanism is presented in the next section.

4 Securing the Querying and Analysis Stage

At this point, we can guarantee that the assertions generated by entities and stored in the Provenance Store have not been maliciously altered during the recording and storage stages. Then, they can be queried to obtain provenance graphs containing the provenance of some data. To maintain the integrity of these provenance graphs during the querying stage, the Provenance Store includes new cryptographic components in them. To achieve that, we have developed a *Secured Provenance Graph*, which defines a data structure that is included in each node of a provenance graph and is later used to verify its integrity. By including this structure, we are protecting the provenance graphs from any malicious alteration performed by an attacker, including the auditors. In the next section, the Secured Provenance Graph is presented and explained.

4.1 Secured Provenance Graph

Let us consider a set of node identifiers Id, a set of references to data D, a set of hash-values H, and a set of relationships' names R. A *Secured Provenance Graph* $G = (V, E, Node, Edge)$ is a directed acyclic graph, where $V = Id$, $E \subseteq Id \times Id$, $Node : Id \rightarrow D \times H$ and edges are labelled using the function $Edge : E \rightarrow R$.

Let us consider a secured provenance graph G. Each node contains a reference to a piece of data and a hash-value. Then, given an $id \in V$, we obtain its corresponding data by the accessor $data_G(id)$ and its corresponding hash-value by the accessor $hash_G(id)$. We also obtain the list of ancestors' identifiers by the accessor $ancestor_G(id)$, which is lexicographically ordered. The hash-value contained in each node is calculated according to Equation (4).

$$compHash_G(id) = \mathtt{h}\left(data_G(id) \Big\| \bigg\|_{id_i \in ancestor_G(id)} edge(id, id_i) \parallel hash_G(id_i)\right) \quad (4)$$

Equation (4) creates a hash-value that is used to verify not only the integrity of the data and the relationships related to id but also the integrity of the past of such data. This is achieved by including the hash-values of the id's ancestor, which creates an unforgeable reference to the id's past. The complete Provenance Secured Graph is protected by the signature of the Provenance Store, so it is not possible for another entity to reproduce or alter it without being noticed. Then, after the graph is created, we compute the signature $S = Sign_{k_{PS}^{-1}}(G)$, which is attached to the corresponding provenance graph.

Figure 3 presents an example of a Secured Provenance Graph, in which nodes are represented by circles containing references to data d_i, the directed edges are labelled with relationships r_i and the hash-values associated with each node are represented as h_i. Note that the Provenance Store does not always have access to the data that is part of a provenance graph, for that reason the nodes contain references to data. This way, we also avoid any problems related to the privacy

of this information. Here, we assume that the data itself is protected by access control techniques implemented in the corresponding data repository.

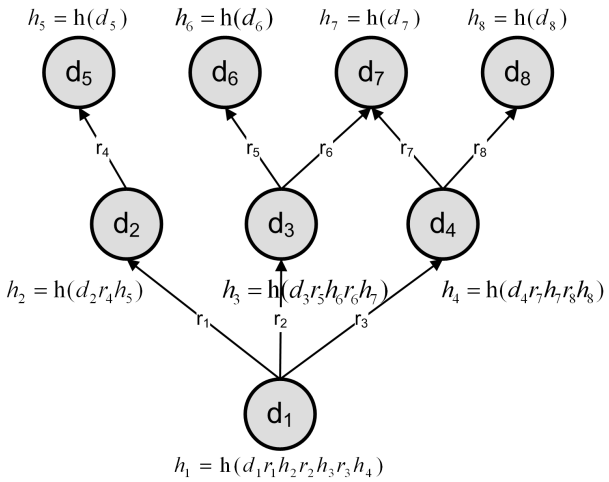

Fig. 3. Secured Graph Example

It is important to note that the order of the relationships and the hash-values in each node is a very important issue. In graphs, the outgoing edges of a node are not ordered. However, if we want to create and later verify the hash-values contained in such a node, it is necessary to preserve certain order in the checking process. For example, the hash-value of node d_3, which is presented in Figure 3, can be created in two different ways. If we take r_5 in first place, we obtain the hash-value $h_3 = \mathtt{h}(\mathrm{d}_3 r_5 h_6 r_6 h_7)$. But, if we take r_6 in first place, its hash-value is $h_3 = \mathtt{h}(\mathrm{d}_3 r_6 h_7 r_5 h_6)$. Both hash-values represent the same node in the provenance graph. Nevertheless, if we do not know the order in which the hash-value was created, its checking will be incorrect as $\mathtt{h}(\mathrm{d}_3 r_5 h_6 r_6 h_7)$ is different from $\mathtt{h}(\mathrm{d}_3 r_6 h_7 r_5 h_6)$. In our case, the list of ancestors' identifiers is lexicographically ordered according to the relationship's names. Then, the correct hash-value is $h_3 = \mathtt{h}(\mathrm{d}_3 r_5 h_6 r_6 h_7)$. Note that a provenance graph can contain nodes with no relationships. This does not mean that such nodes do not have a "past". Instead, however, it means that the provenance graph does not contain the past of such nodes because it is not relevant for the analysis stage. If for some reason, a problem is found in these nodes without explicit past, the auditor can request to the Provenance Store a provenance graph showing its past. Later, this new provenance graph can be checked.

4.2 Secured Provenance Graph Integrity Checking

After a provenance graph is received by an auditor, its integrity needs to be checked. In that way, we can detect any malicious alteration made to it by any attacker or by the auditor. Hence, in this section, we present the algorithm used to verify the integrity of a Provenance Secured Graph.

Algorithm 1 Secured Provenance Graph Integrity Checking

```
1: procedure integrityCheck(G :Secured Provenance Graph, k_PS : publicKey)
2:     id : node identifier ∈ V
3:     if Ext_{k_PS}(Sign_{k_PS^{-1}}(G)) ≠ h(G) then
4:         return 0                                   ▷ signature does not check
5:     end if
6:     for all id ∈ G do
7:         if hash_G(id) ≠ (compHash_G(id)) then
8:             return -1                              ▷ integrity is compromised in id
9:         end if
10:    end for
11:    return 1                                       ▷ Success
12: end procedure
```

In order to check the integrity of a Secured Provenance Graph, the procedure integrityCheck is introduced in Algorithm 1. Initially, the signature associated with such a graph is verified using the public key of the Provenance Store, k_{PS}. This signature is used to check that the content of the complete graph was not altered. If the signature cannot be verified, there is no reason to continue with the rest of the process, then the algorithm returns 0.

If the signature checks, then the algorithm verifies the hash-value of each node in the graph. This is achieved by visiting each graph's node to create a new hash-value by calling the $compHash_G$ function, which is presented in Equation (4). This new hash-value depends on the ancestors' hash, which in turn depends on the hash-values of its ancestors. Later, the new hash-value is compared against the hash-value contained in the node. If they are not the same, the integrity of this node has been compromised, and the algorithm returns -1. If after visiting all the nodes no problem is found, the nodes' integrity is intact, and the algorithm returns 1.

If the integrity of any of the provenance graph nodes has been compromised, it will be indicated which one was altered using the corresponding *id*. Then, the auditor can access the information related to such *id* stored in the Provenance Store to check if it was altered since the recording stage. If none of them were altered, then an audit process is allowed to begin. In this way, we can guarantee that the results derived from the analysis of a secured provenance graph are based on information that has not been maliciously altered.

In our scheme deletion of provenance information is not allowed as, to be able to perform a successful audit, we need all the assertions that our model

records. Moreover, if one or more assertions are deleted, the presented algorithm finds an integrity problem, as the hash-values will not check, and not a problem of deletion of assertions. To avoid that, provenance repositories should implement appropriate access control techniques. In certain scenarios, deletion is used to enforce privacy of the information avoiding the identification of a specific individual through personal data. To support that, instead of deletion we use anonymisation. This is a technique that uses references to data in provenance information instead of real data [13]. These references, which are references to data stored in a database, are solved by accessing such a database. In that way, we only have access to the anonymised data if we have the right permissions and credentials.

5 Securing Provenance Based-Audits

In this section, we explain how we check the model of the Provenance-based Auditing Architecture presented in Section 3.1. The architecture model consist of three sequence diagrams (Data Request, Task Request and Query Request, which represents the processes presented in Section 4), which need to be verified separately. For space restrictions, we only present the verification of the integrity property in the Data Request sequence diagram presented in Section 3.

To verify that the integrity property is held by the data exchanged in our sequence diagram, we use UMLsec to create attacks against the modelled protocol using an adversary model. The adversary model we use here represents a network attacker that can eavesdrop, modify or insert messages on the communication channel with malicious intentions, and shows that these attacks fail. This adversary model relies on an extended Dolev-Yao adversary model [14], in which an adversary can read messages sent over the communication channel to include them in its knowledge set to later use them to derive new knowledge. If the adversary breaks the integrity of the sent messages, then it can modify the messages without being detected.

The adversary object contains three types of predefined values: `secret`, `initial knowledge` and `guard`. The values associated with `secret` describe the types of data that should be protected from the attacker, in this case they should hold the integrity property to ensure the integrity of the data. The values associated with `initial knowledge` denote the information known by the attacker beforehand, whereas, $\texttt{guard}_n$ represents the operations to be performed by the receiver of message n before such a message is received.

Returning to the diagram presented in Section 3, the items `purpose`, `data` and `OK` are part of the secret type indicating that they need to be protected during the execution of the process. The initial knowledge set contains the public keys of the actors in the diagram (k_{DC} and k_{DS}).

In the Data Request sequence diagram, we model the messages exchanged between entities and the assertions recorded by them in the Provenance Store. The integrity of the messages is verified by using a digital signatures scheme (hash) whereas the integrity of assertions is checked by using the included hash-

value and its corresponding checking, which is represented by the guards in the diagram.

To verify that the integrity property is maintained during the execution of the protocol modelled in the Data Request sequence diagram, we use the model checker Viki [15]. Viki is a software that receives as input a UML sequence diagram and its adversary model to return the possible attacks that can be performed by the given attacker in the modelled protocol. Viki obtains the security requirements from the UMLsec elements and the predefined values used in sequence diagrams [15]. Then, these requirements are formalised in First-Order Logic and analysed with automatic theorem provers (e-SETHEO [16] and SPASS [17]) to find a flaw. If a flaw is found, a Prolog engine can be used to generate the attack trace of such flaw and solve it. Each modelled sequence diagram and its corresponding adversary model are executed at the same time to verify if the defined security properties are held during the whole execution [18]. As in this context, an attack means that a property is not held [18], we define an integrity attack as follows.

Definition 1 (Successful Integrity Attack). *A sequence of protocol transitions that lead to a piece of data to be modified without being noticed.*

Then, the integrity property that Viki checks is the following.

Lemma 1 (Integrity Property). *For the Data Request protocol, no successful integrity attack is possible.*

The verification of this lemma relies on the collision resistant nature of the used cryptographic hash function guaranteeing that an adversary cannot alter the integrity of a piece of data (messages or assertions) without having a visible effect in the output. Neither an adversary can insert an entirely new piece of data without being detected. Then, under the assumption that we use a collision resistant hash function, we can guarantee the integrity property in the modelled protocol. After performing the verification using Viki, the outcome is that the modelled protocol holds the Integrity Property. Therefore, we can guarantee that the assertions generated by this protocol hold the Integrity Property and can be used for creating query results.

For each protocol of the Provenance Based-Auditing Architecture four lemmas have been derived, which cover the confidentiality, integrity, non-repudiation and authentication properties. Considering that we model three protocols in our architecture, then we derived a total of 12 lemmas. Due to the lack of space, these lemmas are not shown. However, we present the following theorem, which covers the complete architecture.

Theorem 1 (Secure Provenance Based-Auditing Architecture). *A Provenance Based-Auditing Architecture is secure if for the protocols Data Request, Task Request and Query Request, the Integrity, Confidentiality, Non-Repudiation and Authentication properties are held.*

The verification of this theorem relies on the proofs of each of the property lemmas derived from each of the protocols of the architecture. If each of the

properties is held by all of the protocols in our architecture, then the theorem holds.

Since Theorem 1 holds for our architecture, we can conclude that the architecture is secure and, therefore, the audits performed on it are secure too. Then, the results derived from these secure audits are based on correct information. This theorem was verified by using Viki, which concluded that the modelled architecture is secure.

6 Related Work

Recently, researchers have realised that provenance should be preserved in its original form while is created, transported, recorded and queried. This way, we are able to trust in all result derived from its analysis. For that reason, some researchers [19–24] have focused on presenting and solving the problem of securing provenance information.

Tan *et al.* [19] expose and discuss the problem of security provenance in a SOA-based Provenance System. Here, to ensure accountability, liability and integrity of assertions, they make use of digital signatures providing non-repudiation and ensuring that assertions are not changed intentionally or accidentally. Contrary to our work, they discuss basic security issues within provenance system and mention some solutions but they do not explain how these solutions can be implemented in practice. Moreover, this work mostly relies on access control techniques implemented in the provenance repository.

Hasan *et al.* [20] present the problem of securing provenance as an issue that had not been explored but is essential when provenance is used in law, digital forensic, regulatory compliance and authorships context. They identify integrity, availability and confidentiality as the main properties that a provenance-aware system should handle to provide trustworthy provenance. They also base their analysis in a different provenance model in which provenance is represented by linear chains. Even the secure provenance problem is introduced and discussed, a practical approach to implement it is not presented. In another work by Hasan *et al.*, they present a secure provenance scheme for linear chain provenance representation [24]. Such model support confidentiality and integrity of provenance information. Their scheme is similar to our approach in the sense that they also include extra cryptographic information to the provenance information to ensure the mentioned properties. The main difference between their work and ours is that we protect the complete information flow of a provenance-aware system. We also are able to protect a non-linear provenance representation (i.e. provenance graphs as in OPM).

Braun *et al.* also discuss the securing provenance problem [21]. In this case, they use a similar provenance model to the one we use, in which provenance is represented as a causality graph. For that reason, provenance information differs from traditional data and, therefore, the existing security models used to protect "traditional information" do not apply to graphs and are not easy to extend. Thus, new solutions should be developed and specially designed. Their

work focuses on access control and how each of the elements of a causality graph needs different levels of access control. In this paper, we have developed a new technique specially designed to protect the integrity of a provenance graph but focusing on the integrity property and not on access control. However, our work is compatible with access control techniques.

Chong *et al.* discuss the problem of confidentiality and privacy of provenance information from a semantic point of view in a "provenance traces" approach [23]. They develop semantic definitions and mechanism to enforce these security properties. They also mention that data and provenance have different security requirements and, therefore, special mechanism to protect provenance information should be designed. Although, in this paper we focus on the integrity of provenance information, we have also modelled and verified confidentiality and privacy of provenance information by using cryptographic and anonymisation techniques, respectively. This work is not presented due to space restrictions but we expect to publish it later.

Xu *et al.* present the secure provenance problem from the management point of view. Here, they present some desirable requirements for secure provenance management systems and propose a framework that satisfies these requirements [22]. Integrity is among those requirements for which they adopt a similar approach to ours: integrity of both data and provenance information is important. To ensure that, they propose the creation of a layer in their framework that maintains the integrity of data and provenance information during storage, transferring and processing. However, unlike us, they do not present any practical solution to support that property.

Finally, a set of approaches [25–28] used by the database community to support similar security properties as the ones presented in this paper can complement our work. Even these approaches were not created to specifically protect provenance information, the solutions presented to solve security issues (such as privacy) can be adapted to be implemented in the presented provenance model.

7 Conclusions

Securing provenance is critical for making systems accountable on the Open Provenance Vision, as described by Moreau [29]. This paper presents a solution for this. Initially, we have presented a framework that guarantees a set of security properties in a Provenance-based Auditing Architecture to increase the level of trust in provenance-based audit results. Due to space restrictions, we focus on the integrity property and only on one protocol of the architecture, Data Request. In this protocol, we can guarantee the integrity of the assertions created by the participant entities as well as the integrity of provenance query results. We secure them by including cryptographic elements to both that can later be verified. First, we define a secure communication protocol that ensures the integrity of the information exchanged between entities and of the assertions sent to the Provenance Store. This way, we secure the creation and storage of assertions. Second, to ensure that provenance query results have not been ma-

liciously altered, we design the Secured Provenance Graph, which contains a specially designed hash-value in its nodes along with the Provenance Store signature. Later, with the Integrity Checking Algorithm, we can verify the integrity of this graph.

Finally, we present an automatic verification of the integrity property in the communication protocol presented in Section 3. This verification allows us to guarantee the integrity of the information exchanged and the created assertions. Although, just one property and one protocol have been presented, the complete architecture and more security characteristics have been verified. Our future work is focused on extending our Secured Provenance Graph to the Open Provenance Model [8]. We are also working on measuring the overhead generated by hash-values during the recording of process documentation and the querying process.

Acknowledgements This research was partially supported by the Programme Alβan, the European Union Programme of High Level Scholarships for Latin America, (scholarship number E06D103956MX) and by the Mexican Council CONACyT (scholarship number 182546). Thanks to the anonymous reviewers for their useful comments.

References

1. Weitzner, D.J., Abelson, H., Berners-Lee, T., Feigenbaum, J., Hendler, J., Sussman, G.J.: Information accountability. Communications of the ACM **51**(6) (2008) 82–87
2. Groth, P., Moreau, L.: Recording process documentation for provenance. IEEE Transactions on Parallel and Distributed Systems (September 2009)
3. Miles, S., Groth, P., Munroe, S., Moreau, L.: PrIMe: A Methodology for Developing Provenance-Aware Applications. ACM Transactions on Software Engineering and Methodology (June) (2009)
4. Groth, P., Miles, S., Moreau, L.: A Model of Process Documentation to Determine Provenance in Mash-ups. Transactions on Internet Technology (TOIT) **9** (2009) 1–31
5. Miles, S.: Electronically querying for the provenance of entities. In: Proceedings of the International Provenance and Annotation Workshop IPAW, Springer (November 2006) 184–192
6. Aldeco-Pérez, R., Moreau, L.: Provenance-based Auditing of Private Data Use. In: International Academic Research Conference, Visions of Computer Science (BSC 08), London, UK, BCS (2008) 141–152
7. Moreau, L., Groth, P., Miles, S., Vázquez, J., Ibbotson, J., Jiang, S., Munroe, S., Rana, O., Schreiber, A., Tan, V., Varga, L.: The provenance of electronic data. Communications of the ACM **51**(4) (April 2008) 52–58
8. Moreau, L., Clifford, B., Freire, J., Gil, Y., Futrelle, J., Kwasnikowska, N., Miles, S., Missier, P., Myers, J., Simmhan, Y., Stephan, E., Van Den Bussche, J., Pale, B.: The Open Provenance Model Core Specification (v1.1). Future Generation Computer Systems (2010) 1–30
9. Jürjens, J.: Secure Systems Development with UML. Springer (2005)
10. Blake-Wilson, S., Nystrom, M., Hopwood, D., Mikkelsen, J., Wright, T.: Transport Layer Security (TLS) Extensions. RFC 3546 (June 2003)

11. Institute, N.S.: Trusted computer system evaluation criteria (5200.28-std). Technical report, Department of Defence Standard (1985)
12. HomeOffice: Data Protection Act (1998)
13. Kifor, T., Varga, L., Vazquez-Salceda, J., Alvarez, S., Willmott, S., Miles, S., Moreau, L.: Provenance in Agent-Mediated Healthcare Systems. IEEE Intelligent Systems **21**(6) (November 2006) 38–46
14. Dolev, D., Yao, A.: On the security of public key protocols. Information Theory, IEEE Transactions on **29**(2) (March 1983) 198–208
15. Jürjens, J.: Using interface specifications for verifying crypto-protocol implementations. In: Foundations of Interface Technologies, FIT'08 @ ETAPS (2008)
16. Stenz, G., Wolf, A.: e-SETHEO: An Automated Theorem Prover. In: TABLEAUX '00: Proceedings of the International Conference on Automated Reasoning with Analytic Tableaux and Related Methods, London, UK, Springer-Verlag (2000) 436–440
17. Weidenbach, C., Brahm, U., Hillenbrand, T., Keen, E., Theobald, C., Topic, D.: S PASS Version 2.0. In: CADE-18: Proceedings of the 18th International Conference on Automated Deduction, London, UK, Springer-Verlag (2002) 275–279
18. Shabalin, P.: Model Checking UMLsec Models. Master's thesis, Department of Informatics, TU München, Germany (2004)
19. Tan, V., Groth, P., Miles, S., Jiang, S., Munroe, S., Tsasakou, S., Moreau, L.: Security Issues in a SOA-based Provenance System. In: Proceedings of the International Provenance and Annotation Workshop (IPAW'06), Chicago, Illinois, Springer-Verlag (2006) 203–211
20. Hasan, R., Sion, R., Winslett, M.: Introducing Secure Provenance: Problems and Challenges. Proceedings of the ACM Workshop on Storage Security and Survivability (StorageSS), ACM Press (2007) 13–18
21. Braun, U., Shinnar, A., Seltzer, M.: Securing Provenance. Proceedings of the 3rd USENIX Workshop on Hot Topics in Security (HotSec 08), USENIX Association (2008)
22. Xu, S., Ni, Q., Bertino, E., Sandhu, R.: A Characterization of the problem of secure provenance management. In: International Conference on Intelligence and Security Informatics, Texas, USA, IEEE (June 2009) 310–314
23. Chong, S.: Towards semantics for provenance security. First workshop on Theory and practice of provenance (2009)
24. Hasan, R., Sion, R., Winslett, M.: The Case of the Fake Picasso: Preventing History Forgery with Secure Provenance. In: FAST '09 7th conference on File and storage technologies (2009), Berkeley, CA, USA, USENIX Association (2009) 1–14
25. Miklau, G., Levine, B.N., Stahlberg, P.: Securing history: Privacy and accountability in database systems. In: Conference on Innovative Data Systems Research (CIDR '07), Asilomar, CA, USA (2007) 387–396
26. Stahlberg, P., Miklau, G., Levine, B.N.: Threats to privacy in the forensic analysis of database systems. In: International Conference on Management of Data (SIGMOD'07), New York, NY, USA, ACM (2007) 91–102
27. Vaughan, J.A., Jia, L., Mazurak, K., Zdancewic, S.: Evidence-based audit. In: 21th IEEE Computer Security Foundations Symposium (CSF '08), Washington, DC, USA, IEEE Computer Society (2008) 177–191
28. Lu, W., Miklau, G.: Auditing a Database under Retention Restrictions. In: IEEE International Conference on Data Engineering (ICDE '09), Washington, USA, IEEE Computer Society (2009) 42–53
29. Moreau, L.: The foundations for provenance on the web. Foundations and Trends in Web Science (In Press) (November 2010)