Preserving Integrity and Confidentiality of a Directed Acyclic Graph Model of Provenance

Amril Syalim, Takashi Nishide, and Kouichi Sakurai

Department of Informatics, Kyushu University, Fukuoka, Japan
amr@itslab.csce.kyushu-u.ac.jp,{nishide,sakurai}@inf.kyushu-u.ac.jp

Abstract. This paper describes how to preserve integrity and confidentiality of a directed acyclic graph (DAG) model of provenance database. We show a method to preserve integrity by using digital signature where both of the provenance owner and the process executors (i.e. contributors) sign the nodes and the relationships between nodes in the provenance graph so that attacks to integrity can be detected by checking the signatures. To preserve confidentiality of the nodes and edges in the provenance graph we propose an access control model based on paths on the provenance graph because an auditor who need to audit a result normally need to access all nodes that have causal relationship with the result (i.e. all nodes that have a path to the result). We also complement the path-based access control with a compartment-based access control where each node is classified into compartments and the auditor is not allowed to access the nodes included in a compartment that can not be accessed by him/her (because of the sensitivity of the compartment). We implement the path-based access control by encrypting the nodes and later store encrypted encryption's keys in the children of the nodes. The compartment-based access control is implemented by encrypting the nodes in different compartments with different keys. We developed a prototype of the model and performed experiments to measure the overhead of digital signature and the double encryptions.

1 Introduction

In a system where we need to understand the processes that have been executed to produce a result we need to record provenance of the execution [1, 2]. By recording provenance we may trace who have contributed to the creation of the result [3, 4]. This feature is very important whenever we need to verify the process of result's creation, for example in a distributed system (i.e. a grid system), where a result may be produced by many parties in different computers [5]. Another real life example is in a hospital, a medicine prescription may be created by a doctor based on the examination result of another doctor. The result of the another doctor may be based on the examination results of the other doctors working in the same or other hospitals. Whenever there is something wrong with the prescription we need to trace who produce incorrect examinations. By using provenance of examination we can trace the process to create the prescription.

To be useful, the process executors (i.e doctors) should not have ability to alter, delete or add the provenance of their examination results with intention to make errors go to other doctors. The doctors also can not deny their examination results.

For a sequential execution of processes, provenance can be represented in a form of chain [3, 4]. A more expressive model that is suitable for a parallel execution is a directed graph model where nodes in the graph represent processes and the edges represent relationships between the processes (nodes) [6, 7]. Because provenance is tightly associated with time, many models of provenance take the form of a directed acyclic graph (DAG) [8].

In this paper, we are focusing on securing a directed acyclic graph model of provenance in terms of integrity and confidentiality. To ensure integrity of the provenance graph (i.e. nodes and edges) we need to assure immutability and non-repudiation properties of each node and edge in the provenance graph. The contributors (i.e. the people or processes that contribute in the provenance graph) can not cheat for any purposes. The other parties in the provenance system (i.e. the manager of the provenance graph that we refer in this paper as the provenance owner), although powerful enough to manage access to provenance, also can not cheat (i.e by changing the provenance graph) without being detected. We propose a method to protect integrity of provenance by employing digital signature. Using this method, the contributors and provenance owner both sign the provenance's nodes and edges. To alter the nodes and edges without detected needs collusion from the two parties which means to repeat the process execution from beginning.

To support confidentiality of the provenance graph we need to define access control model to provenance and how to enforce the access control model. Provenance should only be accessed by the person who has the right to access, for example an auditor who need to audit the process [3, 4]. The system should support restricting access to only some parts of the provenance [1]. Many access control models employ grouping mechanism to improve efficiency and security (i.e. by using groups, roles, security levels/compartments). We propose a grouping mechanism for access control to provenance by utilizing two grouping methods: grouping of entities in the provenance graph based on paths and grouping entities based on compartments. Grouping by paths is useful because the auditors who audit the process should be interested in the causal relationship in the provenance graph. However access control by paths alone is not expressive to enforce a more specific policies (i.e. an auditor only can access a part of nodes/edges in the paths). We complement the paths-based access control with a compartment-based access control so that we can enforce such policies. By using a compartment-based access control, each node is assigned with a compartment and the provenance owner grants access to the nodes in a compartment by granting access to that compartment.

2 Related Works

Hasan et al. [9, 4, 3] show a threat model to a chain model of provenance and the method to prevent/detect the attacks associated with the threats by using digital signature and broadcast/threshold encryption. Our method extends their method by applying the digital signature method to a directed acyclic graph model of provenance. While Hasan et. al need to sign the provenance record in the provenance chain and including checksum of previous provenance record in the chain to maintain the integrity of record and the chain structure, we need to sign the nodes in the provenance graph and including the signed checksum of the parent nodes. Hasan et. al use a broadcast and threshold encryption to support confidentiality so that they do not propose a specific access control model, while we propose a specific access control model based on paths and compartments.

Braun et.al. argue that provenance needs new security model [8]. They also propose a security model for provenance based on observation of the usage of provenance [10]. They focus on security model but do not deeply discuss how to guarantee integrity of provenance. Their main proposal is that we need to control access to edges (head and tail) and nodes (attributes). Their access control is more expressive because we can define access to the level of a head and tail of an edge. However, there is no analysis of efficiency of the model and the mechanism to implement the access control model.

3 Integrity Mechanism: Digitally Signing the Provenance Graph and Its Security Analysis

An example of provenance graph with six contributors is shown in the Figure 1. The Figure 1 shows that to produce the final result, the contributor C5 uses the outputs of contributors C1 and C2 while contributor C6 uses the output of contributors C3 and C4. Contributor C7 uses the output of C5 and C6 which later used by C8 and C9. The final process is executed by C10 that processes the outputs of C8 and C9. After each process is executed and the provenance of the process (i.e. node) is created/generated, the provenance is stored in the provenance database. The other papers [1, 5, 11–13] call the provenance database as a provenance store.

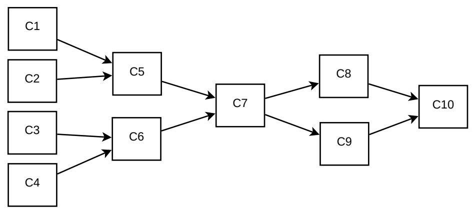

Fig. 1. Provenance graph

We identify three groups of active entities involved in a provenance system: provenance owners, contributors, and auditors. A provenance owner is the owner of provenance that mediates the provenance recording process and manages access to the provenance. The contributors are the people who execute process and contribute the results. Auditors are the people who need to access the provenance graph, for example for reviewing or auditing the process's execution.

Provenance is recorded after each process is executed by the contributor. In a distributed system, before executing the distributed process, a worflow (i.e. a distributed execution plan) should be defined and sent to the provenance owner. The process to create a workflow may involve some or all of contributors. Based on the workflow, provenance owner sends each contributor information that is needed by the contributor to execute each process in the workflow (i.e inputs of the process). After a contributor execute a process, the contributor should produce outputs which we refer as a document. The provenance of the document is documentation of process execution to produce the document. The provenance can be automatically generated by the system where the contributor execute the process or manually created by the contributor. After execution of a process, the document and provenance of the document are sent to the provenance owner which later record them as a node in the provenance database. The provenance owner may also send the document to contributors that need the documents for their inputs.

After the provenance is recorded, there are some possible integrity problems with provenance. We identify four main problems: repudiation, alteration, deletion, and addition. A contributor may deny that she/he has contributed the document and its provenance. The document and provenance (i.e. nodes) may be altered by an attacker so that they do not reflect original process. Attacker may also delete a node or add a fake node.

The basic idea of the digital signature mechanism is whenever a provenance of a document is recorded, both of the contributor and provenance owner sign the document and the provenance before storing the provenance to the database. Whenever a contributor uses an output document of other contributor as an input, the contributor should create the hash/checksum of the input and store them as a provenance of the process executed by the contributor.

We assume that each contributor, auditor and provenance owner has a pair of public key and private key and each party can retrieve the public keys of the other parties securely. The private keys can only be accessed by the owner of the key. Let D_n is the document created by a contributor identified by n and P_n is a provenance of the document. The function $H(D_n)$ is a function that produce hash value of D_n. The function S_n is a signing function where $S_n(D_n)$ is a function that produce digital signature of contributor n to document D_n.

If a contributor n needs to use a document (i.e. D_{n-1}) produced by another contributor (i.e. contributor $n-1$) as input, before the contributor n executes the process, the provenance owner sends the input that has been signed by the provenance owner and the another contributor: $S_o(S_{n-1}(D_{n-1}))$. After verifying the document and the signatures, the contributor n execute the process. The contributor n signs the result D_n, its provenance P_n and hash of the input $H(D_{n-1})$. The signed result, its provenance and hash of the input is $S_n(P_n, D_n, H(D_{n-1}))$. The contributor sends them to the provenance owner. The provenance owner signs them and store them in the database.

This scheme supports integrity by preventing contributor deny a node after committing the node and detecting other attacks: alteration, deletion, and addition. A contributor cannot deny a node after committing the node because if the contributor deny a node means that the signature has been forged which is very unlikely. As for alteration, the possibilities are as follow:

1. A contributor alters the content of the nodes (i.e. documents or provenances) and create a new signature. The attack is not possible because the contributor cannot create a new signature of the provenance owner.
2. A provenance owner alters the content of the nodes and create the new signature. The attack is not possible because the provenance owner can not create a new signature for the contributor.
3. The other people accessing the system alter the nodes. The attack is not possible because they can not create the signatures of contributors and provenance owners.
4. The provenance owner and a contributor collude to alter a node, they still need to collude with all successors of the node because the children of the node include the hash of the parent's documents in their provenance. Colluding with all children mean repeating the process from the beginning.

As for addition, the possibilities are as follow:

1. If the provenance owner adds a new node, the provenance owner cannot create the contributor's signature of the new node.
2. If a contributor inserts a node between a parent and its children and change the references in the nodes so that the new node become the children of the parent and the previous children become the children of the new node, she cannot create the signature of the previous children so that the previous children are still refer to the original parent as their parent. The contributor also can not forge the signature of the provenance owner.

For deletion, the possibilities are as follow:

1. If a contributor or provenance owner deletes a node and want to change the relationship so that the children of the node become the children of any other nodes. They cannot change the signature consistently without colluding with all contributors of successors of the node.
2. The other people deletes the nodes. The attack is not possible because they can not create the signatures of contributors and provenance owners.

4 Confidentiality Mechanism: Path-based Access Control and Encrypting the Provenance Graph

To protect confidentiality of provenance we need to prevent confidential provenance information be accessed by unauthorized people accessing the system.

However, the system should also support authorized access to provenance (i.e. authorized auditors who need to access provenance to do audit and verify the process of object creation). We propose an access control model based on path on the provenance graph. The arguments of our proposal is that an auditor normally needs to access all nodes that have a path to the result because the nodes have causal relationship to the result. We believe that this model is more efficient and comfortable because the provenance owner can easily create access based on paths in the provenance graph.

However, by using path-based only access control, we can not create a more expressive policy (for example an auditor can only access a part of the paths). We combine path-based policy with another access policy based on compartments. Compartments define separation between nodes in different security level/classes and the auditors that can access those compartments.

We propose to implement the access control model by using cryptographic mechanisms (i.e. encryption). This method is especially important if we store the provenance in an untrusted server (i.e. the provenance owner wants to outsource the storage of provenance to a third party who may be not trusted). This method can also be used if the provenance owner wants to implement cryptographic-based access control (where the data is encrypted and access rights are granted by giving the encryption keys). The idea of our implementation for path-based access control is to encrypt the nodes and store the encryption keys in the children of the nodes. Below of the detail of the encryption process (see Figure 2).

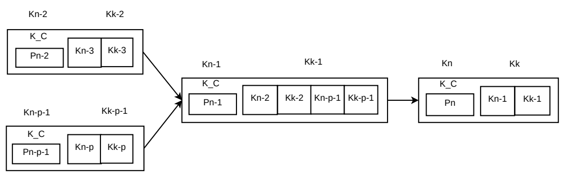

Fig. 2. Encrypting the Provenance Graph

Let P_n is the node that has been signed by the contributor n and the provenance owner o and let $E_k(P_n)$ is an encryption function that encrypt P_n with private key k. To encrypt the node P_n, the provenance owner define compartment of the node and find the parent nodes. The provenance owner retrieves the key associated with the compartment K_C, the keys to encrypt the parent nodes K_{n-1} and the key to encrypt the grandparent node K_{k-1}. The provenance owner generates two random keys: node's key K_n and parent-key's key K_k and store the keys in a key database managed by the provenance owner. The provenance owner encrypts the node P_n with key K_C. Then the provenance owner re-encrypts the node with the key K_n. After that the provenance owner encrypts the keys K_{n-1} and K_{k-1} with parent-key K_k. Encrypted form of the node is

$E_{K_n}(E_{K_C}(P_n))|E_{K_k}(K_{n-1}|K_{k-1})$. The provenance owner stores encrypted form of the node in the provenance database.

The provenance owner may combine both path-based access and compartment access policy or only use a path-based access policy. To create a policy, the provenance owner first define the compartment of each provenance and encrypt the provenance with key K_C for that compartment. The provenance owner assigns which compartment that can be accessed by an auditor by giving the K_C to the auditor.

The provenance owner can define the policy of access of the auditor to the nodes based on the document that should be audited by the auditor. There are two keys in each node: K_n for encrypting provenance in the node and K_k is the key for encrypting the parent nodes (i.e. the nodes that have paths to the current node). By providing/not providing the keys K_n and K_k, there are four possible access policies of an auditor to a node:

1. The auditor cannot access any component of the node (provenance and the parent nodes). In this case the auditor is not provided any keys.
2. The auditor can access the node but not the parent node. In this case the auditor should be provided the key K_n.
3. The auditor cannot access the the node but can access the parent nodes. In this case the auditor should be provided with the key K_k.
4. The auditor can access the node and also the parent nodes. In this case the auditor should be provided both of the key K_n and K_k.

5 Experimental Results

We implemented the digital signature and encryption scheme and did two experiments to measure the overhead of the digital signature and encryption mechanisms. In the first experiment, we stored the provenance without first signing and encrypting the provenance. In the second experiment, we stored the identical provenance and used the digital signature and encryption scheme to sign and encrypt the provenance.

We performed experiments for workflows that produce provenance with the number of nodes 8, 16, 32, 64 and 128. The size of documents and provenance of the documents were between 100KB to 150KB. We measured the time to sign, encrypt, and store documents and provenance (the time to execute the process to produce the documents was not measured). The program was implemented with Java version 1.6 and used DSA for digital signature and AES for encryption. In the experiments, we executed the program on a Linux machine (Linux version 2.6.31) with Intel Core 2 Duo 2.00GHz processor and main memory 2GB. The documents and provenance were stored in a Postgresql database (version 8.4) run on another machine (a Linux version 2.6.31 with Pentium Dual-Core 2.50GHz processor and main memory 2GB) connected by a Wireless Local Area Network (speed 54Mbps). For each experiment and the number of nodes we executed and measured the execution times three times.

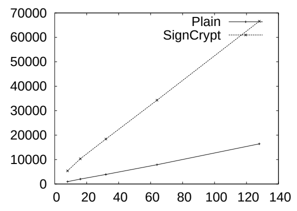

Fig. 3. Execution time (ms)

The Figure 3 shows the experimental results. The X axis is the number of nodes, the Y axis is the times to store the provenance (in milliseconds). From the table we can find that the overhead to sign and encrypt provenance is about 5 times in compare to store provenance without signing and encrypting. This overhead shows that to sign and encrypt provenance graph takes time much higher than the process to store plain provenance graph to the database.

References

1. Groth, P., Jiang, S., Miles, S., Munroe, S., Tan, V., Tsasakou, S., Moreau, L.: An architecture for provenance systems. Technical report, University of Southampton (November 2006)
2. Buneman, P., Khanna, S., Tan, W.C.: Why and where: A characterization of data provenance. In: ICDT. (2001) 316–330
3. Hasan, R., Sion, R., Winslett, M.: Preventing history forgery with secure provenance. ACM Transactions on Storage **5**(4) (December 2009) 12:1–12:43
4. Hasan, R., Sion, R., Winslett, M.: The case of the fake picasso: Preventing history forgery with secure provenance. In: FAST. (2009) 1–14
5. Groth, P., Luck, M., Moreau, L.: A protocol for recording provenance in service-oriented grids. In: Proceedings of the 8th International Conference on Principles of Distributed Systems (OPODIS04. (2004) 124–139
6. Moreau, L., Freire, J., Futrelle, J., McGrath, R.E., Myers, J., Paulson, P.: The open provenance model: An overview. In: Second International Provenance and Annotation Workshop, IPAW 2008. (June 2008)
7. Shawn Bowers, T.M.M.B.L.S.C.S.B.D.: A model for user-oriented data provenance in pipelined scientific workflows. In: IPAW 2006. (2006) 133–147
8. Braun, U., Shinnar, A., Seltzer, M.I.: Securing provenance. In: HotSec. (2008)
9. Hasan, R., Sion, R., Winslett, M.: Introducing secure provenance: problems and challenges. In: StorageSS. (2007) 13–18
10. Braun, U., Shinnar, A.: A security model for provenance. Technical report, Harvard University (2006)
11. Paul Groth, S.M., Moreau, L.: Preserv: Provenance recording for services. In: UK e-Science All Hands Meeting 2005. (September 2005)
12. Altintas, I., Barney, O., Jaeger-Frank, E.: Provenance collection support in the kepler scientific workflow system. In: IPAW. (2006) 118–132
13. Chapman, A., Jagadish, H.V., Ramanan, P.: Efficient provenance storage. In: SIGMOD Conference. (2008) 993–1006