Securing Data Provenance in the Cloud

Muhammad Rizwan Asghar[1,2], Mihaela Ion[1,2],
Giovanni Russello[1], and Bruno Crispo[2]

[1] Create-Net, Italy
{asghar,ion,russello}@create-net.org
[2] University of Trento, Italy
crispo@disi.unitn.it

Abstract. Cloud storage offers the flexibility of accessing data from anywhere at any time while providing economical benefits and scalability. However, cloud stores lack the ability to manage data provenance. Data provenance describes how a particular piece of data has been produced. It is vital for a post-incident investigation, widely used in healthcare, scientific collaboration, forensic analysis and legal proceedings. Data provenance needs to be secured since it may reveal private information about the sensitive data while the cloud service provider does not guarantee confidentiality of the data stored in dispersed geographical locations. This paper proposes a scheme to secure data provenance in the cloud while offering the encrypted search.

Keywords: Secure Data Provenance, Encrypted Cloud Storage, Security, Privacy

1 Introduction

Cloud storage has recently received great attention from the IT industry ranging from small-scale to large-scale enterprises. It offers flexibility of accessing data at any time from anywhere and any terminal, such as computers, laptops or hand-held devices. It not only provides scalability but also reduces IT infrastructure and management costs. The cloud raises security challenge of protecting data confidentiality. Thus, users may not trust the cloud provider for storing their data securely in dispersed geographical locations. Data security, which is of great concern for the users, is a strong obstacle in widespread adoption of the cloud for a number of applications involving sensitive data of the healthcare and banking domains.

Unfortunately, today's clouds are missing to manage data provenance. Data provenance describes how a particular piece of data has been produced. It is generated once the data is processed. An auditor can obtain it by querying the store where it is recorded. Data provenance plays a vital role in forensic analysis, enabling the collection of digital evidence by a post-incident investigation. It is widely used not only for forensics analysis but also for scientific collaborations and in legal proceedings. Generally, data provenance may include, but not limited to, what action was taken, who took it, where it was taken, why it was taken, how it was taken, when it was taken, in which environment it was taken and what the sequence of those actions is [18].

Consider a healthcare scenario where a patient visits the general physician, Dr. Alice, assigned to her. Dr. Alice performs a medical checkup and prepares a preliminary medical report based on her observations. Next, Dr. Alice recommends the patient to go for some medical tests in any medical lab. The patient goes to a medical lab where a lab assistant Bob conducts the medical tests recommended in the medical report. After conducting the medical tests, Bob adds the details of results in the medical report and gives it to the patient. The patient comes back to Dr. Alice with the medical report. Dr. Alice reviews the medical report and forwards the patient to the cardiologist, Dr. Charlie. Dr. Charlie reads the medical report and starts diagnosing the disease. In this scenario, data provenance describes how the medical report has been generated and who has worked on it and what the sequence of processing is. Since, each action taken on the medical report is recorded, the data provenance may answer the queries like *who took action on the medical report?*, *what actions are taken by Alice today?*, *did Bob and Charlie work on the medical report?*, etc. In other words, the data provenance may be obtained as a result of the query.

1.1 Motivation

The provenance of sensitive data may reveal some private information. For instance, in the above scenario, we can notice that even if the medical report is protected from unauthorised access, the data provenance still may reveal some information about a patient's sensitive data. That is, an adversary may deduce from the data provenance that the patient might have heart problems considering the fact that a cardiologist has processed patient's medical report. Therefore, in addition to provide protection to the sensitive data, it is vital to make the data provenance secure.

Another motivation to secure data provenance is for providing the unforgeability and non-repudiation. For instance, in the aforementioned healthcare scenario, assume that carelessness in reporting has resulted in the mis-diagnosis. In order to escape the investigation of mis-diagnosis, a victim would try to either forge the medical report with the fake data provenance or repudiate his/her involvement in generating the medical report. Moreover, the query to data provenance and the response should be encrypted, otherwise, a victim may threaten the auditor by eavesdropping the communication channel to check if his/her case is being investigated. The data, such as the medical report, may be critical; therefore, it should be subject to the availability at any time from anywhere.

A significant amount of research has been conducted on securing data provenance. For instance, secure data provenance schemes proposed in [9, 10] ensures confidentiality by employing state-of-the-art encryption techniques. However, this scheme does not address how an authorised auditor can perform search on data provenance. The scheme proposed in [12] provides anonymous access in the cloud environment for sharing data among multiple users. The scheme can track the real user if any dispute occurs. However, there is no detail about how the scheme manages data provenance in the cloud. Both [11] and [7] assume a trusted infrastructure, restricting the possibility of managing data provenance in cloud environments. Unfortunately, the existing research lacks in securing data provenance while offering search on data provenance stored in the cloud.

1.2 Research Contribution

This paper investigates the problem of securing data provenance in the cloud and proposes a scheme that supports encrypted search while protecting confidentiality of data provenance stored in the cloud. One of the main advantages of the proposed approach is that neither an adversary nor a cloud service provider learns about the data provenance or the query. Summarising, the research contributions of our approach are threefold. First of all, the proposed scheme ensures secure data provenance by providing confidentiality, integrity, non-repudation, unforgeability and availability in the cloud environment. Second, proposed solution is capable of handling complex queries involving non-monotonic boolean expressions and range queries. Third, the system entities do not share any keys and the system is still able to operate without requiring re-encryption even if a compromised user (or auditor) is revoked.

1.3 Organistaion

The rest of this report is organised as follows: Section 2 lists down the security properties that a secure data provenance scheme should guarantee. Section 3 provides a discussion of existing data provenance schemes based on the security properties listed in Section 2. Section 4 describes the threat model. The proposed approach is described in Section 5. Section 6 focuses on the solution details. Section 7 provides a discussion about how to optimise performance overheads incurred by the proposed scheme. Finally, Section 8 concludes this paper and gives directions for the future work.

2 Security Properties of a Data Provenance Scheme

A data provenance scheme must fulfil the general data security properties in order to guarantee the trustworthiness. In the context of data provenance, the security properties are described as follows:

- **Confidentiality:** Data provenance of a sensitive piece of data (that is, the source data) may reveal some private information. Therefore, it is necessary to encrypt not only the source data but also the data provenance. Moreover, a query to and/or a response from the data provenance store may reveal some sensitive information. Thus, both the query and its response must be encrypted in order to guarantee confidentiality on the communication channel. Last but not least, if data provenance is stored in the outsourced environment such as the cloud then the data provenance scheme must guarantee that neither the stored information nor the query and response mechanism must reveal any sensitive information while storing data provenance or performing search operations.
- **Integrity:** The data provenance is immutable. Therefore, the integrity must be ensured by preventing any kind of unauthorised modifications in order to get the trustworthy data provenance. The integrity guarantees that data provenance cannot be modified during the transmission or on the storage server without being detected.

– **Unforgeability:** An adversary may forge data provenance of the existing source data with the fake data. The unforgeability refers that the source data is tightly coupled with its data provenance. In other words, an adversary cannot forge the fake data with existing data provenance (or vice versa) without being detected.
– **Non-Repudiation:** Once a user takes an action, as a consequence, the data provenance is generated. A user must not be able to deny once data provenance has been recorded. The non-repudiation ensures that the user cannot deny if he/she has taken any actions.
– **Availability:** The data provenance and its corresponding source data might be critical; therefore, it must available at any time from anywhere. For instance, the life-critical data of a patient is subject to high availability, considering emergency situations that can occur at any time. The availability of the data can be ensured by a public storage service such as provided by the cloud service provider.

3 Related Work

In the following subsections, we describe state-of-the-art data provenance schemes which can be categorised as the **general data provenance schemes** and the **secure data provenance schemes**. The schemes in the former category are designed without taking into consideration the security properties while the schemes in the latter category explicitly aim at providing the certain security properties.

3.1 General Data Provenance Schemes

Several systems have been proposed for managing data provenance. Provenance-Aware Storage Systems (PASS) [15] is a first storage system towards the automatic collection and maintenance of data provenance. PASS collects information flow and workflow details at the operating system level by intercepting system calls. However, PASS does not focus on security of data provenance. Open Provenance Model (OPM) [14] is a model that has been designed as a standard. In OPM version 1.1 [13], data provenance can be exchanged between systems. Moreover, it defines how to represent data provenance at very abstract level. The focus of OPM is standardisation, however, it does not take into account the security and privacy issues related to data provenance. Muniswamy-Reddy *et al.* [16,17] explain how to introduce data provenance to a cloud storage server. They define a protocol to prevent forgeability between the data provenance and the source data. However, they leave the data provenance security as an open issue. Sar and Cao [19] propose *Lineage File System* that keeps record of data provenance of each file at the file system level. Unfortunately, they do not address the security and privacy aspects of the file system.

Table 1. Summary of data provenance schemes

Scheme	Year	Application Domain	Confidentiality				Integrity	Unforgeability	Non-Repudiation	Availability
			Provenance	Query	Response	Source Data				
Buneman *et al.* [5]	2001	Database System	-	-	-	-	-	-	-	-
Lineage File System [19]	2005	File System	No	No	No	No	No	No	No	-
PASS [15]	2006	Operating System	No	No	No	No	No	No	No	-
Tan *et al.* [20]	2006	SOA	Yes	-	-	-	Yes	Yes	Yes	-
OPM [14]	2008	-	-	-	-	-	-	-	-	-
Braun *et al.* [3]	2008	-	-	-	-	-	-	-	-	-
SPROV [9, 10]	2009	Operating System	Yes	-	-	No	Yes	Yes	Yes	-
Zhou *et al.* [21]	2009	Networks	No	-	-	-	Yes	-	Yes	-
ExSPAN [22]	2010	Networks	No	No	No	-	No	No	No	Yes
Muniswamy-Reddy and Seltzer [17]	2010	Cloud Storage	-	-	-	-	-	-	-	Yes
Muniswamy-Reddy *et al.* [16]	2010	Cloud Storage	No	No	No	No	No	Yes	No	Yes
Aldeco-Perez and Moreau [1]	2010	-	No	No	No	No	Yes	-	Yes	-
Lu *et al.* [12]	2010	Cloud Computing	Yes	-	-	Yes	Yes	Yes	Yes	Yes
PSecON [11]	2010	E-Science	Yes	-	-	Yes	Yes	Yes	Yes	Yes
Davidson *et al.* [6, 7]	2010 - 2011	-	-	-	-	-	-	-	-	-

'-' means not applicable

Buneman *et al.* [5][3] use the term *data provenance* to refer to the process of tracing and recording the origin of data and its movements between databases. Data provenance, as defined by [4], broadly refers to a description of the origins of a piece of data and the process by which it arrived in the database. They explain *why-provenance*, who contributed to or why a tuple is in the output, and *where-provenance*, where does the a piece of data comes from. Unfortunately, they do not focus on the security of data provenance.

Zhou *et al.* [21] use the notion of data provenance to explain the existence of a network state. However, they do not address the security of data provenance. In EXtenSible Provenance Aware Networked systems (ExSPAN) [22], Zhou *et al.* extend [21] and propose ExSPAN which provides the support for queries and maintenance of the network provenance in a distributed environment. However, they leave the issue of protecting the confidentiality and authenticity of provenance information as open.

3.2 Secure Data Provenance Schemes

The Secure Provenance (SPROV) scheme [9,10] automatically collects data provenance at the application layer. It provides security assurances of confidentiality and integrity of the data provenance. In this scheme, confidentiality is ensured by employing state-of-the-art encryption techniques while integrity is preserved using the digital signature of the user who takes any actions. Each record in the data provenance includes the signed checksum of previous record in the chain. For speeding up the auditing, they have introduced the spiral chain where the auditors can skip verification of records wrote by the users they already trust. However, the SPROV scheme has some limitations. First, it does not provide confidentiality to the source data whose data provenance is being recorded. Second, it does not provide any mechanisms to query data provenance. Third, it assumes that secret keys are never revoked or compromised. Last but not least, it cannot be employed in the cloud as it assumes a trusted infrastructure in order to store data provenance.

Jung and Yeom [11] propose the Provenance Security from Origin up to Now (PSecOn) scheme for e-science, a cyber laboratory to collaborate and share scientific resources. In an e-science grid, researchers can ensure integrity of the scientific results and corresponding data provenance through the PSecOn scheme. The PSecOn scheme ensures e-science grid availability from anywhere at any time. When an object is created, updated or transferred from one grid to another then the corresponding data provenance is prepared automatically. Each e-science grid has its own public history pool that manages the signature on data provenance, signed with the private key of an e-science grid. The public history pool prevents repudiation of both the data sender and the data receiver. The PSecOn scheme encrypts the source data. It revokes the secret key of a user who is compromised. However, it does not provide any query-response mechanisms to search data provenance. The main drawback of PSecOn is its strong assumption of relying on a trusted infrastructure, restricting the possibility of managing data provenance in the cloud.

[3] This is only the scheme that follows data-oriented approach while rest of the schemes in this paper are based on process-oriented approach.

Lu *et al.* [12] introduce a scheme to manage data provenance in the cloud where data is shared among multiple users. Their scheme provides users access to the online data. To guarantee confidentiality and integrity, a user encrypts and signs the data while a cloud service provider receives and verifies the signature before storing that data. Once the data is in dispute, a cloud service provider can provide the anonymous access information to a trusting authority who uses the master secret key of the system to trace the real user. The shortcoming of this approach is that it only traces the user while it does not provide any details about how the data provenance is managed by the cloud service provider.

Aldeco-Perez and Moreau [1] ensure integrity of data provenance by providing concrete cryptographic constructs. They describe the information flow of an auditable provenance system which consists of four stages including recording provenance, storing provenance, querying provenance and analysing provenance graph in order to answer questions regarding execution of the entities within the system. They ensure integrity at two levels. The first level is when data provenance is recorded and stored while the second level is at the analysis stage. Unfortunately, they do not provide any details about how to provide confidentiality to data provenance.

Braun *et al.* [3] focus on security model of data provenance at the abstract level. They consider data provenance as a causality graph with annotations. They argue that the security of data provenance is different from the source data it describes. Therefore, each of these need different access controls. However, they do not address how to define and enforce these access controls. Tan *et al.* [20] discuss security issues related to a Service Oriented Architecture (SOA) based provenance system. They address the problem of accessing data provenance for auditors with different access privileges. As a possible solution, they suggest to restrain auditors by limiting the access to the results of a query using cryptographic techniques. However, there is no concrete solution. Davidson *et al.* [7] consider the privacy issue while accessing and searching data provenance. In [6], Davidson *et al.* formalise the notion of privacy and focus on a mathematical model for solving privacy-preserving view as a result of query by an auditor. However, their approach is theoretic and there is no concrete construction for addressing security.

Table 1 summarises existing data provenance schemes based on the security properties listed in Section 2. Currently, there is not a single data provenance scheme that could guarantee all the security properties listed in Section 2.

4 Threat Model

This section describes the system entities involved, potential adversaries and possible attacks. The proposed system may include following entities:

- **User:** A User is an individual who takes action on the source data and generates data provenance. It is managed in the trusted environment. In the healthcare scenario, medical staff members, such as doctor and lab assistant, are Users.
- **Auditor:** An Auditor is the one who audits actions taken by a User. An Auditor also verifies data provenance up to the origin and identifies who took what action on the source data. An auditor may be an investigator or a regular quality assurance

checker to check processes within an organisation. It is managed in the trusted environment.

- **Cloud Service Provider (CSP):** A CSP is responsible for managing the source data and its corresponding data provenance in the cloud environment. It is assumed that a CSP is honest-but-curious, means it is honest to follow the protocol for performing required actions but curious to deduce stored or exchanged data provenance and the source data. The CSP guarantees the availability of data provenance store from anywhere at any time.
- **Trusted Key Management Authority (KMA):** The KMA is fully trusted and responsible for generating and revoking the cryptographic keys involved. For each authorised entity described above, the KMA generates and transmits the keys securely. The KMA requires less resource and less management efforts. Since a very limited amount of data needs to be protected, securing the KMA is much easier and it can be kept offline most of the time.

The proposed scheme assumes that a CSP will not mount active attacks such as modifying the exchanged messages, message flow and the stored data without being detected. The main goals of an adversary is to gain information from the data provenance record about the actions performed, the provenance chain, and modifying existing data provenance entries.

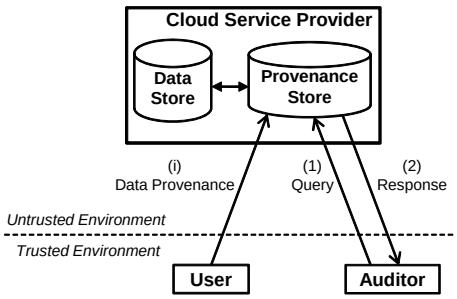

Fig. 1. An abstract architecture of the proposed scheme

5 The Proposed Approach

The proposed scheme provides the support for storing and searching data provenance in the cloud environment. The proposed scheme aims at providing the security properties listed in Section 2. In the proposed scheme, a CSP manages a Provenance Store to store data provenance. Moreover, the proposed scheme provides the support for storing the source data corresponding to the data provenance. The source data is stored in the Data Store which is also managed by a CSP. The CSP is in the untrusted environment while both the User and the Auditor are in the trusted environment. Figure 1 shows an abstract

architecture of the proposed scheme. In the proposed scheme, after a User has taken an action on the source data, he/she (i) sends the corresponding data provenance to the Provenance Store. An Auditor may (1) send a query to the Provenance Store and as a result he/she (2) obtains the Response.

Table 2. Representation of data provenance

Revision	Date	Time	User ID	Action				Previous Revision	Hash	Signature
				Name	Reason	Description	Location			
1	01-01-08	14:40:30	Alice	Create	Clinic Visit	Medical Report	Trento	0	X-bits	Y-bits
2	02-01-08	09:30:00	Bob	Append	Lab Visit	Blood Test	Rovereto	1	X-bits	Y-bits
⋮	⋮	⋮	⋮	⋮	⋮	⋮	⋮	⋮	⋮	⋮

Typically, X and Y may be of size 128, 512 or more.

5.1 Structure of a Data Provenance Entry

This subsection describes how data provenance may look like. Typically, a provenance record may include, but not limited to, the following fields:

- *Revision:* indicates the version number.
- *Date and time:* indicating when the action was taken.
- *User ID:* who took the action.
- *Action:* provides the details of action taken on the source data. It is divided into four parts: *Name*, *Reason*, *Description* and *Location*. *Name* describes what action was taken. *Reason* states why the action was taken. *Description* gives the additional information that may include how the action was taken. *Location* indicates where the action was taken.
- *Previous Revision:* indicates the version number of the previous action taken on the same source data.
- *Hash:* of the current source data after the action has been taken. This guarantees the unforgeability.
- *Signature:* is obtained after signing the hash of the all above fields with the private key of the User who took the action. This ensures the integrity and non-repudiation.

Once a User takes an action, the corresponding data provenance entry is sent to the Provenance Store. Table 2 illustrate how a typical data provenance entry looks like. The first entry in Table 2 has revision 1 with date 01-01-08 and time 14:40:30 hrs where action was taken by Alice who created a medical report when a patient visited her clinic located in Trento. The previous revision of this data provenance is 0 since it is the first entry. Bob adds the details of the blood test after that patient has visited his lab in Rovereto on 02-01-08 at 09:30:00 hrs. The previous revision corresponds to 1 as Bob is appending the existing medical report. Each entry includes the hash

of the corresponding source data and signatures of Alice and Bob on entry 1 and 2, respectively.

In order to support the search, for an Auditor, on the encrypted data provenance stored in the Provenance Store, each field of the data provenance entry is transformed in to string or numerical attributes. One string attribute represents a single element while a numerical attribute of size *n* bits represents *n* elements. In the proposed scheme, we consider that the maximum revision number possible is represented by a numerical attribute of size *m*. For the ease of understanding, let us assume that the value of *m* is 4. The first entry in Table 2 contains the revision with value 1, which is 0001 in a 4-bit representation. This can be transformed in to 4 elements, i.e., $0***$, $*0**$, $**0*$ and $***1$. The date can be considered as 3 numerical attributes, the first numerical attribute to represent day in 5 bits, the second numerical attribute to represent month in 4 bits and the third numerical attribute to represent year in 7 bits. Similarly, the time can be considered as 3 numerical attributes, the first numerical attribute to represent hour in 5 bits, the second numerical attribute to represent minute in 6 bits and the third numerical attribute to represent second in 6 bits. The user ID is a string attribute. Each sub-field of action can be treated as a string attribute. The previous revision is again a numerical attribute of size *m*. In the proposed scheme, we omit the search support for the hash and the signature fields as we assume that an Auditor cannot query based on these fields as these are just large numbers of size X and Y bits, respectively. Typically, one can have both the User and the Auditor roles simultaneously.

The source data is stored in the Data Store managed by the CSP. For each revision in the Provenance Store, there is a corresponding data item in the Data Store. In other words, the Data Store maintains a table containing two columns: one column to keep the revision while the other to store the source data item after the action has been taken.

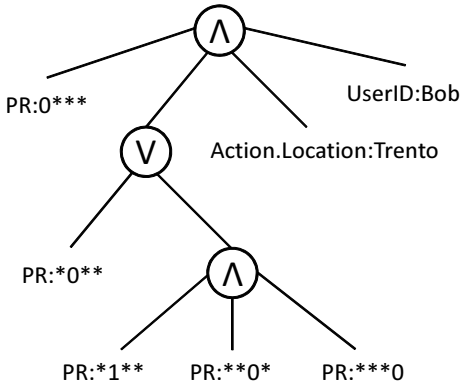

Fig. 2. Query representation

5.2 Query Representation

This section provides an informal description of the query representation used in the proposed scheme. To represent the query, we use the tree structure similar to one used in [2]. The tree structure of the query allows an Auditor to express conjunctions and disjunctions of equalities and inequalities. Internal nodes of the tree represent AND and OR gates while leaf nodes represent the values of conditional predicates. The tree employs the representation of *bag of bits* in order to support comparison between numerical values. Let us consider that an Auditor sends the following query: search all actions taken by Bob in Trento with previous revision (PR) between 1 to 4. Alternatively, this query can be written as follows: $UserID = Bob$ AND $Action.Location = Trento$ AND $PR \geq 1$ AND $PR \leq 4$. The query is illustrated in Figure 2.

6 Solution Details

The main idea is to perform encryption for providing confidentiality to the data provenance both on the communication channel and in the cloud. In order to search the data provenance, an Auditor sends a query that is also encrypted. In fact, the search is performed in an encrypted manner, which is based on the Searchable Data Encryption (SDE) proposed by Dong *et al.* [8]. The SDE scheme allows an untrusted server to perform search on the encrypted data without revealing information about the data provenance or the query. The advantage of this scheme is the multi-user support without requiring any key sharing between Auditors/Users. In other words, each Auditor or User has a unique set of keys. The data provenance encrypted by a User can be searched and decrypted by an authorised Auditor. However, the SDE scheme in [8] only allows an Auditor to perform query containing comparison based on equalities. For supporting complex queries, we extend the SDE scheme to handle complex boolean expressions such as non-conjunctive and range queries in the multi-user settings.

In addition to providing support for search on the encrypted data provenance, each entry of the data provenance is encrypted using Proxy Encryption (PE) scheme proposed in [8]. In other words, an Auditor performs search on the encrypted data provenance using the extended version of the SDE scheme while the searched data corresponding to the query is accessed by the PE scheme. Furthermore, the source data corresponding to the data provenance is also encrypted using the PE scheme. The proposed solution guarantees all the security properties listed in Table 1 that the existing research on data provenance lacks.

In general, there are three main phases in the data provenance life cycle: the first phase is the **storing data provenance** in to the Provenance Store; the second phase is the **searching data provenance** when an Auditor sends a query; and the third phase is the **accessing data provenance**. In the following, we provide the details of algorithms involved in each phase, where the SDE and the PE schemes are used in each phase.

6.1 Intialisation

In this phase, the proposed scheme is intialised for generating the required keying material for all involved entities in the system.

- $Init(1^k)$: The Trusted KMA takes as input the security parameter 1^k and outputs two prime numbers p, q such that q divides $p-1$, a cyclic group $\mathbb{G}$ with a generator g such that $\mathbb{G}$ is the unique order q subgroup of $\mathbb{Z}_P^*$. It chooses $x \xleftarrow{R} \mathbb{Z}_q^*$ and computes $h = g^x$. Next, it chooses a collision-resistant hash function H, a pseudorandom function f and a random key s for f. Finally it publicises the public parameters $Params = (\mathbb{G}, g, q, h, H, f)$ and keeps securely the master secret key $MSK = (x, s)$.
- $KeyGen(MSK, i)$: For each User (or Auditor) i, the Trusted KMA chooses $x_{i1} \xleftarrow{R} \mathbb{Z}_q^*$ and computes $x_{i2} = x - x_{i1}$. It securely transmits $K_{u_i} = (x_{i1}, s)$ to the User (or Auditor) i and $K_{s_i} = (i, x_{i2})$ to the CSP which inserts K_{s_i} in the Key Store, that is, $K_S = K_S \cup K_{s_i}$[4].

6.2 Storing Data Provenance

During this phase, a User takes an action and creates a data provenance entry and the source data which are encrypted using the SDE and PE schemes. For both the SDE and PE schemes, the first round of encryption is performed by the User while the Second round of encryption is performed by the CSP. After this phase, the data provenance entry is stored in the Provenance Store while the source data is stored in the Data Store.

- $Hash(D)$: The User calculates hash over the source data D and populates the hash field of the data provenance entry with the calculated value.
- $Signature(e, K_{u_i})$: The User i calculates a hash $H(e)$ over the all fields (except the signature) in a data provenance entry e. Then, the User populates the signature field of the data provenance entry with the value calculated as follows: $g^{-x_{i1}} H(e)$.
- $User\text{-}SDE(m, K_{u_i})$: The User encrypts each element m of the fields (except the hash and the signature) of the data provenance entry in order to support encrypted search. The User chooses $r \xleftarrow{R} \mathbb{Z}_q^*$ and computes $c_i^*(m) = (\hat{c}_1, \hat{c}_2, \hat{c}_3)$ where $\hat{c}_1 = g^{r+\sigma}, \sigma = f_s(m), \hat{c}_2 = \hat{c}_1^{x_{i1}}, \hat{c}_3 = H(h^r)$. The User transmits $c_i^*(m)$ to the CSP.
- $User\text{-}PE(m, D, K_{u_i})$: The User encrypts each element m of the fields (except the hash and the signature) of the data provenance entry and the source data D. The User chooses $r \xleftarrow{R} \mathbb{Z}_q^*$ and outputs the ciphertexts $PE_i^*(m) = (g^r, g^{r x_{i1}} m)$ and $PE_i^*(D) = (g^r, g^{r x_{i1}} D)$, which are sent to the CSP.
- $Server\text{-}SDE(i, c_i^*(m), K_{s_i})$: The CSP retrieves the key K_{s_i} corresponding to the User i from the Key Store. Each User encrypted element $c_i^*(m)$ is re-encrypted to $c(m) = (c_1, c_2)$, where $c_1 = (\hat{c}_1)^{x_{i2}}$. $\hat{c}_2 = \hat{c}_1^{x_{i1}+x_{i2}} = (g^{r+\sigma})^x = h^{r+\sigma}$ and $c_2 = \hat{c}_3 = H(h^r)$. The re-encrypted entry $c(e)$ (where each $c(m) \in c(e)$) of data provenance is stored in the Provenance Store.
- $Server\text{-}PE(i, PE_i^*(m), PE_i^*(D), K_{s_i})$: The CSP retrieves the key K_{s_i} corresponding to the User i from the Key Store. Each User encrypted element $PE_i^*(m)$ is re-encrypted to $PE(m) = (p_1, p_2)$, where $p_1 = g^r$ and $p_2 = (g^r)^{x_{i2}} g^{r x_{i1}} m = g^{r(x_{i1}+x_{i2})} m = g^{rx} m$. Similarly, the $PE_i^*(D)$ is re-encrypted to $PE(D)$. Finally, the ciphertexts $PE(e)$ (where $PE(m) \in PE(e)$) and $PE(D)$ are sent to and stored[5] in the Provenance Store and the Data Store, respectively.

[4] The Key Store is initialised as $K_S = \Phi$.

[5] In the CSP, each entry $c(e)$ of the data provenance corresponds with the ciphertexts $PE(e)$ and $PE(D)$.

6.3 Searching Data Provenance

During this phase, an Auditor encrypts the search query and then sends the search query to the CSP. The CSP performs the encrypted matching against data provenance entires in the Provenance Store.

- *Auditor-Query-Enc*(Q, K_{u_j}) : An Auditor transforms the query in to a tree structure Q, as shown in Figure 2. The tree structure Q denotes a set of string and numerical comparisons. Each non-leaf node a' in Q represents a threshold gate with the threshold value $k_{a'}$ denoting the number of its children subtrees that must be satisfied where a' has total $c_{a'}$ children subtrees, i.e. $1 \leq k_{a'} \leq c_{a'}$. If $k_{a'} = 1$, the threshold gate is an OR and if $k_{a'} = c_{a'}$, the threshold gate is an AND. Each leaf node a represents either a string comparison or subpart of a numerical comparison (because one numerical comparison of size n bits is represented by n leaf nodes at the most) with a threshold value $k_a = 1$. For every leaf node $a \in Q$, the Auditor chooses $r \xleftarrow{R} \mathbb{Z}_q^*$ and computes trapdoor $T_j(a) = (t_1, t_2)$ where $t_1 = g^{-r} g^{\sigma}$ and $t_2 = h^r g^{-x_{j1} r} g^{x_{j1}\sigma} = g^{x_{j2} r} g^{x_{j1}\sigma}$, where $\sigma = f_s(a)$. The Auditor encrypts all leaf nodes in Q and sends the encrypted tree structure $T_j^*(Q)$ to the CSP.
- *Server-Search*$(j, T_j^*(Q), K_{s_j}, c(e))$: The CSP receives the encrypted tree structure $T_j^*(Q)$. Next, it it retrieves the key K_{s_j} corresponding to the Auditor j and the data provenance entries. For each encrypted entry $c(e)$, the CSP runs a recursive algorithm starting from the root node of $T_j^*(Q)$. For each non-leaf node, it checks if the number of children that are satisfied is greater than or equal to the threshold value of the node. If so, the node is marked as satisfied. For each encrypted leaf node $T_j^*(a) \in T_j^*(Q)$, there may exist a corresponding encrypted element $c(m) \in c(e)$. In order to perform this check, it computes $T = t_1^{x_{j2}} \,.\, t_2 = g^{x\sigma}$ and tests if $c_2 \stackrel{?}{=} H(c_1 . T^{-1})$. If so, the leaf node is marked as satisfied. After running the recursive algorithm, if the root node of the encrypted tree structure $T_j^*(Q)$ is marked as satisfied then the entry $c(e)$ is marked as matched. This algorithm is performed for each encrypted entry $c(e)$ in the Provenance Store and it finds sets of ciphertexts $PE(e)$ and $PE(D)$ corresponding to the matched entries.

6.4 Accessing Data Provenance

During this phase, the data provenance entries can be accessed and then ultimately be verified by the Auditor. First, the CSP performs one round of decryption for sets of ciphertexts found during the search. The Auditor performs the second round of decryption to access data provenance and its corresponding source data. Furthermore, an Auditor gets the verification key from the CSP in order to verify the signature on the data provenance entries.

- *Server-Pre-Dec*$(j, PE(e), PE(D), K_{s_j})$: The CSP retrieves the key K_{s_j} corresponding to the Auditor j from the Key Store. Each encrypted element $PE(m) \in PE(e)$ is pre-decrypted by the CSP as $PE_j(m) = (\hat{p}_1, \hat{p}_2)$, where $\hat{p}_1 = g^r$ and $\hat{p}_2 = g^{rx} m \,.\, (g^r)^{-x_{j2}} = g^{r(x - x_{j2})} m = g^{r x_{j1}} m$. Similarly, $PE(D)$ is pre-decrypted by the CSP as $PE_j(D)$. Finally, the ciphertexts $PE_j(e)$ and $PE_j(D)$ are sent to the Auditor.

– *Auditor-Dec*$(PE_j(e)), PE_j(D)), K_{u_j})$: Finally, the Auditor decrypts the ciphertext $PE_j(m) \in PE_j(e)$ as follows: $\hat{p}_2 \,.\, \hat{p}_1^{-x_{j1}} = g^{rx_{j1}} m \,.\, g^{-rx_{j1}} = m$. Similarly, the source data D is retrieved from $PE_j(D))$.
– *Get-Verification-Key*(i) : In the proposed solution, an Auditor may verify the signature by first obtaining the verification key of the User who took the action. This algorithm is run by the CSP. It takes an input the User ID i. For calculating the verification key, the CSP first obtains the key $K_{s_i} = (i, x_{i2})$ corresponding to the User i and then calculates the verification key as follows: $h \,.\, g^{-x_{i2}} = g^x \,.\, g^{-x_{i2}} = g^{x-x_{i2}} = g^{x_{i1}}$.
– *Verify-Signature-Key*$(e, g^{-x_{i1}} H(e'), g^{x_{i1}})$: Given the signature $g^{-x_{i1}} H(e')$ over the data provenance entry e and the verification key $g^{x_{i1}}$, an Auditor can verify the signature first by calculating $g^{-x_{i1}}\ H(e')\ g^{x_{i1}} = H(e')$. Next, an Auditor calculates the hash over the data provenance entry e as $H(e)$. Finally, an Auditor checks if $H(e) \stackrel{?}{=} H(e')$. If so, the signature verification is successful and this algorithm returns *true* and *false* otherwise.

6.5 Revocation

In the proposed solution, it is possible to revoke a compromised User (or Auditor). This is accomplished by the CSP.

– *Revoke*(i) Given the User (or Auditor) i, the CSP removes the corresponding key K_{s_i} from the Key Store as $K_S = K_S \setminus K_{s_i}$. Therefore, the CSP needs to check the revocation of a User or an Auditor before invoking any actions including storing, searching and accessing the data provenance.

7 Discussion

This section provides a discussion about how to optimise the storage and performance overheads incurred by the proposed scheme.

7.1 Storage Optimisation

The storage can be optimised if the source data changes are stored as difference (as is done in any subversion system) instead of managing a complete source data item against each revision. In other words, the complete source data item is stored against the first revision while for the subsequent revisions, only the changes are stored.

7.2 Performance Optimisation

In order to improve the search performance, the indexing and partitioning of the data provenance can be done. However, this is subject to the future work. Moreover, the performance at the Auditor level can be improved by maintaining a list of verification keys of the Users who are taking actions very frequently instead of interacting each time with the CSP.

8 Conclusion and Future Directions

This paper has investigated the problem of securing provenance and presented a proposed scheme that supports encrypted search while protecting confidentiality of data provenance stored in the cloud, given the assumption that the CSP is honest-but-curious. The main advantage of our proposed scheme is that neither an adversary nor a cloud service provider learns about the data provenance or the query. The proposed solution is capable of handling complex queries involving non-monotonic boolean expressions and range queries. Finally, the system entities do not share any keys and even if a compromised User (or Auditor) is revoked, the system is still able to perform its operations without requiring re-encryption.

As future research directions, the proposed solution will be formalised in more rigorous terms to prove its security features. Moreover, a prototype would be developed for estimating the overhead incurred by the cryptographic operations of the proposed scheme. Other long-term research goals are 1) how to apply the scheme in the distributed settings 2) to investigate how to make such architecture more efficient in terms of query-response time without compromising the security properties.

Acknowledgment

The work of the first and third authors is supported by the EU FP7 programme, Research Grant 257063 (project Endorse).

References

1. Roco Aldeco-Prez and Luc Moreau. Securing provenance-based audits. In Deborah McGuinness, James Michaelis, and Luc Moreau, editors, *Provenance and Annotation of Data and Processes*, volume 6378 of *Lecture Notes in Computer Science*, pages 148–164. Springer Berlin / Heidelberg, 2010.
2. John Bethencourt, Amit Sahai, and Brent Waters. Ciphertext-policy attribute-based encryption. In *IEEE Symposium on Security and Privacy*, pages 321–334. IEEE Computer Society, 2007.
3. Uri Braun, Avraham Shinnar, and Margo Seltzer. Securing provenance. In *Proceedings of the 3rd conference on Hot topics in security*, pages 4:1–4:5, Berkeley, CA, USA, 2008. USENIX Association.
4. Peter Buneman, Sanjeev Khanna, and Wang Chiew Tan. Data provenance: Some basic issues. In *Proceedings of the 20th Conference on Foundations of Software Technology and Theoretical Computer Science*, FST TCS 2000, pages 87–93, London, UK, 2000. Springer-Verlag.
5. Peter Buneman, Sanjeev Khanna, and Wang Chiew Tan. Why and Where: A Characterization of Data Provenance. In Jan Van den Bussche and Victor Vianu, editors, *International Conference on Database Theory*, pages 316–330. Springer, LNCS 1973, 2001.
6. Susan B. Davidson, Sanjeev Khanna, Sudeepa Roy, and Sarah Cohen Boulakia. Privacy issues in scientific workflow provenance. In *Proceedings of the 1st International Workshop on Workflow Approaches to New Data-centric Science*, Wands '10, pages 3:1–3:6, New York, NY, USA, 2010. ACM.

7. Susan B. Davidson, Sanjeev Khanna, Sudeepa Roy, Julia Stoyanovich, Val Tannen, and Yi Chen. On provenance and privacy. In *Proceedings of the 14th International Conference on Database Theory*, ICDT '11, pages 3–10, New York, NY, USA, 2011. ACM.
8. Changyu Dong, Giovanni Russello, and Naranker Dulay. Shared and searchable encrypted data for untrusted servers. *J. Comput. Secur.*, 19:367–397, August 2011.
9. Ragib Hasan, Radu Sion, and Marianne Winslett. The case of the fake picasso: preventing history forgery with secure provenance. In *Proccedings of the 7th conference on File and storage technologies*, pages 1–14, Berkeley, CA, USA, 2009. USENIX Association.
10. Ragib Hasan, Radu Sion, and Marianne Winslett. Preventing history forgery with secure provenance. *Trans. Storage*, 5:12:1–12:43, December 2009.
11. Im Y. Jung and Heon Y. Yeom. Provenance security guarantee from origin up to now in the e-science environment. *Journal of Systems Architecture*, In Press, Corrected Proof:–, 2010.
12. Rongxing Lu, Xiaodong Lin, Xiaohui Liang, and Xuemin (Sherman) Shen. Secure provenance: the essential of bread and butter of data forensics in cloud computing. In *Proceedings of the 5th ACM Symposium on Information, Computer and Communications Security*, ASIACCS '10, pages 282–292, New York, NY, USA, 2010. ACM.
13. Luc Moreau, Ben Clifford, Juliana Freire, Joe Futrelle, Yolanda Gil, Paul Groth, Natalia Kwasnikowska, Simon Miles, Paolo Missier, Jim Myers, Beth Plale, Yogesh Simmhan, Eric Stephan, and Jan Van den Bussche. The open provenance model core specification (v1.1). In *Future Generation Computer Systems*, 2010. (In Press).
14. Luc Moreau, Juliana Freire, Joe Futrelle, Robert E. Mcgrath, Jim Myers, and Patrick Paulson. Provenance and annotation of data and processes. chapter The Open Provenance Model: An Overview, pages 323–326. Springer-Verlag, Berlin, Heidelberg, 2008.
15. Kiran-Kumar Muniswamy-Reddy, David A. Holland, Uri Braun, and Margo Seltzer. Provenance-aware storage systems. In *Proceedings of the annual conference on USENIX '06 Annual Technical Conference*, pages 4–4, Berkeley, CA, USA, 2006. USENIX Association.
16. Kiran-Kumar Muniswamy-Reddy, Peter Macko, and Margo Seltzer. Provenance for the cloud. In *Proceedings of the 8th USENIX conference on File and storage technologies*, FAST'10, pages 15–14, Berkeley, CA, USA, 2010. USENIX Association.
17. Kiran-Kumar Muniswamy-Reddy and Margo Seltzer. Provenance as first class cloud data. *SIGOPS Oper. Syst. Rev.*, 43:11–16, January 2010.
18. Sudha Ram and Jun Liu. Understanding the semantics of data provenance to support active conceptual modeling. In Peter Chen and Leah Wong, editors, *Active Conceptual Modeling of Learning*, volume 4512 of *Lecture Notes in Computer Science*, pages 17–29. Springer Berlin / Heidelberg, 2007.
19. Can Sar and Pei Cao. Lineage file system, 2005. Available at: `http://theory.stanford.edu/~cao/lineage`.
20. Victor Tan, Paul Groth, Simon Miles, Sheng Jiang, Steve Munroe, Sofia Tsasakou, and Luc Moreau. Security Issues in a SOA-based Provenance System. In *Proceedings of the International Provenance and Annotation Workshop (IPAW'06)*, volume 4145, pages 203–211, Chicago, Illinois, 2006. Springer-Verlag.
21. Wenchao Zhou, Yun Mao, Boon Thau Loo, and Martín Abadi. Unified declarative platform for secure netwoked information systems. In *Proceedings of the 2009 IEEE International Conference on Data Engineering*, pages 150–161, Washington, DC, USA, 2009. IEEE Computer Society.
22. Wenchao Zhou, Micah Sherr, Tao Tao, Xiaozhou Li, Boon Thau Loo, and Yun Mao. Efficient querying and maintenance of network provenance at internet-scale. In *Proceedings of the 2010 international conference on Management of data*, SIGMOD '10, pages 615–626, New York, NY, USA, 2010. ACM.