International Journal of Parallel Programming (2020) 48:344–366
https://doi.org/10.1007/s10766-018-0601-y

A Secure Provenance Scheme for Detecting Consecutive Colluding Users in Distributed Networks

Idrees Ahmed[1] · Abid Khan[1] · Adeel Anjum[1] · Mansoor Ahmed[1] · Muhammad Asif Habib[2]

Received: 8 June 2018 / Accepted: 8 September 2018 / Published online: 25 September 2018

Abstract

Data provenance is becoming extremely important these days for distributed environment, due to the ease in sharing and modifying data stored (e.g. cloud storage systems). However, the protection of provenance chain has been greatly understudied problem. This paper presents a secure provenance scheme for a distributed environment, designed to ensure data confidentiality, integrity, and non-repudiation. Specifically, the proposed scheme is designed to detect attacks on a provenance chain launched by multiple concurrent adversaries, such as forged provenance records and provenance record shuffling attacks. Moreover, the proposed scheme detects the provenance record, which has been perturbed and identifies the malicious or compromised user. We then evaluate our scheme empirically and analytically with the state of the art to demonstrate its security and performance in terms of computational and storage overheads.

Keywords Secure provenance · Integrity · Collusion attacks · Aggregated signatures

✉ Abid Khan
abidkhan@comsats.edu.pk

Idrees Ahmed
idrees.ciit@gmail.com

Adeel Anjum
adeel.anjum@comsats.edu.pk

Mansoor Ahmed
mansoor@comsats.edu.pk

Muhammad Asif Habib
dr.m.asif.habib@gmail.com

[1] Department of Computer Science, COMSATS University, Islamabad, Pakistan

[2] National Textile University, Faisalabad, Pakistan

1 Introduction

Provenance in computing generally refers to the genealogy or history of objects, in the sense of who created a piece of data, and who performed what operations on the data during its life cycle (i.e. information about the activity, user and the entity involved in producing the data) [1]. Provenance has applications in a number of domains, such as databases [2, 3], workflow systems and e-science [4–6], Web [7, 8], and file systems [9–11]. With the increasing digitalization of data, the distributed nature of data storage and dissemination, and the ease where digital data can be modified, it is important to ensure the provenance of the data. In other words, we need to be able to trust data coming from diverse sources to make accurate decisions, particularly in critical missions such as battlefields. Data provenance is not linear, in the sense that it can be represented as a directed acyclic graph (DAG) where every provenance record has a relationship with other records.

Thus, in such a chained structure data, simply encrypting the data is not likely to ensure the data's provenance, as we also need to protect the relationships of the data with other records. Generally, a secure provenance scheme should provide the following properties.

- *Confidentiality* Ensures unauthorized access to or disclosure of the data (e.g. agents, entities, and activities).
- *Tamper resilient (also known as integrity)* Integrity ensures that the provenance record or chain cannot be forged/modified undetected. A secure provenance scheme should be tamper resilient even in the event that one or more colluding users attempted to forge the data.
- *Non-repudiation* Once the provenance record is made, a user cannot repudiate his/her actions (e.g. using digital signature schemes).
- *Availability* Source data and provenance information should be available anytime whenever queried, for as long as permitted or required by the data retention policy/legislation.

As previously discussed, a number of provenance schemes have been proposed to for distributed environment, such as those of Hassan et al. [12, 13], Wang et al. [14] and Rangwala et al. [15]. Existing security solutions are generally tamper resilient only when two or more consecutive adversaries cannot launch attacks concurrently, as these schemes, among users, assume transitive trust. Such an assumption is not practical, particularly in a distributed network (e.g. federated cloud environment). In addition, existing schemes generally require an additional element for provenance records chaining (e.g. public keys of users).

Seeking to address the above two limitations, we present a scheme that is tamper-evident even when provenance chain and data are forged by several consecutive adversaries. Also, our scheme aggregates the signature of users to chain the provenance records; thus, reducing the storage overheads. In our scheme, provenance information is represented and stored in an XML file. We then prove the security of the proposed scheme and evaluate its performance. The main contributions of the paper are listed below.

- Detection of attacks by multiple consecutive adversaries besides providing confidentiality, integrity, and non-repudiation.
- Detection of attacks launched by malicious users to shuffle the order of provenance records.
- Our scheme chains the provenance records by aggregating the signatures of users without introducing an extra element in a provenance record.
- Our scheme presents the security analysis to validate the results and assurance of security properties of a secure provenance scheme.
- We evaluate our scheme empirically and analytically to demonstrate its security and performance in terms of computational and storage overheads. The verification with current state-of-the-art shows promising results.

The rest of the paper is organized as follows: In Sect. 2 a detailed overview of related works is provided. Background and models are presented in Sect. 3. The proposed scheme is presented in Sect. 4, and its performance evaluation is given in Sect. 5. The security analysis of the proposed scheme is given in Sect. 6. Section 7 concludes this paper.

2 Related Work

In this section, we provide an overview of schemes, which are related to our work. Kairos [16] is a framework designed to protect the ownership and temporal data in a grid using digital signatures and time-stamp protocol (TSP). Although the authors demonstrate that provenance record is protected in their framework, there is no discussion on the structure of provenance record and chaining. In a separate work, the authors in [17] formalize the requirements for location provenance in personal mobile devices. To protect location provenance, hash chain, block hash chain, Bloom filter, shadow hash chain, multi-linked hash chain, and RSA accumulator are used. Hassan et al. [18] propose a witness-endorsed scheme for personal mobile phone and location-aware services. The scheme uses a hash chain and bloom filter to avoid collusion and tampering attacks in location proofs. A secure framework, based on [18, 17], is proposed in [19] to securely generate the location proofs in mobile devices. Asghar et al. [20] propose a scheme for the cloud to protect the provenance and communication channel. Similarly, in [21], a secure provenance scheme for data forensic for the cloud is proposed. Bilinear pairing is used to protect the user's privacy. However, the scheme does not protect the provenance records. Izuan et al. [22] propose a trust model for cloud computing using secure data provenance. To achieve the properties of a secure provenance scheme, this framework works in two phases, namely: protecting user privacy or anonymity is achieved via authentication, and protecting data provenance is achieved using cryptographic techniques. Storing and accessing the provenance data are the two processes involved in this trust model. In [23], a PDP scheme is proposed for the cloud to protect the privacy of users. Hash chaining and group signatures are used for securing the provenance data in the cloud. SECAP is proposed in [24] to achieve secure application provenance in the cloud. The scheme uses the digital signatures and bloom filter to protect the provenance information. Similarly, ProvChain [25], a provenance framework is proposed for the cloud environment. Blockchain technology is used to ensure privacy and availability of data provenance (history of operations). Xu et al. [26] propose a framework for a secure provenance management. In the framework, there is a layer in between the application and physical device. Sultana et al. propose a scheme for secure provenance transmission in sensor networks. The provenance information is hidden inter-packet delays. At the destination, provenance data is retrieved using a threshold-based mechanism. For secure transmission of provenance data in the sensor network, spread spectrum watermarking technique is used [27]. Sultana et al. [28, 29] utilized in-packet bloom filter to encode the provenance in, which detects the packet dropping and provenance forgery attacks. Hussain et al. [30] propose a secure provenance compression technique for wireless sensor networks using arithmetic coding. To detect unauthorized or malicious modifications of packet and provenance, this scheme binds the data packet and its associated provenance along with its path. Integrity and authenticity of the data packet and its associated provenance are achieved using the secure packet sequence number generation along with AM-FM sketch technique. A provenance compression scheme using the dynamic Bayesian network is proposed for sensor network by Wang et al. [31]. This scheme applies the same techniques as [30], to secure the data packet and its provenance. However, the main focus of this scheme is provenance compression. In [32], a dictionary-based scheme for sensor networks is designed to achieve provenance security and compression. Distributed message digest mechanism is used to achieve confidentiality, integrity, and availability of data packet and its provenance.

Similar to Wang et al. [12, 13], an Onion scheme is proposed by Hassan et al. to detect integrity and confidentially attacks on provenance data. In [14], PKLC is proposed for distributed information network, designed to mitigate some of the limitations of Onion scheme. For provenance integrity, and ensuring data confidentiality, RSA, AES, and SHA-1 are used. A mutual-agreement scheme is proposed by Rangwala et al. [15] to detect confidentiality and integrity attacks on data provenance. For non-repudiation, digital signatures are used. Amril et al. [33] propose a scheme to protect the directed acyclic graph model of provenance. They use digital signatures to maintain the integrity of provenance data. The data owner and contributor both sign the nodes and their relationships. To ensure data confidentiality, they use path- based access control and compartment based access control. Schaler et al. [34] proposed a watermarking based scheme for reliable provenance for multimedia data.

Aman et al. [35] propose a provenance scheme for Internet of Things (IoT) devices. Physical Unclonable Function (PUF) and symmetric encryption are used to ensure the device's physical and provenance security. Message Authentication Code (MAC) is deployed to ensure data integrity. A detailed overview of secure provenance schemes for various distributed systems from the perspective of trustworthy data is provided in our paper [36]. Fuzel et al. [37] presented a secure provenance using a Merkle hash tree (MHT) as an authenticated data structure. However, this scheme required the involvement of a trusted auditor to generate the root of the tree.

3 Background and Models

In this section, we first provided the necessary background knowledge, definitions, and terminologies in the subsection Preliminaries. Then, we described the system model and the threat model for our proposed scheme.

3.1 Preliminaries

Here, we discuss the relevant concepts, system, and threat model.

- *Document* The document could be a word file, PDF file, website, or database tuple, created and modified by agents for which provenance needs to be generated. In the context of this paper, we will use a word file created and modified by the user as the document, which is modified each time when it is sent to another user resulting in a provenance record being added to the chain.
- *Provenance chain* Whenever a document is created or modified, the action performed by users is recorded in form of provenance record denoted by PR_i (‘i’ represents the provenance record’s number). Provenance records collectively make a provenance chain $\{PR_1, PR_2 \ldots PR_n\}$. Thus, the provenance chain is a complete history of the document being created and modified by the users.
- *Agent* Registered users who perform operations on the document are referred to as agents. An agent can be a user, owner of the document, a malicious attacker, or a compromised user.
- *Activity* Operations performed on a document, such as create, read, write and delete, are referred to as activity.
- *Trusted auditor* This is a trusted and powerful third party tasked to verify the provenance chain.
- *Adversary* An adversary is one who seeks to attempt undetected forgeries on the provenance chain, and the adversary can be an external attacker or a compromised user (which also captures the notion of a malicious insider, such as consecutive colluding users described next).
- *Consecutive Colluding Users* Consecutive adversaries are attackers who seek to launch attacks on a provenance chain to forge, add or re- move provenance records collaboratively. Suppose Agent_2 and Agent_3 are consecutive adversaries and they collude to launch attacks on provenance chain. Such attacks are not considered in existing schemes, such as those reported in [12–15] (i.e. these schemes assume transitive trust among participating users; thus, implying that only non-consecutive users collude to initiate attacks on the provenance chain, e.g. Agent_2 and Agent_4 are non-consecutive). Unlike these schemes, our scheme is designed to detect attacks launched by consecutive colluding users.

3.2 System Model

Our system model consists of trusted auditor(s) and a number of users working collaboratively to perform specified task(s), and these registered users are denoted Agents ($Agent=\{Agent_1, Agent_2, Agent_3, \ldots, Agent_n\}$)—see Fig. 1.

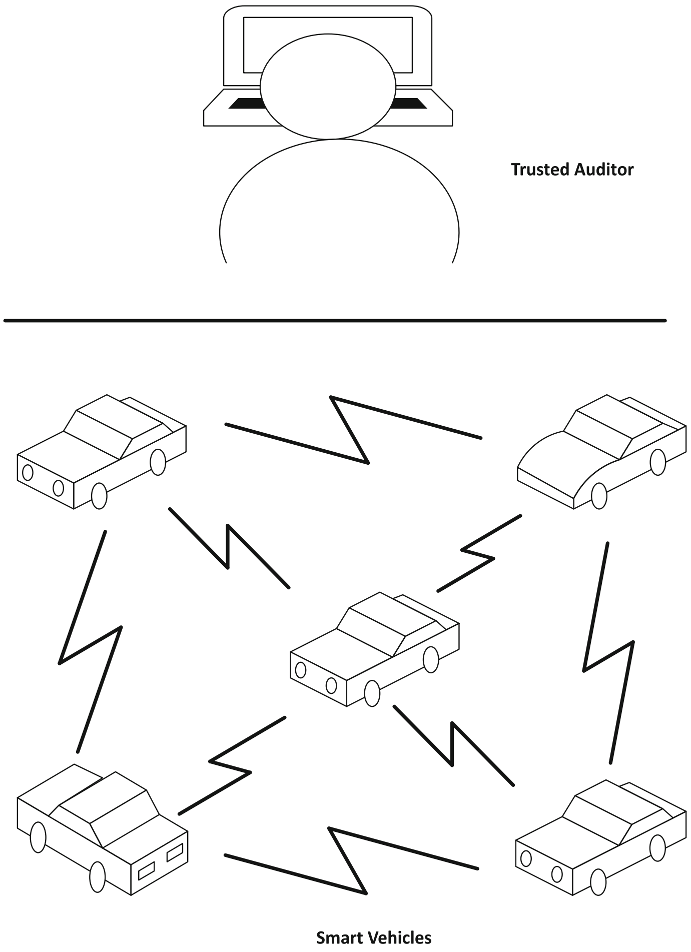


Fig. 1 System model

3.3 Threat Model

In our threat model, the adversary has the following capabilities.

- A malicious user or multiple colluders, and can add and/or remove selective provenance records from the provenance chain.
- An adversary can repudiate the actions performed on a document.
- An attacker can forge existing provenance records, user signature, and history of ownership.
- An attacker can gain access to the information of a provenance record.
- A malicious user acting alone or colluding with other consecutive users can shuffle the order of provenance records.

4 Proposed Secure Provenance Scheme

A provenance chain is made up of provenance records in a chronological order. The provenance record provides information about the agents and operations on a document, and the structure of each provenance record in our scheme is as follows:

$$PR_i = \left\{Agent_i^*,\ Activity_i^*,\ S_i^*,\ H\left(Doc_i\right),\ Sig_i^*,\ RV\right\}$$

In the above provenance record, $Agent_i^*$ is an encrypted *ID* of the agent who creates or modifies a document, $Activity_i^*$ is the encoded information about the operation (create, delete and write), S_i^* is the encrypted symmetric key of $Agent_i$, and $H\ (Doc_i)$ represents the hash value of the document Doc_i, Doc_n is the final stage of the document, and Sig_i is the signature of $Agent_i$. Sig_i includes the signed hash of $Agent_i^*$, $Activity_i^*$, S_i^*, and $H(Doc_i)$, and the next signature is an aggregated signature: $Sig_2^* = Sig_1 \oplus Sig_2$. Non- repudiation attacks can be detected by inspecting the signature field. In our scheme, provenance records chaining is achieved by aggregating signatures of agents.

RV is a root value in PR_1. Whenever a provenance record is added to the provenance chain, RV is updated at PR_1. To calculate RV, $Agent_i$ sends his/her signature to the auditor. The auditor first decrypts the RV and then signs the XOR of aggregated signature and decrypted RV. The updated RV is sent back to $Agent_i$ to append in the provenance record. The first RV is the XOR of Sig_1 (which is not the aggregated one) and IV. The next RV is calculated by the XOR of decrypted RV and Sig_2^*. For

$$\begin{aligned}
PR_1 : RV &= E_{K_{Aud}^{+}}(IV \oplus Sig_1)\\
PR_2 : RV &= E_{K_{Aud}^{+}}\left(Decrypt_{K_{Aud}}(RV) \oplus Sig_2^*\right)\\
&\vdots\\
For\ PR_n : RV &= E_{K_{Aud}^{+}}\left(Decrypt_{K_{Aud}}(RV) \oplus Sig_n^*\right)
\end{aligned}$$

The Key generation and distribution operation is performed off-line. Every user and auditor is assigned a pair of public and private keys along with a symmetric key. The public key of every user/auditor is known publicly, whereas the private key remains confidential. There is a difference between private key and symmetric key in our approach. The symmetric key is used for encryption and decryption, and specifically AES in our scheme; while the private key is used for the RSA digital signature. We assume that the keys and communication channel between participating users are secure (Table 1).

4.1 Secure Provenance Generation

When the first user creates a document, a provenance record will be saved in the associated XML file. From lines 3–10 of Algorithm 1, a complete provenance record is generated. In line 5, the user's identity is encrypted using the symmetric key of user 1. Activity is empty because the owner of the document has just created the document (see line 4). In lines 7 and 8, the user's symmetric key is encrypted using the public key

Table 1 Symbols and their description

Notation	Description
$Agent_i$	ID of $Agent_i$ in plain text
$Agent_i^*$	Encrypted ID of $Agent_i$
$Activity_i$	$Activity_i$ in plain text
$Activity_i^*$	Encrypted $Activity_i$
S_i	Symmetric key of $Agent_i$ in plain text
S_i^*	Encrypted symmetric key of $Agent_i$
Sig_i	Digital signature of $User_i$
Sig_i^*	Aggregated digital signature of $User_i$
IV	Initialization vector
RV	Root value
$\oplus$	XOR
ϕ	Empty/blank
E	Encryption
K_{Aud}	Public key of auditor
K_{Aud}^+	Symmetric key of auditor
$H(Doc_i)$	Hash value of $Document_i$
$Decrypt_{k_{Aud}}$	Decryption process (where K_{Aud} is the public key of auditor)
$Decrypt_{k_i}$	Decryption process (where K_i is the public key of $Agent_i$)

of the auditor, and the hash value of the document is calculated, respectively. In line 9, the signature of the user is created that is formed by encrypting the hash value of the user's identity, activity, symmetric key and hash of document using user $1'$s private key. In line 10, the root value is calculated and encrypted by the auditor that is an XOR of the signature and the IV. The first user then passes the document and provenance to the next user for further processing. The next user performs some operations on the document and generates a second provenance record. In lines 12–14, the identity of second user, activity and symmetric key are encrypted. The hash of the document is generated in line 15, and the signature is XORed with the previous signature in line 16. Finally, in line 17, the auditor decrypts the RV before XORing the decrypted value and signature and encrypting the result. The working of provenance generation algorithm is explained in form of a flow diagram in Fig. 2.

Algorithm 1 Secure Provenance Generation

```
1: Input: $GenerateProvenance();$
2: Output: $PR_i = \{Agent_i^*, Activity_i^*, S_i^*, H(Doc_i), Sig_i^*, RV\}$
3: if ($Agent_i$ = Owner of the $Document_1$) then
4:     $Agent_1^* = ES_1(Agent_1)$
5:     $Activity_1^* = \phi$
6:     $S_1^* = E_{K_{Aud}}(S_1)$
7:     $H(Doc)_1$ = Hash of the created document
8:     $Sig_1 = Sig_1\{H(Agent_1^*, Activity_1^*, S_1^*, H(Doc_1))\}$
9:     $RV = E_{K_{Aud}^+}(Sig_1 \oplus IV)$
10: else
11:     $Agent_i^* = E_{S_i}(Agent_i)$
12:     $Activity_i^* = E_{S_i}(Activity_i)$
13:     $S_i^* = E_{K_{Aud}}(S_i)$
14:     $H(Doc_i)$ = Hash of the modified document
15:     $Sig_i^* = Sig_i\ \{H(Agent_i^*,\ Activity_i^*,\ S_i^*,\ H(Doc_i)\} \oplus \quad Sig_{i-1}^*$
16:     $RV = E_{K_{Aud}^+}(Decrypt_{K_{Aud}}(RV) \oplus Sig_i^*)$
17: end if
```

The provenance generation algorithm is depicted in Fig. 3, where $Agent_A$ is the owner of document Doc_A and PR_1 is generated as: $PR_1 = Agent_A^*,\ Activity_A^*,\ S_A^*,\ H\ (Doc_A),\ Sig_A,\ RV$

$$\begin{aligned} Sig_A &= Sig_A H\left(Agent_A^*,\ Activity_A^*,\ S_A^*, H\left(Doc_A\right)\right) \\ RV\ &= E_K + \left(Sig_A \oplus IV\right) \end{aligned}$$

$Agent_A$ and $Activity_A$ are encrypted using the symmetric key of $Agent_A$. To provide access to the auditor for auditing purpose, the symmetric key of $Agent_A$ is encrypted using K_{Aud} and stored in the field S_A^*. The RV is the XOR of Sig_A and IV signed by auditor K^+. When $Agent_D$ receives the document Aud from $Agent_B$ and $Agent_c$, all fields are calculated in the same manner, with the exception of signature and RV. The signature and RV are as follows:

$$Sig_D^* = Sig_B^* \oplus Sig_D, Sig_C^* \oplus Sig_D$$

here $Agent_D$ has two signatures because $Agent_D$ received the document from two users $Agent_B$ and $Agent_C$ as shown in Fig. 3.

$$RV = E_K^+\left(Decrypt_{K_{Aud}}(RV) \oplus Sig^*\right)$$

4.2 Provenance Chain Verification

During audit, to detect malicious modifications in the document, the hash value of Doc_n is calculated to compare with H (Doc_n) of PR_n (see lines 2 and 3 of Algorithm 2). If this equation holds true, then the document has not been forged; otherwise, its integrity has been compromised. The RV is decrypted in line 4 and the auditor XOR both Sig_n^* and IV in line 5. If the outcome of line 6 is true, this means the provenance records are in their original form. Moreover, the auditor checks the order of provenance

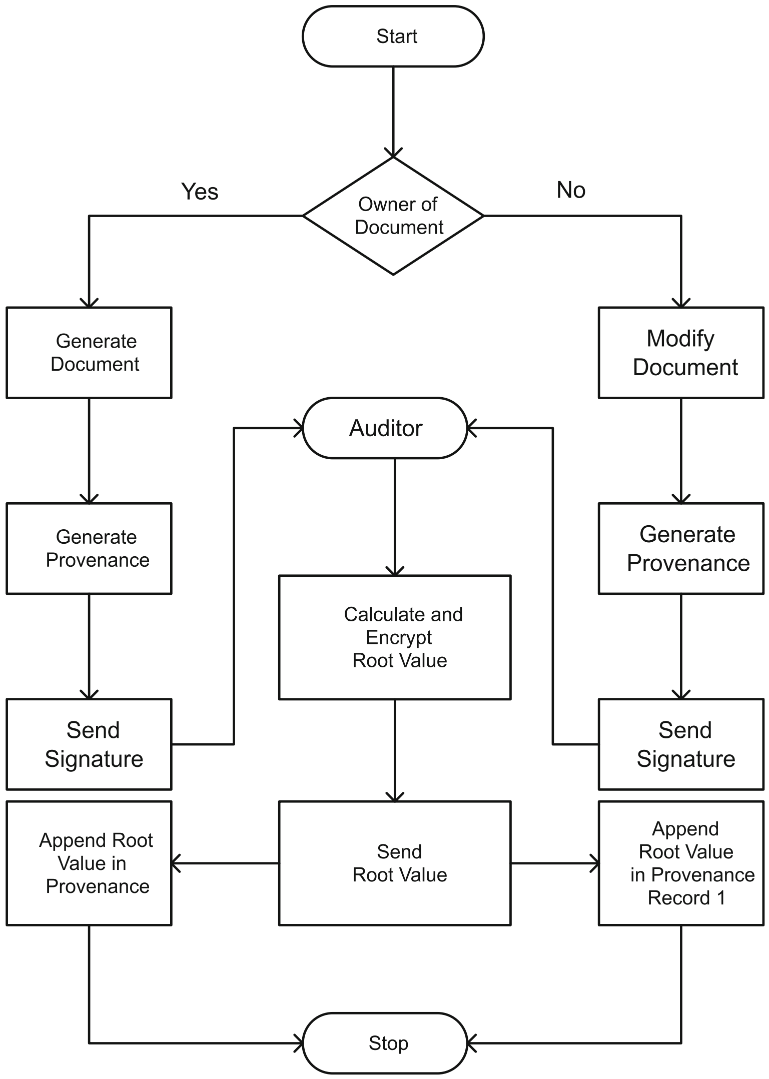


Fig. 2 Provenance generation workflow of our scheme

records and verifies every provenance record to detect the malicious user. In line 8, the auditor performs an operation of *XOR* using the current signature and previous signature. The auditor then decrypts the signature in line 9 and calculates the hash value of all fields of provenance record in line 10. The decrypted signature and the hash value is then compared to check the integrity of individual provenance record in line 12. To verify the first record, the signature is decrypted and hash value is calculated in lines 14 and 15. If the outcome in line 16 holds true, it means the provenance record is not forged. The flow diagram in Fig. 4 show the working of provenance verification algorithm.

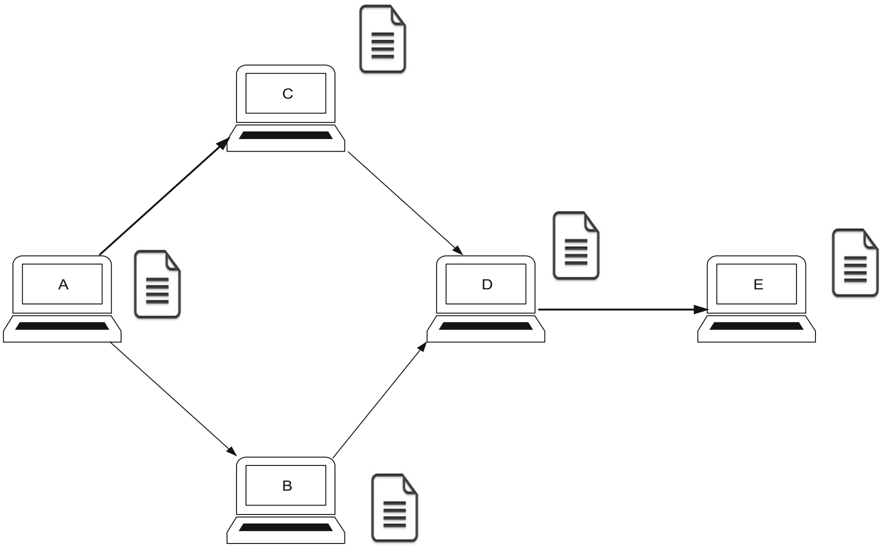


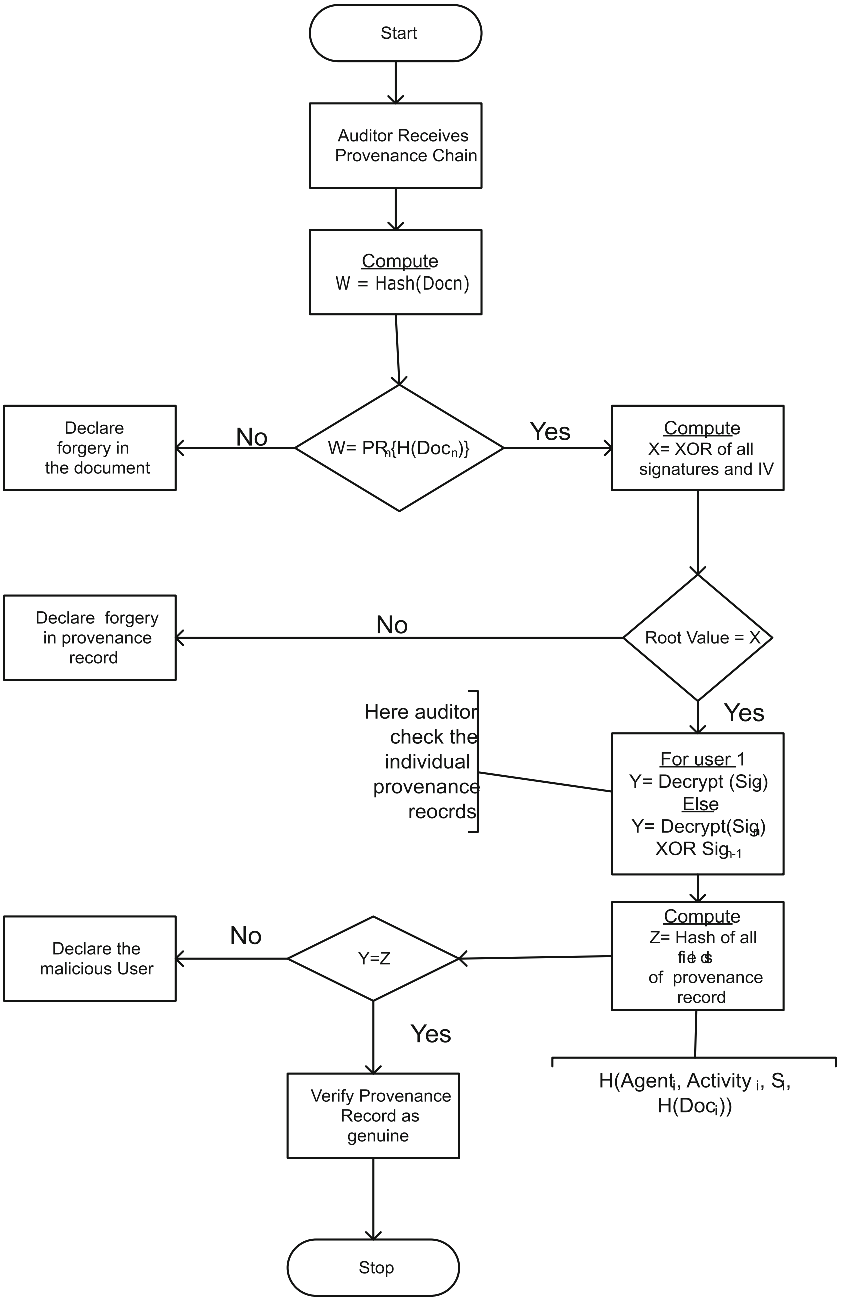
Fig. 3 Document sharing scenario

Algorithm 2 Provenance Chain Verification

```
1: Input: ${PR_1, PR_2, PR_3, ..., PR_{n-1}, PR_n}$
2: Output: $Provenace\ Record/Chain\ forged$ = Yes/No
3: $V \leftarrow h(Doc_n)$
4:     Verify: $V = PR_n\{H(Doc_n)\}$
5:     $Y = Decrypt_{K_{Aud}}(RV)$
6: $Z = Sig_n^* \oplus IV$
7:         Verify : $Z = Y$
8: For all $\{PR_n, PR_{n-1}, PR_{n-2}, ...., PR_2\}$ do
9:     $Sig_i = Sig_i^* \oplus Sig_{i-1}^*$
10:    $W_i = DecryptK_i(Sig_i)$
11:    $X_i = H(Agent_i^*, Activity_i^*, S_i^*, H(Doc_i))$
12:    Verify: $W_i = X_i$
13: End for
14: if $PR_n = 1$ then
15:    $U = Decrypt_{K_1}(Sig_1)$
16:    $V = H(Agent_1^*, Activity_1^*, S_1^*, H(Doc_1))$
17:    Verify: $U = V$
18: End if
```

The Agent and Activity fields in a provenance record are sensitive in nature; thus, to achieve confidentiality, we encrypt these fields using the relevant user' symmetric key. To ensure integrity of the data and provenance record/chain, we use a cryptographic hash function. This allows us to detect any modifications in the data and provenance chain simply by comparing the calculated hash with the hash value obtained by decryption of the signature field. Non-repudiation is achieved using digital signatures, since every user signs whenever an operation on document is performed. The provenance chain is bind with the data, which is available whenever an audit activity is performed.

In lines 2–6 of the provenance verification algorithm, the auditor determines whether the document and provenance chain has been modified. If the comparison



Start
Auditor Receives Provenance Chain
Compute W = Hash(Docn)
W= PR_n{H(Doc_n)}
No
Declare forgery in the document
Yes
Compute X= XOR of all signatures and IV
Root Value = X
No
Declare forgery in provenance record
Yes
Here auditor check the individual provenance reocrds
For user 1 Y= Decrypt (Sig_1) Else Y= Decrypt(Sig_n) XOR Sig_n-1
Compute Z= Hash of all fields of provenance record
H(Agent_i, Activity_i, S_i, H(Doc_i))
Y=Z
No
Declare the malicious User
Yes
Verify Provenance Record as genuine
Stop

Fig. 4 Extended provenance verification workflow for detection of malicious users and provenance forgery

in lines 3 and 6 of Algorithm 2 holds true, then the integrity of the document and provenance chain is assured. Regardless of the outcomes of lines 3 and 6 of Algorithm 2, the auditor continues the auditing process to detect provenance records shuffling attack and malicious user(s). Each individual provenance record is verified by:

$$Sig_i = Sig_i^* \oplus Sig_{i-1}^*$$

$$W_i = Decrypt_{K_i}(Sig_i)$$

Sig_i is the actual signature of the $Agent_i$, which is then decrypted by the auditor using $Agent_i$'s public key. Moreover, the auditor calculates the hash value of all elements of PR_i, with the exception of the field of Sig_i as:

$$X_i = H\left(Agent_i^*,\ Activity_i^*,\ S_i^*, H\left(Doc_i\right)\right).$$

Finally, the auditor compares the decrypted signature of $\mathrm{Agent_i}$ and the hash value of $Agent_i^*,\ Activity_i^*, S_i^*, H(Doc_i)$. If this comparison holds true, then $\mathrm{PR_i}$ has not been modified; otherwise, PR_i may be forged and $Agent_i$ is either an adversary or it has been compromised. The same procedure continues until the first provenance record in the reverse order (PR_n, PR_{n-1}, PR_{n-2}, ..., PR_2, PR_1). For PR_1, there is no need for a XOR operation because the first signature is not the aggregated one. The auditor simply decrypts the signature and calculates the hash of $\left\{Agent_1^*, Activity_1^*,\ S_1^*,\ H\left(Doc_1\right)\right\}$ as:

$$U = Decrypt_{K_1}(Sig_1)\ V = H\left(Agent_1^*,\ Activity_1^*,\ S_1^*,\ H\left(Doc_1\right)\right).$$

The steps from line 1–6 of Algorithm 2 help us in detection of forgeries in a document and provenance records/chain. However, these steps cannot detect the attack of an adversary who shuffles the order of provenance records without forging them. The remaining steps of Algorithm 2 detect the records shuffling attack. Moreover, these steps help us to identify the malicious user(s) and forged provenance record(s).

5 Implementation and Evaluation

Provenance generation and its security result in additional overheads in terms of computational and storage costs. Here, we calculate the time needed for generating in our schemes, as well as three other schemes outlined in Table 2. We implement these schemes in Java (OS: Windows 8.1 Pro, Core i-3 and RAM GB). For the asymmetric encryption and digital signatures, we use 1024-bit RSA. We use 128-bit AES for the symmetric encryption, and SHA-1 to calculate the hash value. The four schemes are implemented on the same machine and the document is modified whenever it is sent to the next agent. We remark that provenance transmission and storage evaluations are beyond the scope of this paper, so we assume a secure transmission among participating agents. We simulate these four schemes 10 times each and report the average time required to generate and verify the provenance.

Table 2 Comparison table

Schemes	Detect ownership history forgery	Record chaining	Support dynamic information sharing?	Detects attacks by consecutive adversaries
[12, 13]	No	Yes: by incremental signature	Yes	No
[14]	No	Yes: by public keys of users	No	No
[15]	Yes	Yes: by maintaining signature of previous and next user	Yes	No
Proposed scheme	Yes	Yes: by aggregated signatures	Yes	Yes

5.1 Comparison with Related Schemes

As shown in Table 2, our scheme provides better protection than those in [12–15]. For example, the outermost record of the Onion scheme [12, 13] cannot be protected because the provenance records are not linked with the next record. In other words, assume that the provenance chain is (PR_1, PR_2, PR_3, PR_4, … PR_n), an adversary can drop certain provenance records PRi to PR_n $(i \geq 1\hat{}i \leq n)$ and place his/her own signature after PR_{i-1}. This selective record dropping cannot be detected during the audit activity. On the other hand, the RV in our scheme detects changes in the provenance chain wherever they are made. Also, an adversary in the PKLC scheme [14] can delete all records of the provenance chain and claims to be the owner of the document. Thus, such a scheme cannot be applied in a dynamic information-sharing network where the next user is not known. In our scheme, the auditor performs *XOR* operation on *RV* and *IV* (and only the auditor has the knowledge of *IV*). Thus, owner history forgery is not possible. The scheme proposed in [15] maintains the signatures of previous and next users, and the signature is a concatenation of U_i, O_i, $H\,(D_i)$ and S_i^*. This is not cost-effective in terms of storage and computation. If adversaries are consecutive, then the auditor cannot detect the forgeries in the provenance chain. For instance, if U_{i+1}, U_{i+2} and U_{i+3} $(i>1)$ are malicious users and U_{i+1} behaves normally and the next user U_{i+2} inserts records between U_{i+3} and him, such a change goes undetected. Also, when the second last user U_{n-1} and last U_n are adversaries and U_{n-1} deletes the user U_n and adds another record by appending the next user to his own provenance record, such attacks cannot be detected as well. Since both PKLC [14] and Mutual Agreement [15] assumed transitive trust, both schemes will not be able to detect attacks launched by consecutive colluding users. Unlike these schemes, our proposed scheme removes the transitive trust dependency by maintaining *RV* signed by the auditor.

5.2 Secure Provenance Generation

We calculate the time required by the four schemes to generate a provenance record, and the results are shown in Fig. 5a, b. Our proposed scheme outperforms the other schemes, in terms of computational cost. PKLC performs well as compared to the Onion and Mutual Agreement schemes. Our scheme has a lower provenance record generation time, i.e. 17% lower than Onion and 11% lower than PKLC. For provenance records chaining, the Mutual Agreement scheme saves the signatures of previous and next user, which results in 34% computational overhead.

The Mutual agreement scheme also requires a longer time to generate the provenance record because it uses 4 hashes, three signatures along with the public key. The overhead of Onion scheme is greater than PKLC because the former uses two hashes, one signature along with previous signatures, and public key while the later uses two hashes, one signature, and public key. The overhead reduction of our schemes is due to the use of two hashes, one signature and a single root value (for whole provenance chain). We also observe that the computational costs of Onion, *PKLC*, and mutual Agreement schemes increase with the increase in the number of provenance records in the chain.

5.3 Provenance Chain Verification

In the provenance verification phase, we calculate the time needed for verification of a provenance chain. The verification time of Mutual Agreement and Onion scheme increases with the increase of the provenance records in a provenance chain. As depicted in Fig. 6a, b, the verification time of our scheme and PKLC is approximately same. However, our scheme has a higher level of security as discussed earlier.

We are considering the first construction of the Onion scheme where the provenance records contain the signature of the previous provenance record only. The time required for Onion scheme to verify a provenance chain will increase exponentially if we consider the spiral construction, where all the signatures of previously participating users are saved in a single provenance record. To verify a provenance chain, the Mutual Agreement scheme takes a long time as compared to other three schemes because it verifies three signatures and two hashes of a single record.

The average overhead reduction in verification time of our scheme is 19% lower than Onion scheme, 11% lower than *PKLC*, and 50% lower than Mutual Agreement scheme as depicted in Fig. 7. The verification time required by our scheme will be reduced up to 50% if we exclude the last attack of our attacker model (shuffling of provenance records). *RV* can be calculated by the XOR of the Sig_n and *IV* to detect the forgeries in provenance records.

Mathematically:

$$\begin{aligned} Calculated \quad RV &= Sig_n^* \oplus IV \\ Verify : Calculated \quad RV &= or \neq RV(RV\ at\ PR1) \end{aligned}$$

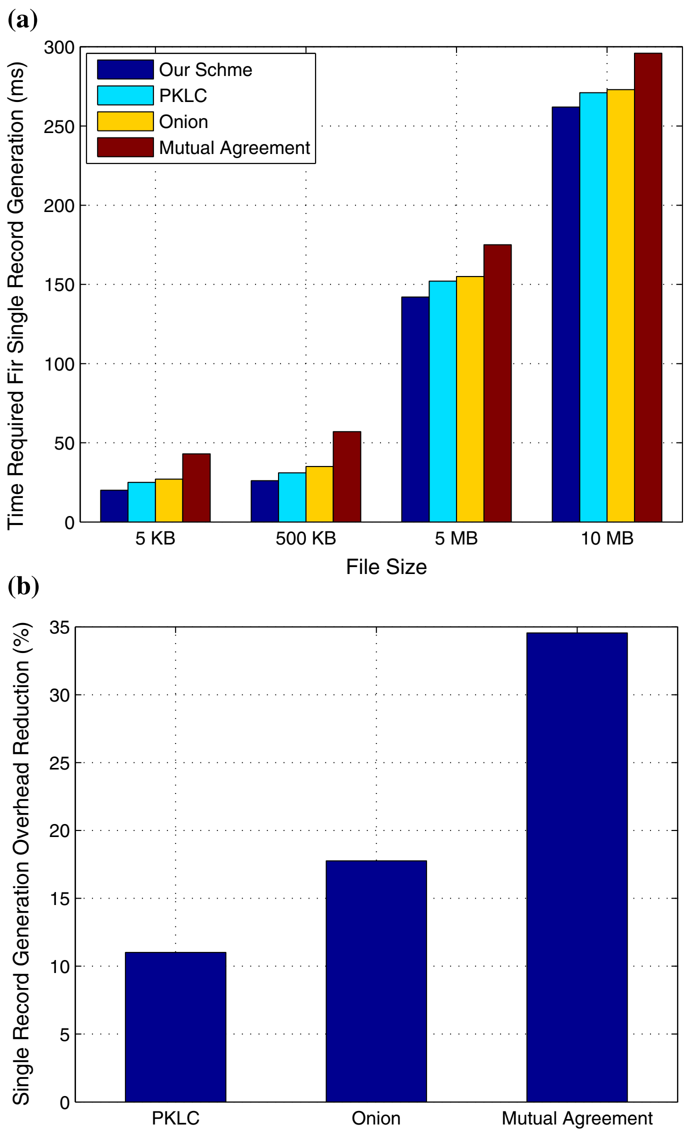
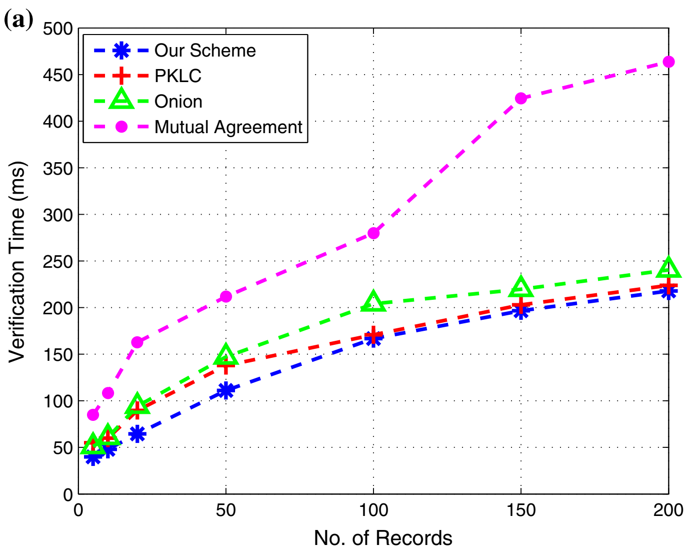


Fig. 5 Provenance record generation time and overhead reduction. **a** A single record generation time and **b** overhead reduction

5.4 Storage Costs

We convert all the fields of a provenance record into strings in order to display in XML file and calculate the number of bytes. PKLC has a lower storage cost as compared to those of Mutual Agreement and Onion schemes. As shown in Fig. 8a, b, our scheme has reduced storage overhead as compared to the other three schemes. Specifically,



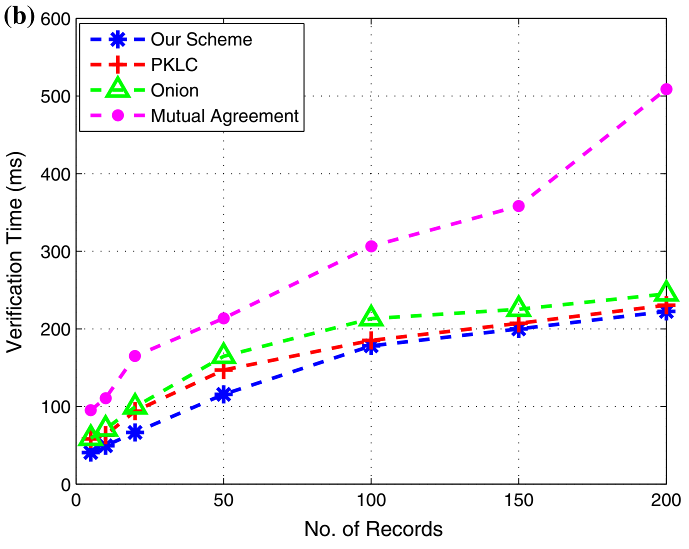


Fig. 6 Provenance verification time. **a** A 5 KB TXT file and **b** a 5 MB PDF file

the proposed scheme has reduced the storage cost of 60, 53 and 27% as compared to [15, 13, 14] respectively. The storage costs in [13, 15] will increase exponentially as the number of provenance records increases. The way our proposed scheme calculates the signatures and root value does not affect the storage cost because of a fix length root value for entire provenance chain.

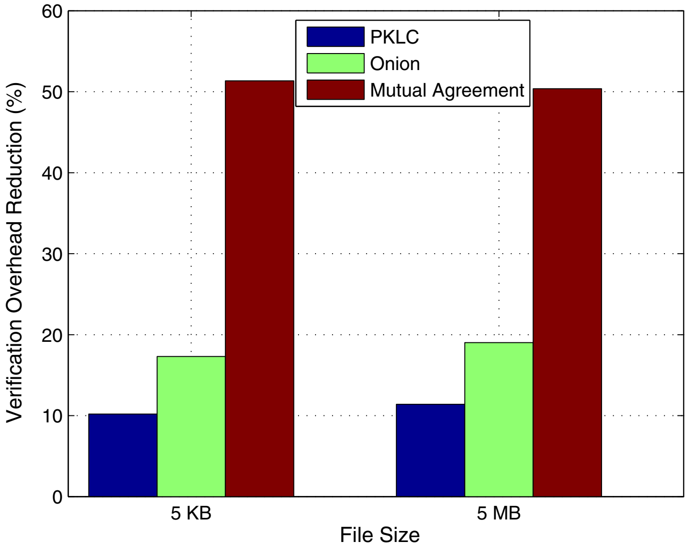


Fig. 7 Reduced verification time in percentage

6 Security Analysis

We will now demonstrate that our provenance scheme achieves the security properties in presence of an adversary as described in Sect. 3.

Claim 1 *A malicious user cannot forge the ownership of any document by exterminating the associated provenance chain undetectably.*

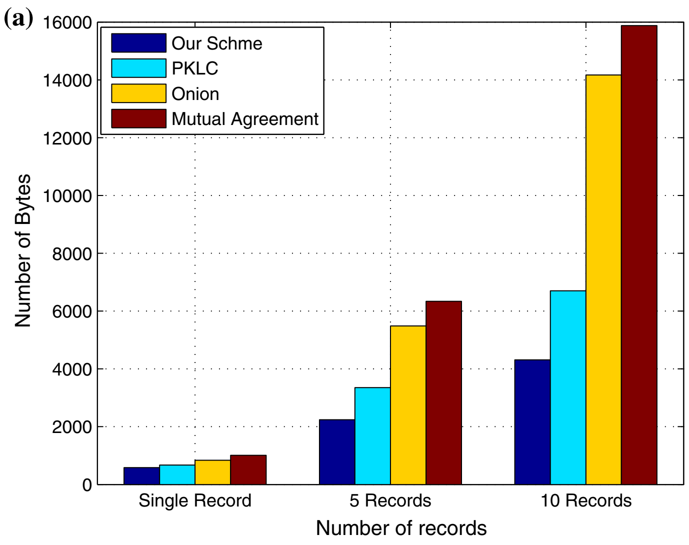
If a user removes the provenance chain of a document and inserts a new provenance record to show his/her ownership, then our scheme detects such an ownership forgery. The RV is generated by the XOR operation of signature and IV that is known to auditor only.

$$RV = E_{K^{+}_{Aud}}(Sig_1 \oplus IV)$$

The IV is oblivious to everyone and therefore, the adversary cannot update the RV. Similarly, the genuine owner cannot impersonate other legitimate users by forging their provenance records.

Claim 2 *Malicious users cannot repudiate their actions or claim that certain provenance records are of another user.*

Whenever a document is created or modified, every user must sign (Sig_i) the document with private key. An attacker modifies the PR_i (provenance record of $Agent_i$) by inserting Sig^{*}_{i-1} of $Agent_{i-1}$ instead of the signature of $Agent_{i-1}$ and forwards to the next user to proclaim that PR_i is of $Agent_{i-1}$. As we know that Sig_{i-1} composes



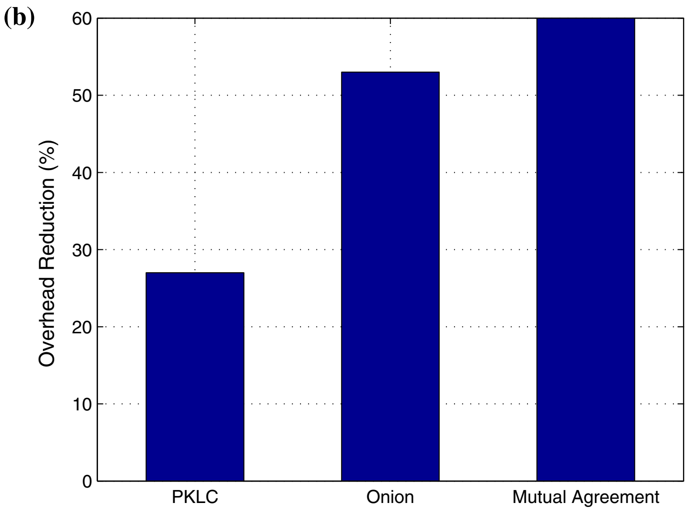


Fig. 8 Storage cost and overhead reduction of the proposed scheme. **a** Storage cost and **b** reduction of storage overhead

a signed hash of $Agent^*_{i-1}$, Activity$^*_{i-1}$, S^*_{i-1}, and *H (Doc*$_{i\text{-}1}$*)*. When the auditor generates the hashes of $Agent^*_i$, Activity$^*_i$, S^*_i, and *H (Doc*$_i$*)*, the hash values will differ. Mathematically,

$$HashVal = Decrypt_{K_{i-1}}\left(Sig^*_{i-1}\right)$$
$$H(R_i) = Hash\left(Agent^*_i,\ Activity^*_i,\ S^*_i,\ H\left(Doc_i\right)\right)$$
$$Therefore,\ HashVal \neq H(R_i).$$

Claim 3 *An adversary acting alone and/or colluding with other users cannot successfully remove the provenance records from the provenance chain being undetected.*

When malicious users remove PR_i and PR_{i+1} from the provenance chain, the auditor can easily detect this attack. The RV calculated by auditor will not be the same as the RV of provenance record PR_1.

Mathematically:

$$RV^* = \left(Sig_n^* \oplus Sig_{n-1}^*, \ldots, \oplus Sig_1 \oplus IV\right)$$

Or

$$RV^* = Sig_n * \oplus IV$$
$$RV^* \neq Decrypt_{K_{Aud}}(RV)$$

Claim 4 *Two or more than two consecutive colluders cannot add provenance records in a provenance chain without being detected.*

Assume that consecutive colluding users $Agent_i$, $Agent_{i+1}$, and $Agent_{i+3}$ add some records in the provenance chain. During the audit, the RV calculated by the auditor by performing XOR operation on all signature fields (Sig_1, Sig_2^*, ..., Sig_n^*) and IV (or $Sig_n^* \oplus IV$) will not be same as the RV at provenance record PR_1 (decrypted RV). Let us assume that the original provenance chain is (PR_1, PR_2, PR_3, ..., PR_5). If some adversaries add provenance records PR_6 and PR_7 anywhere in the provenance chain, then the resultant provenance chain becomes (PR_1, PR_2, PR_3, PR_6, PR_4, PR_5, PR_7), where $CalcRV = Sig_1 \oplus Sig_2^* \oplus Sig_3^* \oplus Sig_6 \oplus Sig_4^* \oplus Sig_5^* \oplus Sig_7 \oplus IV$ or $CalcRV = Sig_7 \oplus IV$

The auditor then compares:

$$CalcRV = or \neq Decrypt_{K_{Aud}}(RV)$$

In this way, the auditor can detect any kind of forgery or addition of provenance records in the provenance chain.

Claim 5 *Malicious user acting alone or colluding with other users cannot successfully perform undetected modifications in the provenance records and signatures.*

Assume that an attacker $Agent_i$ modifies the fields ($Agent_{i-1}^*$, $Activity_{i-1}^*$, S_{i-1}^*, $H(Doc_{i-1})$) of the provenance record PR_{i-1}. During the audit, the hash values of ($Agent_{i-1}^*$, $Activity_{i-1}^*$, S_{i-1}^*, $H(Doc_{i-1})$) will not be equal to the decryption of signature Sig_{i-1} of $Agent_{i-1}$.

Formally,

$$\left\{\left(Agent_{i-1}^*,\ Activity_{i-1}^*,\ S_{i-1}^*,\ H\left(Doc_{i-1}\right)\right)\right\} \neq Decrypt_{K_{i-1}}(Sig_{i-1})$$

If the signature (Sig_{i-1}^{*}) is modified, then the forgery can be detected and Sig_{i-1}^{*} can be retrieved back by:

$$Sig_{i-1}^{*} = Sig_{I}^{*} \oplus Sig_{i} \quad \left(Sig_{I}^{*} = (Sig_{i} \oplus Sig_{i-1}^{*}\right).$$

Claim 6 *Malicious user acting alone or colluding with other users cannot append a provenance chain with another document* Doc_2 *undetected.*

The last provenance record PR_n contains $H\ (Doc_n)$, and the auditor obtains the hash of Doc_2.

$H\ (Doc_2)$, The auditor then verifies: $H\ (Doc_n) \neq H\ (Doc_2)$

If the comparison holds false, it means that the appended provenance chain isnot related to document $Doc2$.

Claim 7 *An adversary acting alone or with a set of adversaries cannot shuffle the order of provenance records without detection.*

If the provenance chain is $(PR_1, PR_2, PR_3, PR_4, PR_5, PR_6)$ and an adversary shuffles the order of records to make the provenance chain as: $(PR_1, PR_3, PR_2, PR_5, PR_4, PR_6)$. During the audit phase, the auditor will detect the forgery when he/she verifies the signatures as:

$$\begin{aligned}
&Sig_6 = Sig_6^{*} \oplus Sig_4^{*} \\
&W_6 = Decrypt_{K_6}(Sig_6) \\
&X_6 = H\left(Agent_i^{*}, Activity_i^{*},\ S_i^{*}, H(Doc_i)\right) \\
&Therefore,\ W_6 \neq X_6.
\end{aligned}$$

7 Conclusion and Future Work

As cloud computing and other distributed networks (e.g. social networks, Internet of Things and cyber-physical systems) become more popular, the need to ensure the provenance of data becomes increasingly pronounced. In this paper, we studied the problem of secure and efficient provenance in a distributed environment. Specifically, we presented a scheme that offers confidentiality; integrity, availability and non-repudiation, in the sense that attacks on a provenance chain by multiple colluding users and provenance records shuffling attacks will be detected (as demonstrated in our security analysis). We also demonstrated that the scheme outperforms other similar schemes (i.e. [12–15]), in terms of performance. However, one limitation of the proposed scheme is that it requires a trusted auditor during provenance generation phase. We believe that such trusted auditor is important to carry out the necessary cryptographic operations to detect forgeries. Thus, in the future, we will seek to remove the trusted auditor from the design without compromising on the security and performance of the scheme.

References

1. Moreau, L., Paolo, M.: PROV-DM: the PROV data model, W3C recommendation (2013)
2. Buneman, P., Khanna, S., Tan, W.C.: On propagation of deletions and annotations through views. In: Proceedings of 21st ACM SIGMOD-SIGACT-SIGART Symposium on Principles of Database Systems, pp. 150–158. ACM (2002)
3. Buneman, P., Khanna, S., Tan, W.C.: Why and where provenance: a characterization of data provenance. In: Database Theory ICDT 2001, pp. 316–330. Springer (2001)
4. Clifford, B., Foster, I., Voeckler, J.S., Wilde, M., Zhao, Y.: Tracking provenance in a virtual data grid. Concurr. Comput. Pract. Exp. **20**(5), 565–575 (2008)
5. Davidson, S.B., Boulakia, S.C., Eyal, A., Ludscher, B., McPhillips, T.M., Bowers, S., Anand, M.K., Freire, J.: Provenance in scientific workflow systems. IEEE Data Eng. Bull. **30**(4), 4450 (2007)
6. Davidson, S.B., Freire, J.: Provenance and scientific work-flows: challenges and opportunities. In: Proceedings of the 2008 ACM SIG-MOD International Conference on Management of Data, p. 13451350. ACM (2008)
7. Groth, P., Gil, Y., Cheney, J., Miles, S.: Requirements for provenance on the web. Int. J. Digit. Curation **7**(1), 3956 (2012)
8. Hartig, O.: Provenance information in the web of data. In: LDOW (2009)
9. Muniswamy-Reddy, K.K., Holland, D.A., Braun, U., Seltzer, M.I.: Provenance-aware storage systems. In: USENIX Annual Technical Conference, General Track, p. 4356 (2006)
10. Sar, C., Cao, P.: Lineage file system, p. 411414. http://crypto.stanford.edu/cao/lineage.html (2005). Accessed 2 Aug 2016
11. Gehani, A., Kim, M., Zhang, J.: Steps toward managing lineage metadata in grid clusters. In: First Workshop on Theory and Practice of Provenance, p. 7. USENIX Association (2009)
12. Hasan, R., Sion, R., Winslett, M.: The case of the fake Picasso: preventing history forgery with secure provenance. In: FAST, vol. 9, pp. 1–14 (2009)
13. Hasan, R., Sion, R., Winslett, M.: Preventing history forgery with secure provenance. ACM Trans. Storage (TOS) **5**(4), 12 (2009)
14. Wang, X., Zeng, K., Govindan, K., Mohapatra, P.: Chaining for securing data provenance in distributed information networks. In: MILCOM 2012, pp. 1–6 (2012)
15. Rangwala, M., Liang, Z., Peng, W., Zou, X., Li, F.: A mutual agreement signature scheme for secure data provenance. Environments **13**, 14 (2016)
16. Luiz, M.R., Gadelha Jr, Marta, M.: Kairos: an architecture for securing authorship and temporal information of provenance data in grid- enabled workflow management systems. In: IEEE Fourth International Conference on eScience, 2008. eScience08, p. 597602. IEEE (2008)
17. Khan, R., Zawoad, S., Md Haque, M., Hasan, R.: OTIT: towards secure provenance modeling for location proofs. In: Proceedings of the 9th ACM Symposium on Information, Computer and Communications Security, pp. 87–98. ACM (2014)
18. Hassan, R., Burns, R.: Where have you been? Secure location provenance for mobile devices. arXiv preprint arXiv:1107.1821 (2011)
19. Hasan, R., Khan, R., Zawoad, S., Md Haque, M.: WORAL: a witness oriented secure location provenance framework for mobile devices. IEEE Trans. Emerg. Top. Comput. (2015)
20. Asghar, M.R., Ion, M., Russello, G., Crispo, B.: Securing data provenance in the cloud. In: Lomet, D.B. (ed.) Open Problems in Network Security, p. 145160. Springer (2012)
21. Lu, R., Lin, X., Liang, X., Sherman Shen, X.: 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, pp. 282–292. ACM (2010)
22. Saad, M.I.M., Jalil, K.A., Manaf, M.: Achieving trust in cloud computing using secure data provenance. In: IEEE Conference on Open Systems (ICOS). October 26–28 2014, Subang Malaysia (2014)
23. Alharbi, K., Lin, X.: PDP: a privacy-preserving data provenance scheme. https://doi.org/10.1109/icdcsw (2012)
24. Zawoad, S., Hasan, R.: SECAP: towards securing application provenance in the cloud. In: 2016 IEEE 9th International Conference on Cloud Computing (CLOUD), pp. 900–903. IEEE (2016)
25. Liang, X., Shetty, S., Tosh, D., Kamhoua, C., Kwiat, K., Njilla, L.: ProvChain: a blockchain-based data provenance architecture in cloud environment with enhanced privacy and availability. In: International Symposium on Cluster, Cloud and Grid Computing, IEEE/ACM (2017). https://doi.org/10.1109/CCGRID.2017.8

26. Xu, S., Ni, Q., Bertino, E., Sandhu, R.: A characterization of the problem of secure provenance management. In: IEEE International Conference on Intelligence and Security Informatics, 2009. ISI' 09, pp. 310–314. IEEE (2009)
27. Sultana, S., Shehab, M., Bertino, E.: Secure provenance transmission for streaming data. IEEE Trans. Knowl. Data Eng. **25**(8), 1364–1378 (2011)
28. Sultana, S., Ghinita, G., Bertino, E., Shehab, M.: A lightweight secure provenance scheme for wireless sensor networks. In: IEEE 18th International Conference on Parallel and Distributed Systems (ICPADS), 2012, p. 101108. IEEE (2012)
29. Sultana, S., Ghinita, G., Bertino, E., Shehab, M.: A lightweight secure scheme for detecting provenance forgery and packet drop attacks in wireless sensor networks. IEEE Trans Dependable Secure Comput (2014)
30. Hussain, S.R., Wang, C., Sultana, S., Bertino, E.: Secure data provenance compression using arithmetic coding in wireless sensor networks. In: Performance Computing and Communications Conference (IPCCC), 2014 IEEE International, pp. 1–10. IEEE (2014)
31. Wang, C., Bertino, E.: Sensor network provenance compression using dynamic bayesian networks. ACM Trans. Sens. Netw (TOSN) **13**(1), 5 (2017)
32. Wang, C., Hussain, S.R., Bertino, E.: Dictionary based secure provenance compression for wireless sensor networks. IEEE Trans. Parallel Distrib. Syst **27**(2), 405–418 (2016)
33. Syalim, A., Nishide, T., Sakurai, K.: Preserving integrity and confidentiality of a directed acyclic graph model of provenance. In: Foresti, S., Jajodia, S. (eds.) Data and Applications Security and Privacy XXIV, vol. 6166, pp. 311–318. Springer, Berlin (2010)
34. Schler, M., Schulze, S., Merkel, R., Saake, G., Dittmann, J.: Reliable provenance information for multimedia data using invertible fragile watermarks. In: British National Conference on Databases, pp. 3–17. Springer, Berlin (2011)
35. Aman, M.N., Chua, K.C., Sikdar, B.: Secure data provenance for the internet of things. In: Proceedings of the 3rd ACM International Workshop on IoT Privacy, Trust, and Security, pp. 11–14. ACM (2017)
36. Zafar, F., Khan, A., Suhail, S., Ahmed, I., Hameed, K., Khan, H.M., Jabeen, F., Anjum, A.: Trustworthy data: a survey, taxonomy and future trends of secure provenance schemes. J. Netw. Comput. Appl. **94**, 50–68 (2017)
37. Jamil, F., Khan, A., Anjum, A., Ahmed, M., Jabeen, F., Javaid, N.: Secure provenance using an authenticated data structure approach. Comput. Secur. **73**, 34–56 (2018)