Chaining for Securing Data Provenance in Distributed Information Networks

Xinlei (Oscar) Wang*, Kai Zeng†, Kannan Govindan*, Prasant Mohapatra*

*Department of Computer Science
University of California, Davis, CA 95616
{xlwang, kgovindan, pmohapatra}@ucdavis.edu
†Department of Computer and Information Science
University of Michigan, Dearborn, Michigan, USA 48128
kzeng@umich.edu

***Abstract*—Entities in an information communication network may use various types of collaborative networking for sharing information such as documents, sensing reports, datasets, etc. The derivation history (i.e., the provenance) of the information plays a very important role in such a networking environment. For example, provenance can be used for information trustworthiness assessment, copyright clearance, data reconciliation, and data replication. While substantial research efforts have focused on these usages of provenance, very limited work has focused on the security issues of the provenance, which is the prerequisite of any provenance-based information analysis systems. In this paper, we explore the security properties of provenance meta-data compared to other general user data in a distributed network environment. We introduce a "chain-structure" provenance scheme to provide security assurance for the provenance meta-data in three dimensions - confidentiality, integrity and availability. Our scheme outperforms the previously proposed "onion-structure" provenance security scheme in terms of the flexibility, protection capability as well as computational overhead.**

***Index Terms*—Distributed Information Networks, Data Provenance, Provenance Security**

I. Introduction

Information networks are networks that use the networking technologies such as the Internet and wireless communication for distributing and sharing information among different information processing entities to enhance knowledge, business or social aims. Essentially, the information processing entities can be individuals or interest groups within and between human institutions such as companies, research organizations, governmental organizations and communities. The basic unit of the dynamics forming information networks can be seen as an "information transaction" involving the exchange of an information item between two or more information processing entities. One of the main factors that influences the quality of information (QoI) is the *data provenance*, which describes the derivation history of an information item. Provenance involves tracking the origin of the information and subsequent transformations performed on it [1]. In order to exclude malicious information sources or prioritize the information according to individuals' preferences, the QoI is commonly assessed based on provenance. In addition to QoI assessment, some other important usages of provenance include copyright clearance, data reconciliation and data replication.

Several research efforts have focused on the areas of tracking, querying and analysis of provenance. However, very limited work has been done on securing provenance data, a vital step in achieving trust and ultimately usability of provenance as a concept. Consider the incentives of protecting provenance in the following military command and control networks scenario: Information such as unit status, target information, intelligence reports, operational plans and logistics activities from multiple sources needs to be distributed to and processed by relevant users throughout the battle space. Provenance needs to be recorded to facilitate the analysis of the information credibility for better decision-making. A compromised node may try to deceive the decision makers by forging or tampering the provenance records. We need a mechanism to detect it when this happens. Furthermore, user identities or certain processing details recorded in the provenance may have to be kept confidential when the information flows from a higher level of command to a lower level of command.

Assuring provenance security is relatively easy if there is a central application server when the information network is online, which can serve as a central administration to manage the provenance data. However, it will be more challenging if there is no online central server available or the server cannot be trusted. In such distributed information networks, the provenance security management work is distributed to the individual users. Although a few recent efforts have tried to define the provenance security problem [2]–[4], they did not consider the challenges posed by the distributed nature of certain information networks. Our first objective of this work is to discuss these unique challenges and requirements. Based on our discussion, we propose a novel provenance framework which provides security assurance and can be applied directly to provenance-based network trust systems like [5], [6]. A related provenance scheme was proposed by Hasan et al. [7]. We refer to their solution as the *Onion* scheme due to its onion-like signature structure for protecting the provenance integrity. The Onion scheme has certain weaknesses when applied to a distributed information network (see detailed discussion in Section II). In contrast to the Onion scheme, our solution is "linked chain" structured which solves their problems and also reduces the overhead significantly especially for provenance verification. To summarize, **our contributions** include:

1) The unique challenges and requirements for securing provenance in a distributed network are discussed and a concrete notion of provenance security is given.
2) A Public-key Linked Chain framework is proposed to provide assurance for the integrity, confidentiality and availability of provenance data. Our scheme provides better protection capability than the Onion scheme and solves some of its implementation difficulties.
3) An analysis is done to show that our scheme fulfills each of the security requirements.
4) Analytical and empirical evaluations are given on the provenance overheads of our framework. Experimental results show that our approach reduces significant computational overhead comparing to the Onion scheme.

The rest of the paper is organized as follows. We highlight the related work of securing provenance in Section II. In Section III, we give a discussion on the challenges in securing provenance in a distributed information network. The threat model will also be detailed in this section. We then present our Public-key Linked Chain (PKLC) provenance framework in Section IV and show that our approach can actually meet the security requirements in Section V. The empirical performance evaluations are presented in Section VI. Finally, Section VII concludes the paper and talks about our future work.

II. Related Work

Hasan et al. [2] were the first to attempt to define the secure provenance problem and argue that it is of vital importance in numerous applications. They were emphasizing the protection of the contents of provenance records. Braun et al. [3] gave a preliminary discussion about how provenance data is different from other data from the security perspective. Kairos [8] is an architecture to allow the secure verification of data authorship and temporal information in provenance records. It uses digital signatures and cryptographic time-stamps, which requires straightforward implementation in provenance-aware workflow systems. A secure provenance scheme based on the bilinear pairing technique is proposed in [9] to provide trusted evidences for data forensics in cloud computing.

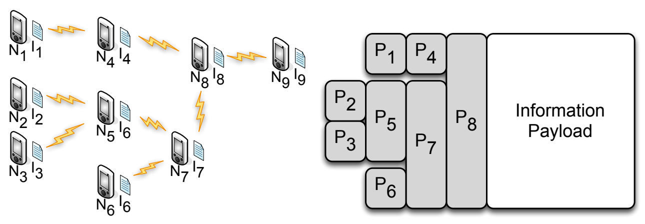
None of these efforts can be applied directly to dynamic information sharing in a distributed network. The most closely related solution we could find that applies to the distributed networks is the Onion scheme [7]. Their solution composes of encryption for sensitive provenance record fields and an incremental signature mechanism that is of onion-like structure (i.e., every provenance record's signature encloses the previous provenance record's signature) for securing the integrity of the chain as a whole. The Onion scheme has certain weaknesses. First of all, the outermost layers of the "onion" signature cannot be protected. When an insider attacker receives an information item with a provenance chain, he/she can extract a part of the provenance chain from the beginning and then wrap it with his/her own signature checksum, with one or more out-most layers of the original "onion" peeled off. Secondly, their scheme requires every auditor to be aware of the NodeID/public key correspondence in order to verify the integrity of the provenance records. However, this may cause privacy issues, i.e., no nodes can really hide their identity since they must put their public key in the records. Also, it is normal that some nodes' keys get revoked/refreshed or some nodes leave the network, it will become hard to verify their previous sent provenance records especially for a long-history provenance. Our approach focuses on solving these problems. Additionally, experimental results show that our approach reduce the provenance verification overhead significantly.

III. Provenance and Its Security

A. Provenance Security Assurance

Each information item consists of meta-data and payload. The meta-data contains the provenance of the information item. Each node needs to generate a provenance record P and the provenance is finally represented as a chain which is a time-ordered sequence of provenance records $P_1|\cdots|P_n$. Notice that although we use the term "provenance chain" to denote the entire provenance meta-data, it is not necessarily a single chain. When a node takes multiple information items as input, the provenance chain will become a tree. Figure 1 shows a scenario of information flow in a distributed network and the resulting information item with its provenance meta-data.

Provenance has characteristics that differentiate it from data typically considered by secure systems. As discussed by Braun et. al. [3]. First, provenance differs from traditional data in that it forms a directed acyclic graph (DAG) that captures information flow from inputs to outputs. Also, the sensitivity of provenance and the information payload it describes may be different. In addition to the above characteristics of the provenance, the distributed nature of the information networks poses more challenges on providing security for provenance.



(a) A scenario of information flow (b) Information item structure received by N_9

Fig. 1. A scenario of information flow and the structure of the final information item

1) Nodes must have access to read the preceding provenance records, at least to certain extent, in order to verify or use the provenance. At the same time, each node should also be able to write its own provenance. This requires different access control on the same provenance data.
2) It is relatively easy to have each node protect their own records, but without a central trusted administration, it is hard to protect the relationship among them.
3) Different security groups may exist in a network. When a piece of information goes through different security groups, differentiated confidentiality control may be needed for the same provenance meta-data.
4) A user from a lower security group may need to verify the integrity of a provenance chain which comes from a higher security group without accessing the content.

We extend and apply the three dimensions of general data security to provenance security as follows:

Confidentiality and Privacy: A provenance record may contain information that is of a confidential nature in two ways: (i) information about the tasks performed may be secret; (ii) the ownership history might contain sensitive information that should not be revealed to unauthorized parties.

Integrity: The integrity of provenance is three-fold:

1) **Data Integrity**: The information contained in each individual provenance record is not tampered with.
2) **Origin Integrity**: The origin of each individual provenance record is not forged. A node cannot deny its ownership of the provenance records created by it.
3) **Chain Integrity**: The order of the owners on the provenance chain is not modified.

Availability: We need to ensure that no nodes in the network can selectively drop a part of the provenance chain without being detected.

B. Threat Model

We will consider a malicious insider or outsider adversary with the goals of:

1) gaining confidential information from provenance records about the actions performed on or the ownership history of the information item.
2) using fake key-pairs or stolen key-pairs to make their own provenance records un-verifiable.
3) altering existing records or adding forged information to the existing provenance chain (this might involve tampering with other nodes' records, changing the sequence or adding forged records).
4) selectively removing a certain part of the preceding provenance chain.

We assume every user has a unique public/private key pair and the information receivers' public keys are known to the senders before the data transmission. We assume a Trusted

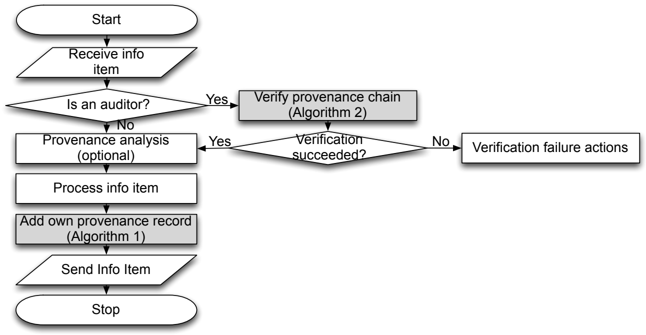

Start
Receive info item
Is an auditor?
Yes
Verify provenance chain (Algorithm 2)
No
Provenance analysis (optional)
Yes
Verification succeeded?
No
Verification failure actions
Process info item
Add own provenance record (Algorithm 1)
Send Info Item
Stop

Fig. 2. Flowchart of an intermediate node's actions with an information item

Central Authority does the user and group registration, key assignment and key management work offline and a scheme like [10] is used for group key distribution.

IV. Secure Provenance Scheme

A. Public-key Linked Chain

Network Framework: In this work, as we are considering distributed information networks, the online users communicate with each other directly with no central server. The nodes can be classified into different roles or groups that have different access rights. Generally, every node in the network is capable of information generation, processing and dissemination. Among all the nodes, there are *auditors* that are responsible for verifying the integrity and availability of the provenance chain. The number of auditors in the network is application specific. It is possible to have all the nodes in the network as auditors. Figure 2 gives an illustration of an intermediate node's actions with a received information item. Our work is focusing on the two parts of "*add own provenance record*" and "*verify provenance chain*". The provenance record generation steps are described in Algorithm 1 and the provenance verification process is described in Algorithm 2.

Alg.1. A node's provenance record generation steps:

```
1: $Node_i \Rightarrow NodeInfo_i|ProcessInfo_i|OptInfo_i$
   ($Node_i$ produces provenance information fields)
2: $Node_i \Rightarrow k_s$
   ($Node_i$ generates a symmetric key)
3: $PInfo_i = Encrypt^{k_s}[NodeInfo_i|ProcessInfo_i|OptInfo_i]$
   (Encrypt sensitive parts of the provenance information fields)
4: For all $Group_m \in AccessibleGroups$
   4.1: $Skey_i = Skey_i \cup \{Encrypt^{K^+_m}[k_s]\}$
5: For all $Node_n \in AccessibleUsers$ And $Node_n \notin AccessibleGroups$
   5.1: $Skey_i = Skey_i \cup \{Encrypt^{K^+_n}[k_s]\}$
6: $PLHash_i = Hash[I_i]$
7: $PubKey_i = \{K^+_{i+1}\}$
8: If $i = 2$, $PubKey_i = PubKey_i \cup \{K^+_1\}$
9: $PInfo_i = PInfo_i|PLHash_i|Skey_i|PubKey_i$
10: $SIG_i = Encrypt^{K^-_i}[Hash[PInfo_i]]$
11: $P_i = PInfo_i|SIG_i$
```

3-Level Provenance Structure: We propose a secure provenance structure composed of three levels as shown in Figure 3. The first level is the *information level*, which is a DAG of provenance records. Each provenance record is generated by one network node and thus is the *node level* provenance information. A node uses one or more information items to produce a new information item. One or more processes may be invoked by the node and then applied on the input information item(s). Therefore, each node level provenance record may contain one or more *process level* details. That is, each provenance record P_i summarizes a sequence of actions taken by a user. Figure 3 illustrates our secure provenance structure and Algorithm 1 shows the detailed steps for a node to generate its own provenance record. Each of the provenance fields shown in the provenance structure is explained below.

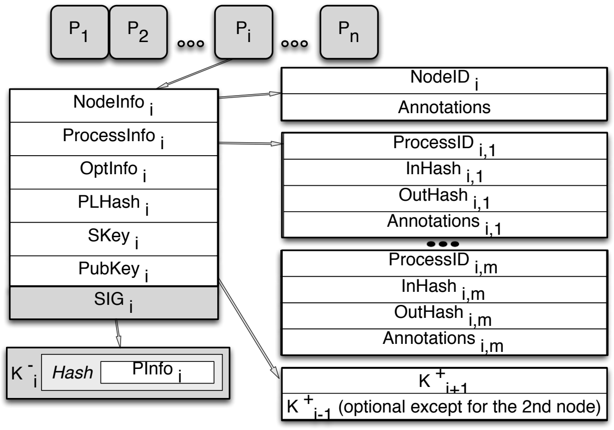


Fig. 3. An illustration of the 3-level provenance structure

NodeInfo is a field which contains encrypted or plaintext information of the processing node. The identifier of the node $NodeID$ must be included. Other semantic description of the node may also be included as property-value annotations.

ProcessInfo contains encrypted or plaintext representation of the processes performed by the node. This field consists of one or more sub-fields, which in turn has the information of a process. Each process sub-field must have the identifier of the corresponding process and the references (hashes) of the input and output information items. Other contextual information about the process can be included as property-value annotations.

OptInfo contains other optional information the node wants to put in the provenance record. It could be the timestamp of the information being sent out or other application specific information. Again, this can be either an encrypted or a plaintext field.

PLHash contains the hash of the final information payload produced by the node.

SKey contains the secret symmetric key K_s that auditors can use to access the encrypted fields or subfields. Multiple copies of K_s are enclosed, with each encrypted by the public key of one intended auditor individual or group.

PubKey is a crucial field used to link each provenance record together and ease the verification process. For every node, this field must contain the plaintext public key K^+_{i+1} of the recipient node. This represents the directional edge in the provenance graph. For the information originators, since they do not have preceding provenance records, they have to put their own public key in the field $PubKey_1$, too. For the second node in the provenance chain, it has to put public keys of both the originator K^+_1 and the next recipient node K^+_3. For the rest of the nodes, it is optional for them to put their preceding node's public key.

SIG contains a digital signature of the node to ensure the integrity of the provenance record and the chain as a whole.

As shown in Figure 3, to create the signature, a node i first generates a hash for the entire $PInfo_i$ (i.e., all above fields) and then use its private key K_i^- to sign it.

Confidentiality Assurance: Certain information in the provenance record may be sensitive, such as the identity of the processing node, the information about a proprietary algorithm used, etc. We desire to keep these fields or subfields to be accessible only by some trusted auditor individuals or groups. If all auditors can be trusted, then providing confidentiality for these sensitive fields would be straightforward. We could just encrypt all of them by a single public key, and give the private key to all the auditors. If a user only trusts auditors in a certain security group or certain specific auditors, we could make several copies of the sensitive fields, encrypt each copy with the public key of a different trusted auditor or security group, and include all of them in the provenance record. However, this wastes lots of space and the asymmetric key encryption/decryption is not efficient. We adopt the similar broadcast encryption scheme as in [7] to selectively regulate the access for different auditor individuals or groups. Instead of encrypting the sensitive fields of the record with the individual or group public keys, we encrypt them once with a single symmetric key k_s generated by the node, make multiple copies of the symmetric key, and then encrypt each copy with the public key of a trusted auditor or security group. In this scheme, the new provenance record contains the encrypted sensitive fields plus several versions of the encrypted secret key, stored in the $SKey$ field. An auditor can subsequently read the record and extract the secret key using its own private key or the group private key.

Alg.2. An auditor's verification steps:

```
1: $Auditor \Leftarrow P_1|\cdots|P_n$
   (Auditor gets the provenance chain)
2: Verify: $PubKey_n/K_{n+1}^+ = K_{Auditor}^+$
3: $Auditor \Rightarrow Hash[I_n]$
   (Auditor produces the hash of $I_n$)
4: Verify: $Hash[I_n] = PLHash_n$
5: For all $P_i \in P_2|\cdots|P_n$
   5.1: Verify:
        $PLHash_{i-1} = ProsessInfo_i/InHash_{i,1}$
6: For all $P_i \in P_1|\cdots|P_n$
   6.1: $Auditor \Rightarrow Hash[PInfo_i]$
        (Auditor produces the hash of $PInfo_i$)
   6.2: If $P_i = P_1$ $Auditor \Leftarrow PubKey_2/K_1^+$
        Else $Auditor \Leftarrow PubKey_{i-1}/K_i^+$
        (Auditor gets the public key $K_i^+$)
   6.3: $Auditor \Rightarrow Decrypt^{K_i^+}[SIG_i]$
        (Auditor decrypts the $SIG_i$ with $K_i^+$)
   6.4: Verify: $Decrypt^{K_i^+}[SIG_i] = Hash[PInfo_i]$
```

Integrity and Availability Assurance: An auditor must be able to detect whether any attackers have tampered, removed or inserted records from the chain and whether a chain has been switched from one information item to another. This is achieved through the hashes and the SIG fields in the provenance chain. Algorithm 2 is a step-by-step explanation of how an auditor verifies a provenance chain. For simplicity and clarity, Algorithm 2 only shows the verification process for single provenance chains (i.e., every node takes one input information item for one output information item). In the case when a certain node on the provenance graph aggregated multiple input information items, the above verification process also applies with minor modifications. First, Step 5.1 requires multiple comparisons to make sure every input information item's hash value can be found in any of the fusion node's $ProcessInfo/InHash$ fields. In addition, each of the inputs' $PubKey$ fields has a copy of the fusion node's public key. The auditor has to compare all these keys and make sure they are consistent.

B. Advantages of the Public-key Linked Chain

We can see our provenance records are interconnected together as a "linked chain" by the $PubKey$ fields, in that every provenance record's $PubKey$ contains the public key of the next processing node. Hence, we name our solution as the *Public-key Linked Chain* (PKLC) scheme. It is simple to be implemented and has many advantages.

First of all, the PKLC scheme provides better protection against selective provenance dropping than the Onion scheme. The Onion scheme cannot protect its outermost layers since each layer has no reference of who the next user is. For instance, when an adversary received an information item with a provenance $P_1|\ldots|P_n$, he/she can remove the provenance records $P_i|\ldots|P_n$ where $1 < i \leqslant n$, wrap the signature of P_{i-1} in his/her own signature and then append his/her provenance record after P_{i-1}. Future auditors are unable to detect this malicious dropping. Our PKLC scheme solves this problem because any selective dropping will break the chain and the next auditor can easily detect it (Proposition 5).

More importantly, by using our PKLC scheme, the auditors can perform their provenance verification without knowing the identity-key correspondence. In a distributed information network, users may not know each other. The verification in the Onion scheme requires the identity-key correspondence must be obtained from a trusted external source. This violates the confidentiality and may not be feasible especially when the provenance chain has a long-history. It is possible that the public keys of certain nodes in the provenance chain have already been revoked/refreshed, or certain nodes already left the network. In these cases, the original public keys may become impossible to be obtained by the auditors and thus makes the verification very hard. Our PKLC scheme enables every auditor to easily obtain an authenticated public key of every node on the provenance chain and then perform the verification without knowing the real identity of the node. This improves the confidentiality assurance and makes the verification process easy and trustworthy even when the key pairs are dynamically changing in the network. In addition, any attacks that try to use fake key-pairs without collusion will also be detected.

V. Analysis of Security Assurance

In this section, we will analyze our scheme and prove that an adversary cannot achieve his/her goals outlined in Section III-B without being detected by the subsequent auditor.

Proposition 1. *Users cannot repudiate their own provenance records even with a fake or another user's private key.*

Every node has to use its private key to sign the SIG field, which serves as a digital signature that no other nodes can forge. Every node has the previous node (or the following node in case the node is an source) providing its public key as an evidence in the $PubKey$ field for verifying the signature. If adversary uses an invalid private key or even a stolen valid private key to repudiate his/her own records, the next auditor will be able to detect it at the verification step 6.4.

Proposition 2. *An adversary cannot claim that a valid provenance chain belongs to a different information item without being detected by the next auditor.*

In a valid provenance chain, $PLHash$ of each provenance record contains a hash of the corresponding output information payload, which is then authenticated using the SIG field. Suppose an adversary extracts the provenance meta-data from its information item I_i and appends it to another information item I_j and then sends I_j out. If the subsequent node is an auditor, he/she can detect that $Hash[I_j]$ does not match with the $PLHash_i$ in P_j at the verification step 4. Otherwise if I_j goes through a sequence of non-auditors and then reaches an auditor, the auditor can detect $PLHash_i$ in P_j does not match with value stored in $ProcessInfo_{j+1}/InHash_{j+1,1}$, which will fail the verification step 5.1.

Proposition 3. *An adversary cannot tamper other nodes' provenance records in the preceding provenance chain without being detected by the next auditor.*

All the $PInfo$ fields in a provenance record is hashed and signed by the corresponding node in SIG. If any fields is tampered by another adversary, the $Hash[PInfo]$ will fail to match with $Decrypt[K_i^+]$ at the verification step 6.4.

Proposition 4. *An adversary cannot add fake records at the beginning or in the middle of the preceding provenance chain without being detected by the next auditor.*

First, let us assume an adversary has inserted a provenance record P_i (either its own record or another node's record) at the beginning of the provenance chain so that the chain becomes $P_i|P_1|\cdots|P_n$. When the auditor verifies the records, he/she will not be able to get K_i^+ from $PubKey_1$ at the verification step 6.2 and thus not be able to successfully pass the verification step 6.4. This indicates P_i is not the original source of the information item. Next, let us assume the adversary has inserted P_i in between P_j and P_{j+1} where $j \geqslant 1$. Similarly, P_j's $PubKey$ field does not have K_i^+ but instead it contains K_{j+1}^+. This again will fail the verification step 6.4 and indicate that P_i is an illegally inserted record.

Proposition 5. *An adversary cannot selectively remove other nodes' records from the preceding provenance chain without being detected by the next auditor.*

First, let us assume a valid provenance chain $P_1|\cdots|P_n$ is supposed to be received by an auditor. Suppose that an adversary has removed the provenance records $P_1|\cdots|P_i$ where $1 \leqslant i < n$, the auditor will assume the P_{i+1} is from the original source and thus expect $PubKey_{i+2}$ to have to K_{i+1}^+. Then the verification step 6.4 will fail. Next, let us suppose an adversary has removed the provenance records $P_i|\cdots|P_j$ where $1 < i \leqslant j < n$. When next auditor verifies the records, again the verification step 6.4 will fail because the auditor cannot obtain K_{j+1}^+ from $PubKey_{i-1}$. Finally, let us suppose an adversary has removed the provenance records $P_i|\cdots|P_n$ where $i > 1$. The verification will fail at the verification step 2 because $PubKey_{i-1}/K_i^+$ is not the auditor's public key, which indicates a part of the provenance chain is missing.

Proposition 6. *An adversary cannot alter the order of the provenance records in the preceding provenance chain without being detected by the next auditor.*

According to Proposition 4 and Proposition 5, we know altering the order of the provenance records in the provenance chain will be detected by the next auditor since order altering consists of a sequence of deleting and adding actions.

Proposition 7. *An auditor can only access provenance details for what he/she is authorized but still can perform integrity and availability verification.*

The symmetric key used to encrypt the provenance details is accessible only to users that can retrieve it by decrypting one of the encrypted symmetric keys in $SKey$. Therefore, only these auditors are able to see all the data encrypted with the symmetric key. Chain verification involves only public key signature operations and no other secret values. All the public keys can be found in the neighbor provenance records' $PubKey$ fields. To verify a node's record, an auditor does not need to know the identity or the identity/public key correspondence. Therefore, even if the identities of certain nodes are inaccessible, an auditor can still perform the verification.

VI. Performance Evaluation

Besides the security assurance capabilities, the computational overhead is of significant concern. In this section, we give an empirical evaluation on the overheads of our scheme and compare it with the Onion scheme.

We implemented both our PKLC scheme and the Onion scheme with Java version 1.6 to measure and compare the performance of these two approaches. We executed the programs on a Linux machine (Ubuntu 10.04 with kernel version 2.6.32) with Intel Core 2 Duo 2.93GHz processor and 3.0GB main memory. Since the information and provenance transmission process is not our concern in this work, we simulated multiple nodes on the same machine. We used 1024-bit RSA for digital signatures and asymmetric encryptions, 128-bit AES for symmetric encryptions and SHA-1 as the hash function. Each provenance chain for a particular information item was created as a separate XML file. It is one of the most commonly used ways to record, store and exchange provenance for different information networks and systems. For the information payload, we used files of 5kb (plain text), 50kb (XML document), 500kb (JPEG image) and 5mb (MP4 video) to measure how the performance varies with different sizes and types of the payload.

The overhead to create, store and verify a provenance record could vary due to different level of provenance details, different number of encrypted fields and authorized groups/individuals. We run our tests based on the following three provenance settings:

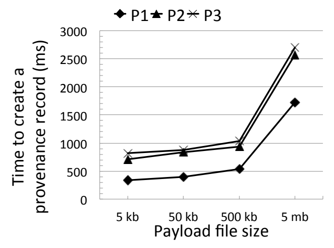
P1 $NodeInfo$ has 1 annotation which is confidential; $ProcessInfo$ has 1 process with 1 plaintext annotation; $OptInfo$ is empty; $Skey$ contains 1 symmetric key.

P2 $NodeInfo$ has 3 annotations and 1 of them are confidential; $ProcessInfo$ has 3 processes each with 3 annotations and 1 of the annotations is confidential; $OptInfo$ contains 2 plaintext annotations; $Skey$ contains 3 symmetric keys.

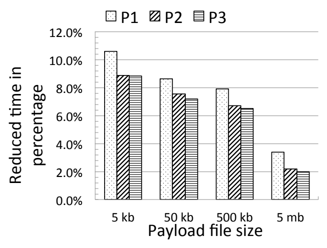
P3 $NodeInfo$ has 3 annotations which are all confidential; $ProcessInfo$ has 3 processes each with 3 confidential annotations; $OptInfo$ contains 2 confidential annotations; $Skey$ contains 5 symmetric keys.

Provenance Construction Overhead: The first experiment we conducted is to measure the computation time to create a provenance record when the payload file size varies based on the above three provenance settings. Figure 4 (a) shows the results with our PKLC scheme. As expected, we can see that as the provenance setting gets more complicated, it takes more time to create a provenance record. Another observation is the P2 and P3 curves are very close to each other whereas they are farther from the P1 curve. Hashing the payload file is a significant component of this difference as it takes a large portion of the total computation time, and P2 and P3 both have three processes in $ProcessInfo$ and each has to at least compute a hash of its output payload while P1 only has one process in $ProcessInfo$.

Figure 4 (b) shows how much our PKLC scheme reduces the provenance construction time as compared with the Onion scheme. The improvement comes from the fact that our scheme does not need to include the previous record's signature before signing it. Basically both of the two approaches require the same cryptographic building blocks. The construction time reduction is not significant when the payload is large, because most of the time would be spent on the payload hashing, which is exactly the same for both schemes. However, the payload is usually small in many cases, e.g.,

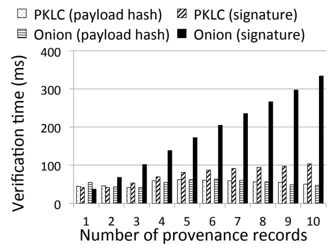


(a) Time to create a provenance record with PKLC scheme

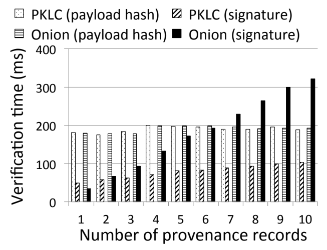


(b) Reduced time to create a provenance record by the PKLC scheme as compared with the Onion scheme

Fig. 4. Provenance construction time of the PKLC scheme and the reduced time as compared with the Onion scheme



(a) Payload file of 50kb



(b) Payload file of 5mb

Fig. 5. Comparing the provenance verification time for an entire provenance chain of the PKLC scheme and the Onion scheme

emails, descriptive reports, etc. When the payload is small, our scheme can reduce the construction time by about 10%.

Provenance Verification Overhead: The next experiment we conducted is to measure the computation time for an auditor to verify an entire provenance chain. Figure 5 shows the results on payload file of 50kb and 5mb. We only measure the time of payload hash verification and signature verification since only they incur costly cryptographic computations. We obtained the computation time of these two steps separately. Since the payload hash verification only requires hashing the received payload file and then comparing with the $PLHash$ of the most recent provenance record, it takes an approximately constant time for both of the PKLC scheme and Onion scheme. The signature verification process is where our approach makes a significant improvement. The Onion scheme requires an auditor to verify the provenance records sequentially from the very first one because each provenance record's validity depends on the validity of the preceding provenance chain. For our PKLC scheme, the verification of a provenance record does not depend on the verification results of the previous records, and thus the verification can be executed concurrently for all the records. As observed in Figure 5, the greater the number of provenance records in the provenance meta-data, the larger improvement our scheme achieves. The ideal signature verification time for our PKLC scheme is constant. However, we can see a small uptick in the verification time needed as the number of records increases. This is due to additional overhead incurred by thread creation and execution in the programs when multiple threads run the verification concurrently.

Storage and Communication Overhead: The entire provenance file can be considered as storage and communication overhead. Our provenance scheme only requires the provenance to contain the hashes of the information payloads and the hashes are of fixed size no matter how large the payload files are. Hence, the size of the provenance XML file is independent of the size of the payload. Our experiment results confirmed that the size of a provenance chain has a linear relationship with the number of provenance records on the chain for a certain provenance setting.

VII. Conclusion and Future Work

Provenance is very important in digital information analysis and forensics. In this paper, we investigated the unique properties of provenance security in distributed information networks and presented the challenges and requirements in assuring provenance security. We proposed a Public-key Linked Chain provenance framework to protect the provenance meta-data as a whole and proved that our approach can properly provide the expected assurances. We gave empirical assessments for the provenance construction, verification and storage overhead. Compared to the Onion scheme, our scheme solves some implementation difficulties, provides better protection and has better performance in terms of computational overhead.

Our scheme depends on the transitive trust to assure the integrity of the data provenance. The transitive trust requires two neighboring nodes not to collude with each other. If two colluding adversaries are not neighboring nodes, their communication can be verified by the intermediate node(s) in between. In the future we will look into the solution of detecting colluding neighbors as well. A good potential direction is to involve a witness when information transaction happens between two adjacent nodes, similar to the conventional *Watchdog* [11] approach used to mitigate collusion attacks in routing protocols. In addition, we will implement our scheme in a real distributed information network application and carry out more in-depth security and performance analysis.

Acknowledgment

This work was sponsored by the Army Research Laboratory and was accomplished under Cooperative Agreement Number W911NF-09-2-0053. The views and conclusions contained in this document are those of the authors and should not be interpreted as representing the official polices, either expressed or implied, of the Army Research Laboratory or the U.S. Government. The U.S Government is authorized to reproduce and distribute reprints for Government purposes notwithstanding any copy right notation here on.

References

[1] Y. Simmhan, B. Plale, and D. Gannon, "A survey of data provenance in e-science," *ACM SIGMOD Record*, vol. 34, no. 3, pp. 31–36, 2005.
[2] R. Hasan, R. Sion, and M. Winslett, "Introducing secure provenance: problems and challenges," in *ACM Workshop on Storage Security and Survivability*, pp. 13–18, 2007.
[3] U. Braun, A. Shinnar, and M. Seltzer, "Securing provenance," in *Proceedings of the 3rd Conference on Hot Topics in Security*, pp. 1–5, 2008.
[4] S. Xu, Q. Ni, E. Bertino, and R. Sandhu, "A characterization of the problem of secure provenance management," in *ISI '09: IEEE International Conference on Intelligence and Security Informatics*, pp. 310–314, 2009.
[5] L. Gomez, A. Laube, and A. Sorniotti, "Trustworthiness assessment of wireless sensor data for business applications," in *AINA '09: Proceedings of the 2009 International Conference on Advanced Information Networking and Applications*, pp. 355–362, 2009.
[6] X. Wang, K. Govindan, and P. Mohapatra, "Collusion-resilient quality of information evaluation based on information provenance," in *SECON '11: IEEE 8th Communications Society Conference on Sensor, Mesh and Ad Hoc Communications and Networks*, 2011.
[7] R. Hasan, R. Sion, and M. Winslett, "The case of the fake picasso: preventing history forgery with secure provenance," in *Proccedings of the 7th Conference on File and Storage Technologies*, pp. 1–14, 2009.
[8] L. Gadelha and M. Mattoso, "Kairos: An architecture for securing authorship and temporal information of provenance data in grid-enabled workflow management systems," in *eScience '08: IEEE 4th International Conference on eScience*, pp. 597–602, 2009.
[9] R. Lu, X. Lin, X. Liang, and X. Shen, "Secure provenance: the essential of bread and butter of data forensics in cloud computing," in *ACM Symposium on Information, Computer and Communications Security*, pp. 282–292, 2010.
[10] B. Wu, J. Wu, and Y. Dong, "An efficient group key management scheme for mobile ad hoc networks," *International Journal of Security and Networks*, vol. 4, no. 1, pp. 125–134, 2009.
[11] S. Marti, T. Giuli, K. Lai, M. Baker, *et al.*, "Mitigating routing misbehavior in mobile ad hoc networks," in *Proceedings of the 6th annual international conference on Mobile computing and networking*, pp. 255–265, 2000.