CERIAS Tech Report 2014-4
Secure Digital Provenance: Challenges and a New Design
by Mohammed Rangwala
Center for Education and Research
Information Assurance and Security
Purdue University, West Lafayette, IN 47907-2086

SECURE DIGITAL PROVENANCE: CHALLENGES AND A NEW DESIGN

A Thesis

Submitted to the Faculty

of

Purdue University

by

Mohammed M. Rangwala

In Partial Fulfillment of the

Requirements for the Degree

of

Master of Science

May 2014

Purdue University

Indianapolis, Indiana

To my parents, brother, late grandfather,

family and friends ...

you keep me going ...

ACKNOWLEDGMENTS

I pride myself for my achievements during my Computer Science graduate degree at IUPUI. Not only did I have the support of the esteemed faculty members at the university but I also had the opportunity to research on various topics. This has been a long cherished dream. I am grateful to everyone who has backed me during this journey.

First and foremost, I would like to thank my advisor Dr. Xukai Zou for extending his support and guidance throughout my graduate studies. He introduced me to the finer nuances of research and encouraged me to explore the unknown. He challenged my imagination and encouraged me through the difficult and frustrating times. His suggestions and insights into matters related to not only this work but studies in general have been invaluable.

I would like to express my deepest gratitude to Dr. Feng Li and Dr. Rajeev Raje for serving on my advisory committee. Their support and guidance helped me to focus better and make wise decisions in my research and studies.

I want to thank my professors Dr. Mihran Tuceryan, Dr. Arjan Durresi, Dr. Yao Liang and Dr. Mohammad Al Hasan whose courses have helped me get a better foundation of the subject. I appreciate the cordial smiles and greetings that I had exchanged with the other faculty members in the corridor and at different seminars. A special thanks to Nicole Wittlief, Scott Orr and Nancy Reddington without whose support things would not be running as smoothly.

I was fortunate to receive the prestigious University Fellowship which gave me the impetus to strive harder towards my goals and perform well in all aspects of my degree program.

This journey would not have been possible without the undying support of my parents, brother and family. They have been instrumental in keeping my morale up

and have given me the courage to believe in myself. This has helped me fight against all odds to reach my goals.

Last but not the least, I would like to acknowledge the support of my friends who have always been with me. I could not have done this research without their willingness to stand by me. I have been asked to follow the quote – "To strive, to seek, to find and not to yield".

TABLE OF CONTENTS

LIST OF TABLES

LIST OF FIGURES

SYMBOLS

K_A	Public key of user A
K_A^-	Private key of user A
$e_A()$	Encryption using key K_A
$h(D_i)$	Cryptographic hash of the document D_i
$sign_A()$	Cryptographic signature using key K_A^-. Signature involves the hash of the data, and is short for $sign_i(h(\ldots))$
U_i	User i
Pr_i	Provenance record corresponding to user i
$Pr_1 \vert Pr_2 \vert \ldots \vert Pr_n$	Chain of provenance records
O_i	Operations performed by user i
C_i	Signature of user i over fields of record Pr_i
$public_i$	Public key certificate of the user i
I_i	Keying material for record Pr_i
$ChainInfo_i$	Sequence of public keys of users involved in the provenance chain till record Pr_i
S_i	Symmetric keys for record Pr_i
$PubKey_i$	Public keys of previous and next users for record Pr_i
P_i	Previous signature field representing the previous user in the chain for record Pr_i
N_i	Next signature field representing the next user in the chain for record Pr_i
IV	Initialization Vector

ABBREVIATIONS

PKLC	Public-Key Linked Chain
DAG	Directed Acyclic Graph

ABSTRACT

Rangwala, Mohammed M. M.S., Purdue University, May 2014. Secure Digital Provenance: Challenges and a New Design. Major Professor: Xukai Zou.

Derived from the field of art curation, digital provenance is an unforgeable record of a digital object's chain of successive custody and sequence of operations performed on the object. It plays an important role in accessing the trustworthiness of the object, verifying its reliability and conducting audit trails of its lineage. Digital provenance forms an immutable directed acyclic graph (DAG) structure. Since history of an object cannot be changed, once a provenance chain has been created it must be protected in order to guarantee its reliability. Provenance can face attacks against the integrity of records and the confidentiality of user information, making security an important trait required for digital provenance. The digital object and its associated provenance can have different security requirements, and this makes the security of provenance different from that of traditional data.

Research on digital provenance has primarily focused on provenance generation, storage and management frameworks in different fields. Security of digital provenance has also gained attention in recent years, particularly as more and more data is migrated in cloud environments which are distributed and are not under the complete control of data owners. However, there still lacks a viable secure digital provenance scheme which can provide comprehensive security for digital provenance, particularly for generic and dynamic ones. In this work, we address two important aspects of secure digital provenance that have not been investigated thoroughly in existing works: 1) capturing the DAG structure of provenance and 2) supporting dynamic information sharing. We propose a scheme that uses signature-based mutual agreements between successive users to *clearly delineate the transition of responsibility of*

the digital object as it is passed along the chain of users. In addition to preserving the properties of confidentiality, immutability and availability for a digital provenance chain, it supports the representation of DAG structures of provenance. Our scheme supports dynamic information sharing scenarios where the sequence of users who have custody of the document is not predetermined. Security analysis and empirical results indicate that our scheme improves the security of the typical secure provenance schemes with comparable performance.

1 INTRODUCTION

In this chapter, we give a brief introduction about digital provenance and our work in this thesis.

1.1 What is Digital Provenance?

Provenance refers to the origin or earliest known history information of an object. The concept of provenance originates from the field of art and archiving, where it refers to information about the artifact's creation, the chain of custody and modifications performed on it. It has been an important concept in many fields other than art, like science and computing where it is used to trace an object to its origin. It is used in work-flow management systems and processes in physics, astronomy, biology, chemical sciences, earth sciences for maintaining context information, auditing and data replication [1]. It finds applications for intelligent re-use of experiments, fault detection, protection against illegitimate intellectual property claims, detecting plagiarism and identity fraud, and assessments of data quality [2]. Depending on the application domain, different properties of the object can be tracked such as owner information, purpose of its creation, processes undergone, state of the object or material at each stage, etc. Since provenance maintains information about the present and past of an object, it is suitable to assess the object's trustworthiness [3].

Digital provenance is the provenance associated with digital objects which can be resources in hardware, software, documents, databases and other entities. It maintains information about the chain of successive custody of the object with different users and the sequence of operations performed on it. It can store functional data such as the process results as well as non-functional data such as the performance of each

step. Most computer systems track information for error correction and debugging, as discussed in [4], such as:

i) Operating systems store logs of important system events which help in system administration and intrusion detection.

ii) File systems store information about file creation, modifications performed, permissions to the file, etc.

iii) Version Control Systems record information about the modifications made to different objects.

iv) Web browsers store history information about the web pages visited and when.

These can be considered as different forms of provenance information, which are application specific. However, each of these systems does not provide a definition for provenance.

Digital Provenance finds applications in a number of areas [5]. Some of them are:

i) Verification of scientific data and experiments [6–14];

ii) Supporting or facilitating data sharing [15–18];

iii) Copyright clearance [19];

iv) Legal proceedings involving data [20];

v) Tracking operations on data in cloud environments [21–23];

vi) Recently in facilitating data mining [24]; tracing system activities in Android devices [25]; stream management [26, 27];

Provenance systems are specially designed application-specific frameworks used to collect, analyze and store all metadata information of an object. They can then be queried to obtain the history information, perform audits and validation checks and detect faults. Research in provenance has focused on developing such frameworks for

a variety of systems like work-flow management, grid computing, file systems, cloud systems. We mention some of these in Chapter 3.

Digital provenance introduces some challenges with respect to its definition, management and security [4]. Some of these challenges are:

i) Completeness: Since provenance contains history information, it is necessary to define how much recorded information is considered to be enough. Depending on the application, it maybe necessary to record the output of each individual operation performed on the object. This is important because completeness of the provenance will define the complexity of a provenance system.

ii) Reliability: Provenance must be reliable since it finds applications in fault detection, identity theft and plagiarism detection, etc. It is necessary for the provenance to be secure against any kind of tampering after it has been created.

iii) Heterogeneity: A digital object can undergo several operations that may produce different meta-data information. It may also be recorded at different levels of granularity. Thus, provenance can contain heterogeneous information which introduces the challenge of uniform consistent representation.

iv) Portability: Since provenance is associated with a digital object, it must be bound to it. As the object moves in the system, its provenance must move along with it. This requires the provenance to support portability in the system.

v) Dynamic nature: Different users may operate on the digital object at different points in its lifetime. In a distributed information network or wireless sensor network, the sequence of nodes through which the object passes is predetermined (to a certain extent). But in different scenarios, it may be possible that the sequence is dynamic. The structure of the provenance and the provenance system itself must be able to support this.

Provenance can be represented as a causality graph that connects different objects with edges that describe the process by which the object transformation took place

[28]. This forms an immutable directed acyclic graph (DAG) structure. Although an object keeps changing with operations performed on it, its history information does not and so, provenance is immutable. The DAG structure is justified since an object can be copied to multiple instances (or provided as input to multiple processes) and it can be created from a combination of objects (or from outputs of multiple processes). In a graph, these cases represent a node having multiple children or multiple parents respectively. Since history inforamtion does not repeat, the graph does not have any cycles. There is no established standard for representing provenance information, but XML is most popularly used [1]. The existing limited security mechanisms for provenance do not appropriately apply to DAG structures.

Depending on the application domain, provenance can be more or less sensitive than the data object itself. For e.g., in an employee review system, the sequence of managers who have added to the review must not be disclosed to the employee. Thus, the ownership information in the provenance chain in such a scenario must be kept confidential. Here, the provenance is more sensitive than the document it is associated with. Consider another example of a professor's recommendation for a student's university application. The recommendation document itself needs to be kept confidential from the student, but the provenance containing the information of the professor(s) can be disclosed. In such a scenario, the document is more sensitive than its provenance. Apart from this, like other information security subjects, digital provenance requires integrity and availability, along with suitable and efficient representation. In this respect, the security requirements of provenance differ from those of traditional data [28]. Thus, a general scheme for secure provenance is needed, which can be modified depending on the application scenario.

Recent research in provenance has focused on developing provenance generation, storage and management frameworks in different fields but limited work has focused on the security and privacy issues related to it. We recognize that two aspects have not gained enough attention: 1) capturing the DAG structure of provenance and 2) supporting dynamic information sharing. In this work, we propose a scheme that uses

signature-based mutual agreements between successive users to secure the provenance chain. It is an interactive protocol that clearly delineates the transition of responsibility of the digital object as it is passed along the chain of users. A related provenance scheme was proposed by Hasan et. al. [29]. This scheme is referred to by Wang et. al. [30] as the *Onion* scheme due to its layered provenance format. They showed that the Onion scheme has certain weaknesses and proposed a linked chain structure of provenance using public keys. This scheme is referred to as the Public-Key Linked Chain ($PKLC$) scheme [30]. The PKLC scheme works well for distributed information systems but cannot handle all the properties required in other digital systems. Our solution extends their work and solves the problems associated with it.

1.2 Main Contributions of this thesis

The contributions of our work can be summarized as:

- A signature-based mutual agreement scheme is proposed to form links between provenance records. Our scheme provides better security than the Onion scheme [29], and, is an extension and improvement over the PKLC [30] scheme to provide secure provenance in digital systems other than distributed information networks.
- Our scheme can adequately support the representation of DAG structures of provenance.
- It can also support dynamic information sharing scenarios where the next user to whom the data will be passed is not predetermined. A summary of the advantages of our scheme is provided in Table 4.1.
- An analysis of the security of our scheme is provided to show that it satisfies the security requirements of a provenance scheme.
- Experimental evaluations are provided for the overhead of our scheme. The overhead of our scheme for provenance record creation is a little more than the

other schemes but we argue that it can be outweighed by the security provided by our scheme. The results show that it performs better than the Onion scheme for provenance chain verification.

1.3 Organization of this thesis

The rest of this thesis is organized as follows. In Chapter 2, we discuss the fundamental concepts involved in a provenance scheme, the important properties required for its security and an attack model that must be considered. We highlight the previous proposed mechanisms in Chapter 3. Chapter 4 describes our mutual agreement signature scheme, along with an example and discusses its properties. We analyze the security of our scheme with respect to the attack model in Chapter 5. Performance evaluation and comparison with existing schemes is provided in Chapter 6. Chapter 7 talks about future work and concludes the thesis.

2 PRELIMINARIES FOR A SECURE PROVENANCE SCHEME

In this chapter we describe and provide definitions for some fundamental concepts related to digital provenance. We discuss the different entities involved in a secure provenance scheme; the security properties required from it; and a general attack model for building a secure provenance scheme.

2.1 Entities involved

A **document** D is a data item such as a file, database tuple or network packet for which provenance is to be generated and maintained. In this work we use the term document abstractly; its exact form is domain and application-specific.

Provenance of a digital document is an account of all the actions performed on it right from the point of creation. Each access to the document can create a provenance record Pr; multiple such records are maintained in order as a **provenance chain** $Pr_1|Pr_2|\ldots|Pr_n$. Provenance of a document forms a directed acyclic graph (DAG) structure [28]. We refer to the provenance of the document as a 'chain' in this work because records are arranged sequentially, but they may not be linearly linked to each other. A provenance record stores in it an account of the operations performed by a user on the document, and relevant information that help maintain links between the different records that are part of the chain.

Users are the entities who have or have had custody of the document. They may perform operations on the documents, e.g. create, rename, read, write, delete in the case of a file system. The user who first creates the document and is associated with the first record of the provenance chain is referred to as the **owner** of the provenance. This is different from the current owner of the document. In this work we refer to owner as the owner of the provenance chain.

An **auditor** is an entity who can check all provenance information to verify the lineage of a document. An auditor performs an **auditing** activity, which involves traversing the provenance records in the chain and checking their fields to ensure that the chain has not been tampered with. Different users may trust different auditors with sensitive information, thus, a document has a set of auditors who can access different sensitive fields in the records.

Outsiders are entities who do not have access to the documents, and subsequently should not have access to any part of its provenance.

An **adversary** has access to the provenance and wants to alter it in some way for malicious intents but remain undetected. An adversary may be a user who has already contributed to the provenance chain or an outsider.

2.2 Properties of a provenance scheme

After discussing the fundamental entities, we discuss the properties that a scheme must provide for the provenance data. Groth et. al. [31] identified a set of properties that any provenance system must provide. We list them here:

i) Verifiability: The provenance scheme must be able to verify a process with respect to the users involved, operations performed and results obtained.

ii) Accountability: The scheme must hold the user accountable for his/her actions, i.e. a user should not be able to repudiate any actions.

iii) Reproducibility: The provenance should contain enough information for it to be possible to reproduce the same results if the sequence of operations recorded is re-executed.

iv) Preservation: Since provenance contains history information, it must be maintained for a sufficiently long period of time.

v) Scalability: For large scale applications, a large amount of provenance data may be generated which requires the scheme to be scalable.

vi) Generality: It is possible for a wide variety of meta-data to be generated from an application and the provenance scheme should be general enough to be able to capture them.

vii) Customizability: The scheme must allow customization to be able to record any application-specific details at different levels of granularity.

viii) Portability: Provenance is associated with a digital object, and there must be a mechanism to ensure that they cannot be separated. Along with this, the scheme must allow the provenance to move in the system when the data moves.

We now discuss the properties that a scheme must provide for securing the provenance chain which are more related to our work. These are extended from the fundamental general properties of data security. We mention the properties here to get an understanding of the security required for a provenance chain and discuss how our scheme achieves them in Section 4.3.

Confidentiality: A provenance record may contain sensitive information regarding the operations performed on the document as well as its ownership history that should not be revealed to unauthorized entities. The sensitivity of these fields is domain and application-specific. For e.g., in an employee review system, the sequence of managers who have added to the review must not be disclosed to the employee. Thus, the ownership information in the provenance chain in such a scenario must be kept confidential. This is different from the confidentiality of the document itself. Thus, provenance and the document may have different confidentiality requirements. The properties that are required are:

i) An auditor should be able to verify the complete lineage of the document, without access to the sensitive information in the records.

ii) Since different users may trust different auditors, the sensitive information may not be revealed to all auditors.

Integrity: Since provenance contains history information which is immutable, integrity of the provenance is the most important property that a scheme must satisfy. There are three types of integrity associated with provenance [30]:

i) Immutability (Chain Integrity): The provenance chain once formed should not be modifiable, i.e. the order of the records cannot be changed.

ii) Data Integrity: The information in the individual provenance records should not be tampered with.

iii) Non-repudiation (Origin Integrity): A user's action in the chain cannot be undone, i.e. the user cannot repudiate his actions.

Availability: Provenance is associated with a document and when it is passed between users, the provenance chain is passed along with it. The scheme must ensure that when the document is passed between users, the chain remains intact and is not modified without being detected in the auditing activity.

Efficiency: Depending on the application domain, the provenance generation process and scheme participates either when operations are being performed on the document (when outputs of individual operations must be recorded) or after all operations have been performed. In both cases, the provenance scheme adds a computational overhead on the application. The scheme must be designed such that the overhead is not significant.

As seen in this discussion, the representation and properties of provenance can be different can be different from those of the document it is associated with. Each individual provenance record can have different confidentiality requirements, whereas the integrity and availability of all records in the chain must be protected as the chain grows. This introduces new challenges in the security of provenance, making it different from the security of traditional data.

2.3 Attack Model

Here we briefly discuss some of the goals of the adversary in a digital provenance scheme similar to discussions in [29, 30]. A detailed analysis of attacks and their prevention in our scheme is discussed in Chapter 5. A provenance scheme must consider an adversary with the intentions of,

i) obtaining confidential information from the provenance records about the operations performed on the document;

ii) obtaining information about the ownership history of the document;

iii) using fake or stolen key-pairs to make their own provenance records un-verifiable;

iv) modifying existing records (tampering or changing order of records) or adding forged information to the existing provenance chain;

v) selectively removing a certain part of the preceding provenance chain.

Our scheme should be designed such that these goals are either prevented or made detectable to an auditor in the auditing activity.

These preliminaries lay the foundation for understanding the existing work done in secure digital provenance (Chapter 3) as well as our scheme (Chapter 4).

3 BACKGROUND & RELATED WORKS

Before discussing our scheme in detail, we discuss some of the research that has been conducted with respect to the security of digital provenance and give an insight into the motivation for our work.

Research has been done to develop conceptual frameworks and models for provenance management [11, 13, 14, 31–42]; to identify the security requirements of provenance systems [4, 5, 43] and provenance management and data forensics in cloud environments [21–23, 44–47].

Hasan et. al. [20] were among the first to propose the concept of secure provenance. Although provenance had been studied in many applications and fields, they identified that the security issues had not been considered. They defined the properties required from a secure provenance scheme along with a threat model and challenges. Braun et. al. [28, 48] discussed some of the essential characteristics of provenance and how it is different from other data in terms of security. They were among the first to recognize that the provenance graph is a directed acyclic graph (DAG) structure to which traditional security measures cannot be directly applied.

Kairos [2] is an architecture for securing the data authorship and temporal information in provenance records suited for work-flow-based grid computing environments. It uses techniques from public key infrastructure (PKI) such as certificate authorities, digital signatures and time stamping protocols to protect provenance records. Kairos has a centralized architecture involving a certificate authority and a time stamp authority, which in combination are responsible for time stamping and signing a provenance record for user of the grid application. The architecture aims at protecting the provenance record but does not give details about the structure of the record itself.

Sultana et. al. [49, 50] proposed a lightweight method for detecting provenance forgery in wireless sensor networks. Since such systems are power and memory constrained it is necessary for the provenance management to be efficient in storage and transmission and not be computation intensive. In this scheme, they use bloom filters to encode provenance information to be able to detect packet drop attacks. Sultana et. al. also proposed a method to securely transmit provenance information in streaming media [27]. Though these works are not directly related, they consider some of the characteristics of confidentiality and integrity preservation required by our scheme.

Alharbi et. al. [51] proposed a privacy-preserving data provenance scheme to ensure the security of provenance for documents on remote servers. The main focus of the scheme is to preserve the privacy of the users through the use of hash chains and group signature techniques, and employs the use of a trusted authority and trusted servers. Our scheme makes use of only a trusted auditor but is not focused at remote document operations.

3.1 The Onion scheme

The Onion scheme [29, 52] is closely related to our work. In this section, we give a brief overview of this scheme and discuss its shortcomings.

3.1.1 Overview of the scheme

The Onion scheme was the first to define a concrete structure of a provenance record. Each individual provenance record in a chain of records has the following structure:

$$Pr_i = < U_i,\ O_i,\ h(D_i),\ C_i,\ public_i,\ I_i >$$

We limit our discussion of the fields of the records in this chapter, since they will be elaborated in Chapter 4 when we discuss our scheme. Here, U contains the user's

information, O is a representation of the operations performed on the document, *h(D)* is the hash of the document, C is the checksum of the record, *public* stores a public key certificate of the user and I contains keying material. Multiple such records arranged sequentially form the provenance chain. The checksum field also contains the checksum of the previous record in the chain and this makes it an incremental chained signature mechanism. Each record uses the checksum signature over the previous record's signature to preserve the integrity of the complete chain. Since the checksum field is layered, it gives the chain an onion-like structure [30]. The U and O fields may contain sensitive information. They can be kept confidential using symmetric keys with auditors, which are stored in I.

3.1.2 Problems with the scheme

The Onion scheme has certain weaknesses [30]. First, it cannot protect the outermost layers of the provenance chain, i.e. the newest records. An insider attacker can easily extract a prefix of the complete chain, sign over the signature of the last record in the extracted chain and insert a new record. The flaw comes from the fact that this scheme is not based on a hand-off mechanism when the document is passed between users. Consecutive records in the chain are loosely linked to each other.

Second, the scheme requires the trusted auditor(s) to maintain user-key relationship which violates the confidentiality of the users. Our scheme involves a *ChainInfo* field in the provenance record, which sequentially stores the public keys of all users involved in the preceding chain. This field provides the keys necessary to perform the operations with the records, but does not reveal any identity information of the users even to the trusted auditor(s).

Also, the scheme cannot support the DAG structure of provenance. It is restricted to the scenario of a single document being passed along a chain of users. Our scheme overcomes these weaknesses by introducing a mutual agreement mechanism, when the document is passed between users, which creates strong links between their prove-

nance records. It also uses multiple fields for signatures to handle the case of multiple parents or children in a DAG structure.

3.1.3 Scheme related to the Onion scheme

Syalim et. al. [53] proposed a scheme based on the Onion scheme. In this scheme, every document and its provenance passed between users is signed by the previous user as well as the owner of the provenance to preserve the integrity of the chain. They define a path-based policy as well as a compartment policy for providing access control on the provenance graph structure. The provenance records are encrypted using multiple keys that are handled by the provenance owner.

The shortcoming of this scheme is that it makes use of the involvement of the provenance owner to satisfy the properties of the system. This heavy involvement of the owner at each step of the provenance scheme is undesirable. This is avoided in our scheme, through a mutual agreement mechanism between only the users who are involved in passing the document at a particular time. Also, a large number of encryption and signature operations are performed which make the scheme inefficient.

3.2 The PKLC Scheme

The Public-Key Linked Chain (PKLC) scheme [30] is most closely related to our work. We give a brief discussion of this scheme.

3.2.1 Overview of the scheme

The PKLC scheme [30] is based on advancements to the Onion scheme applied to a distributed information network. It uses a record format similar to that of the Onion scheme, and has the following structure:

$$Pr_i = < U_i,\ O_i,\ h(D_i),\ S_i,\ PubKey_i,\ C_i >$$

Though the fields of the PKLC scheme are originally named differently, here we use the same notations as those of the Onion scheme for convenience. The structure is suited for distributed information networks or wireless sensor networks. U is similar to that of the Onion scheme where it stores information about the node performing operation on the document. O contains information of every individual operation that is performed on the document by the same node. $h(D)$ is the hash of the document and S, similar to the I field stores symmetric keys used to encrypt sensitive information. C contains the signature of the node over the complete record. The scheme differs in the manner in which it links records of the chain. A record can contain the previous and next user's public keys in the $PubKey$ field to link the records.

3.2.2 Problems with the scheme

The links between the records are weak since they are formed with only the public keys of the users. This scheme is suitable for distributed information or wireless sensor networks where each user initially knows the identity of the next user to which the document passes. In such a scenario, the weak links are enough to preserve the integrity of the chain. The auditors know the path of information flow among users and can thus verify the chain of records. But this cannot be applied directly to dynamic information sharing scenarios where the next user is not predetermined. Our scheme builds on this drawback of the PKLC scheme. It does not require the identity of the next user to be known, but ensures the integrity by having the users engage in a mutual agreement scheme at the moment when the document is passed between them.

To denote the owner of the provenance chain, the first record contains the public key of the owner in the previous field as well. If applied to a general scenario, it is susceptible to an owner forgery attack. An adversary can remove the records of the chain and claim to be the owner by creating a record with his/her own public key in the previous field. Thus, a stronger mechanism is required for representing and

distinguishing the owner of the provenance from other users. Our scheme handles this by involving the auditor.

We have now laid the foundation for a secure provenance scheme and discussed previously proposed provenance schemes along with their shortcomings. We now discuss our proposed design in the next chapter.

4 MUTUAL AGREEMENT SIGNATURE SCHEME

In this section we present our scheme for secure digital provenance.

4.1 Assumptions of our scheme

Before we discuss our scheme in detail, we mention our assumptions.

i) Our scheme considers only the format of the provenance records and chain. It does not focus on the storage and maintenance of the chain. Storage systems such as PASS [33], Flogger [35] can be used for this purpose.

ii) The provenance generation and storing mechanism is not compromised. Our scheme focuses on securing the provenance from attacks after it has been created and stored securely.

iii) The keys used for signatures and encrypting the fields are never compromised or revoked.

iv) The document and its provenance are inseparable, i.e. when a document is passed, the provenance chain is also passed with it. This must be maintained by the provenance storing mechanism.

v) Our scheme relies on transitive trust defined in [30]. That is, pairs of users involved in the document passing trust each other. Thus, we assume that consecutive pairs of users do not collude.

4.2 Structure of provenance record and chain

The provenance chain is composed of a sequence of individual provenance records. Each record stores fields that contain information about the user, operations per-

formed, chain of custody of the document thus far, and a representation of the previous and next users in the chain. It has the following structure:

$$Pr_i = <U_i,\ O_i,\ h(D_i),\ ChainInfo_i,\ S_i^*,\ C_i,\ P_i^+,\ N_i^*>$$

Figure 4.1 gives a representation of the structure of the the provenance chain and each individual provenance record for the scheme. Each of the fields of a record is explained as follows.

$\mathbf{U}_i$ contains identity information about the user i who creates this provenance record. This information is specific to an application domain. For a file system provenance record, U_i includes user ID, process ID, ipaddress, port, host, time, and so on.

$\mathbf{O}_i$ gives a representation of the sequence of operations or modifications performed on the document by user i. This is also dependent on the application domain. For the file system provenance record, O_i includes a file diff, log of changes or operations, or any other reversible representation [29]. It can also contain subfields for representing the operations performed by different processes under the same user as in [30]. O_i contains a reversible representation of the operations if the application domain supports it. By reversible we mean that given document D_i and O_i, it is possible to obtain D_{i-1}.

U_i and O_i contain information about the identity of the user and the operations performed, which may be sensitive to the application. They may be encrypted, in which case the S_i is used.

$\mathbf{h(D}_i\mathbf{)}$ is the cryptographic hash of the contents of the document D_i after user i performs all operations. A hash function is a one way function that is almost always unique for different documents. As the document is modified along the chain, a hash in each record uniquely represents the state of the document at that instant.

ChainInfo$_i$ is a representation of the chain of custody of the document tracked from its origin. It is a sequence of the public keys of all users involved with this document from the owner of the document U_1 to the current user U_i represented as $K_{Aud}|K_1|K_2|\ldots\ |K_i$. K_{Aud} is the public key of the auditor, who must begin the provenance chain. The purpose of K_{Aud} will become clear in the further discussion.

$\mathbf{S}_i^*$ stores symmetric keys that may be used to encrypt the sensitive U_i and O_i fields. We adopt the broadcast encryption scheme of [29] to regulate the access for different auditors. Instead of creating multiple encrypted versions of the sensitive fields for each auditor, user i encrypts them with a symmetric key K_s, and then stores copies of K_s encrypted with the keys K_{Aud_j} of the respective auditors. The * indicates there may be zero or more symmetric keys depending on whether encryption is required and the number of auditors user i trusts.

$\mathbf{C}_i$ is the digital signature over the fields of the same record i signed by the user U_i with key K_i^-, represented as:

$$C_i = sign_i(U_i,\ O_i,\ h(D_i),\ S_i)$$

Since the private key is confidential to a user, assuming it is not stolen, it is not possible to forge the signature of user i. The signature over the fields $< U_i, O_i, h(D_i), S_i >$ ensures the integrity of the record.

$\mathbf{P}_i^+$ is the previous digital signature field which is a representation of the previous provenance record in the chain. The $^+$ indicates there may be more than one previous provenance record from different provenance chains. For the first user U_1 in the provenance chain, this field is signed by the auditor with key K_{Aud}^-.

$\mathbf{N}_i^*$ is the next digital signature field which is a representation of the subsequent provenance record in the chain. The * indicates there may be zero or more subsequent provenance record for a split into different provenance chains.

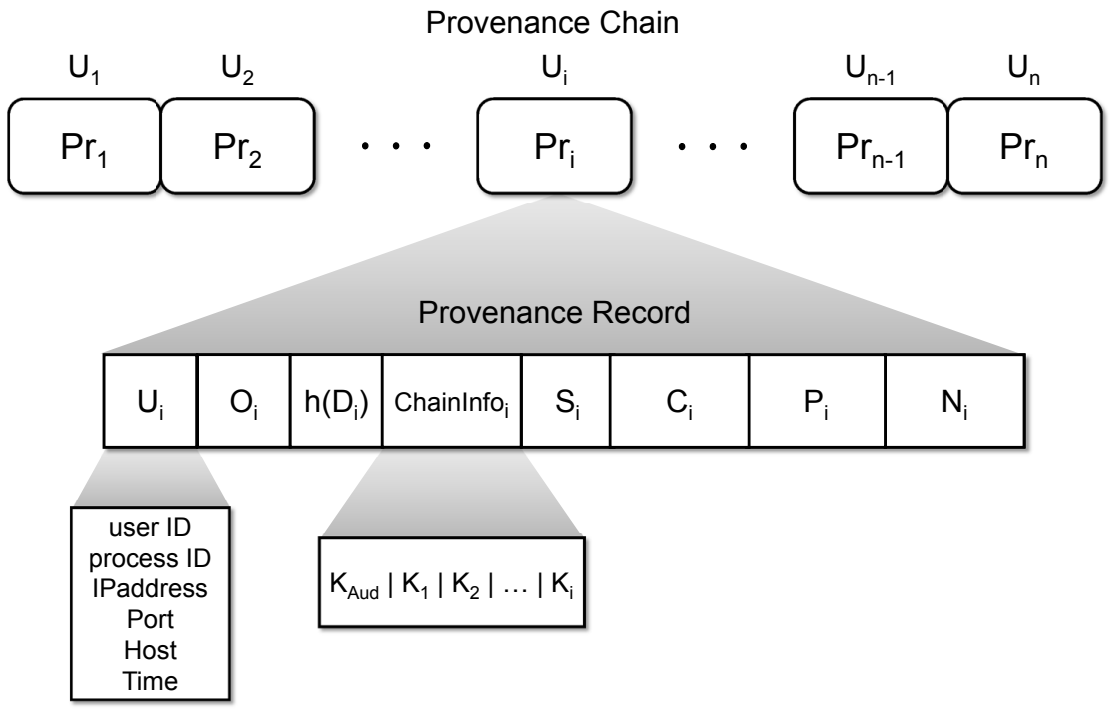

Figure 4.1. An illustration of the structure of provenance

The P^{+} and N^{*} fields are crucial for linking the different provenance records into a chain and for easing the verification process. They form the basis for the mutual agreement scheme. These fields are explained as:
For record i:

$$P_i = sign_{i-1}(h(D_{i-1}),\ ChainInfo_{i-1}|K_i,\ C_{i-1})$$
$$N_i = sign_{i+1}(h(D_i),\ ChainInfo_i|K_{i+1},\ C_i)$$

For record $i+1$:

$$P_{i+1} = sign_i(h(D_i),\ ChainInfo_i|K_{i+1},\ C_i)$$
$$N_{i+1} = sign_{i+2}(h(D_{i+1}),\ ChainInfo_{i+1}|K_{i+2},\ C_{i+1})$$

It can be seen that the N field of record i is signed by user U_{i+1}. The P field of record $i+1$ is signed by user U_i. User U_i after creating provenance record Pr_i passes the document to the user U_{i+1} who can then create the record Pr_{i+1}. An

Algorithm 1 Provenance record creation steps

User i creates record Pr_i:

$Pr_i =< U_i, O_i, h(D_i), ChainInfo_i, S_i^*, C_i, P_i^+, N_i^* >$

if $i = 1$ **then** ▷ User 1 is the creator of the document

  $U_1 \leftarrow e_{S_1}$(User 1 Information)

  $O_1 \leftarrow \phi$ ▷ U_1 is the creator of the document

  $h(D_1) \leftarrow$ Hash of document created D_1

  $ChainInfo_1 \leftarrow K_{Aud}|K_1$

  $S_1 \leftarrow e_{Aud}(\mathrm{K}_{S_1})$ ▷ Multiple for different auditors

  $P_1 \leftarrow sign_{Aud}(IV, K_{Aud}|K_1, C_1)$ ▷ Auditor creates unique IV for this document

  $N_1 \leftarrow \phi$

  $C_1 \leftarrow sign_1(U_1, O_1, h(D_1), S_1)$

else ▷ User i gets the document from user *i-1*

  $ChainInfo_i \leftarrow ChainInfo_{i-1}|K_i$

  $P_i \leftarrow sign_{i-1}(h(D_{i-1}), ChainInfo_i, C_{i-1})$

  $N_{i-1} \leftarrow sign_i(h(D_{i-1}), ChainInfo_i, C_{i-1})$

  User i modifies document D_{i-1} to D_i

  $U_i \leftarrow e_{S_i}$(User i Information)

  $O_i \leftarrow e_{S_i}$(Operations performed)

  $h(D_i) \leftarrow$ Hash of modified document D_i

  $S_i \leftarrow e_{Aud}(\mathrm{K}_{S_i})$ ▷ Multiple for different auditors

  $N_i \leftarrow \phi$

  $C_i \leftarrow sign_i(U_i, O_i, h(D_i), S_i)$

end if

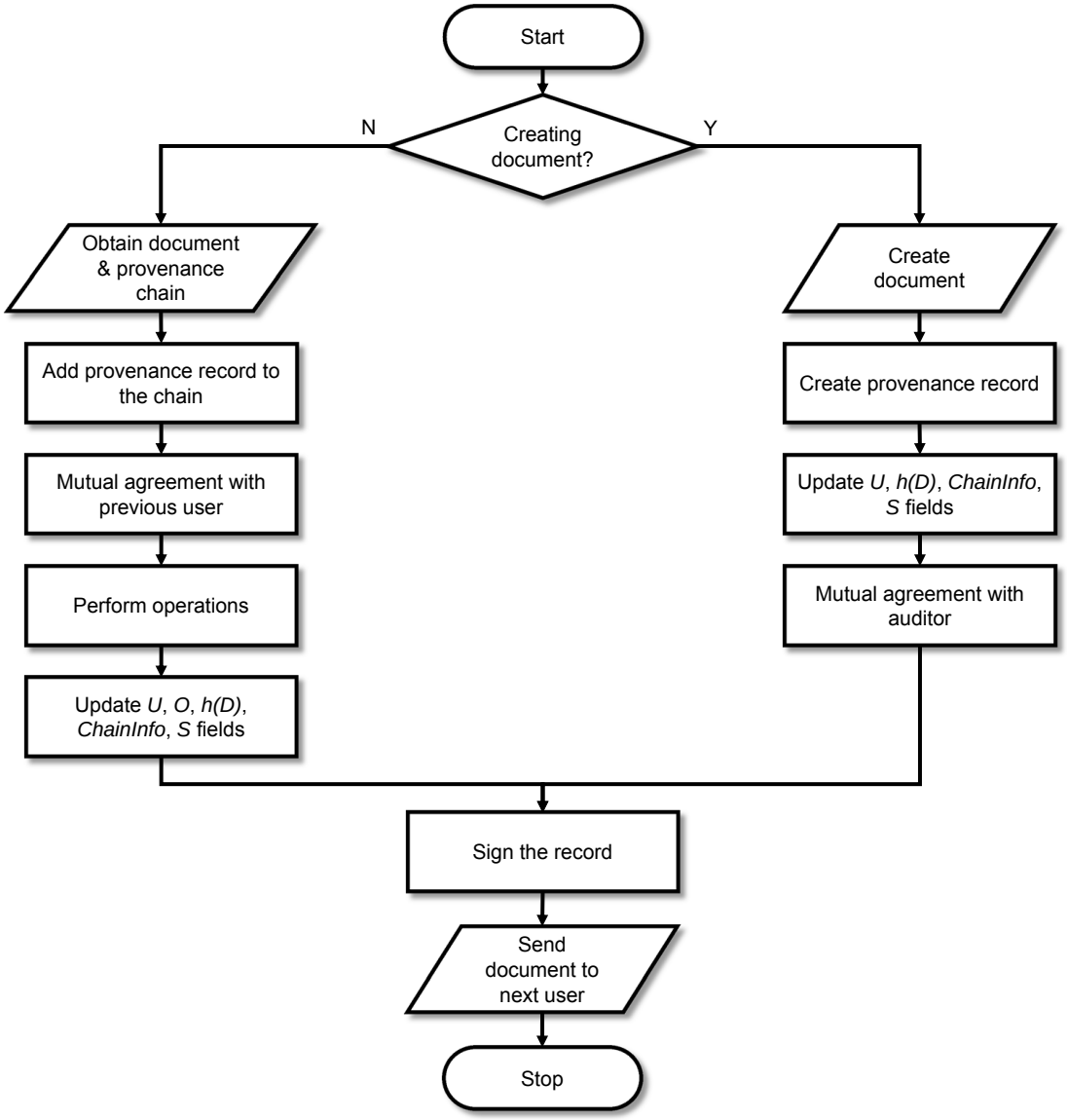

Figure 4.2. Flowchart of a user's actions for creating a provenance record

agreement is signed by both users, such that the record Pr_i contains the signature of U_{i+1} and Pr_{i+1} contains the signature of U_i. As can be seen, the fields N_i and P_{i+1} hold signatures over the same data which is the agreement between the users. The agreement between users U_i and U_{i+1} consists of the fields:

$$<h(D_i),\ ChainInfo_i|K_{i+1},\ C_i>$$

It indicates that the user U_i passes a document having hash $h(D_i)$ to U_{i+1}, the sequence of users in the history of the document including U_{i+1} is $ChainInfo_i|K_{i+1}$, and C_i is the representation of the actions performed by $h(D_i)$. The agreement is an interactive hand-off mechanism where user U_i passes the document to user U_{i+1} and claims to have passed the provenance intact. It delineates the transition of responsibility of the document from user U_i to U_{i+1}.

The previous field for the first user in the provenance chain is signed by the auditor with key K_{Aud} and contains an initialization vector IV in the hash field which is known to the auditor. This IV is unique for each provenance chain. The purpose of IV and the P_1 field signed by the auditor is to prevent an owner forgery attack which is discussed in detail in Chapter 5. IV is a place filler for the hash field, but is not required.

The steps followed by a user for creating a provenance record are described in Algorithm 1. It can be seen that the provenance record is constructed incrementally, the fields are created at each step when the user obtains the document, performs operations and passes it to the next user. Figure 4.2 shows a flowchart of the steps followed by a user for creating a provenance record. The user follows different courses of actions depending on whether he/she is creating the document, or is obtaining the document from another user. As can be seen, in both cases a mutual agreement takes place either with the auditor (if the user creates the document) or with the previous user (when the user receives the document).

4.3 Properties satisfied by our scheme

We briefly analyze our scheme for the properties discussed in Section 2.2.

Confidentiality: The information contained in the provenance records may be required to be kept confidential, for e.g., proprietary algorithms, identity of the user, etc. which are stored in the U and O fields. These fields need to be kept accessible only to the trusted auditor or group of trusted auditors. One approach is to

encrypt multiple of copies of U and O with different keys for each trusted auditors. However, this leads to a large number of copies of the data. Instead, we achieve the confidentiality by using the broadcast encryption scheme of [29]. The U and O fields are encrypted using a unique symmetric key K_s generated for the record. Multiple copies of this key are then encrypted each with the key of a different trusted auditor. The S field in the provenance record stores encrypted versions of the symmetric keys. This scheme reduces the number of encrypted copies of the data to be maintained, by maintaining multiple copies of the key instead. This encryption ensures that the information in the U and O fields is only accessible to the owner of the provenance record and the auditor(s) he/she trusts. No other entity should be able to access them.

Integrity: It is required that any attacks on the provenance chain, such as tampering of the content of the records, addition or removal of records from the chain should be detected by an auditor while performing the auditing activity. In order to facilitate this, our scheme uses hashes and signature fields to hold users accountable for their actions. Our scheme has the following mechanisms to provide the three types of integrity we have discussed:

i) The current signature C is a representation of the actions performed by the user on the document. It protects the fields of the same record providing the property of data integrity.

ii) The previous P and next N signature fields form the mutual agreement between users that links the provenance records of the chain when a document is passed. This provides chain integrity.

iii) Since each record contains the N field, a user's actions in the agreement are signed by the next user as well. This provides origin integrity i.e. a user cannot repudiate his/her actions.

iv) The previous field of the first record is signed by the auditor and the agreement contains a unique initialization vector. This is done to protect the chain

against an owner forgery attack where another user claims to be the owner of the provenance chain.

Availability: A provenance chain must remain intact when it is passed between users along with the document. Through the use of mutual agreements and current signature fields, the scheme ensures that the provenance chain remains intact and any modifications or malicious activity can be detected in the auditing process. The mechanisms for providing confidentiality and integrity indirectly also provide the property of availability. An auditor can perform auditing as per the steps in Algorithm 2, which is discussed in Section 4.5.

Our scheme relies on transitive trust among users. That is, it relies on the assumption that when a document is passed between two users, the users trust each other and do not collude. Our scheme can prevent collusion between users as long as they are not consecutive in the chain. However, if consecutive users collude to modify their provenance records, it is possible for the records to have appropriate signatures for the agreement signature fields. This change can go undetected in the auditing process. Consider the following scenario: Users A, B, C and D are successive users in the provenance chain of a document. After B has created the record, it is passed to C. Assume B and C collude to change the operations B performed on the document. In such a case, B can change the representation of O_B, $h(D_B)$ and the current signature C_B. The mutual agreement between B and C contains C's signature over N_B. Since they collude, it is possible for them to change the mutual agreement to reflect their malicious changes and sign over them. This change will go undetected in the auditing activity. Our scheme does not prevent this scenario of successive users colluding.

A thorough security analysis of our scheme with respect to the attack model is provided in Chapter 5. As shall be seen, maintenance of these properties helps prevent different attacks on the provenance records and chain.

4.4 Example scenario

The mutual agreement signature scheme is better explained referring to a scenario as shown in Figure 4.3. Each user along the chain follows the provenance record creation steps discussed in Algorithm 1. The important events in the scheme are highlighted.

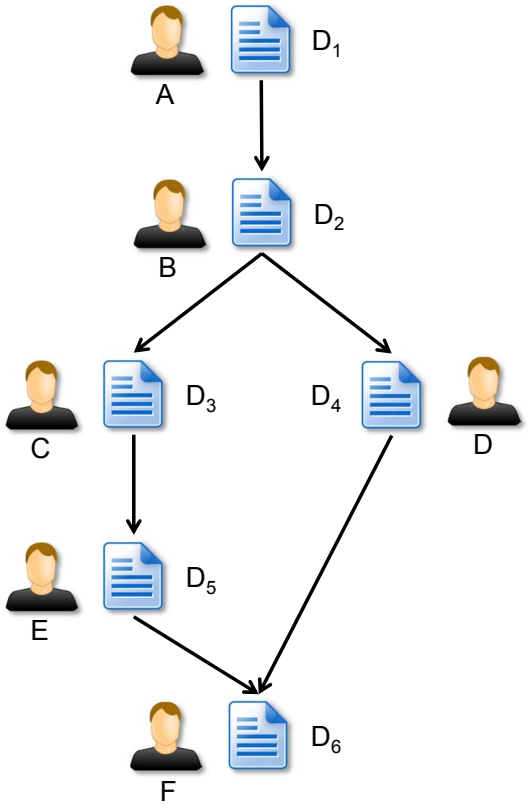

Figure 4.3. An example of document passing

The user A first creates the document D_1.

A is the creator of this document and the owner of its corresponding provenance chain. The provenance record Pr_A generated is:

$$Pr_A = <U_A, O_A, h(D_1), ChainInfo_A, S^*_A, C_A, P^+_A, N^*_A>$$
$$ChainInfo_A = K_{Aud} | K_A;$$

$$C_A = sign_A(U_A, O_A, h(D_1), S_A);$$
$$P_A = sign_{Aud}(IV, K_{Aud} | K_A, C_A)$$

The operations field O_A is empty, because A is the owner and creator of the document D_1. The next field N_A is currently empty, because A has not yet passed the document to another user. The field U_A can be encrypted using symmetric key K_{S_A}, and K_{S_A} can then be encrypted using K_{Aud} and stored in S_A. This preserves A's confidentiality. The previous field P_A is signed by the auditor and contains a unique initialization vector (IV) in the hash field known only to the auditor.

A passes the document D_1 to user B.

A new provenance record Pr_B is generated:

$$Pr_B = < U_B, O_B, h(D_2), ChainInfo_B, S_B^*, C_B, P_B^+, N_B^* >$$

It contains only the previous field P_B and $ChainInfo_B$ field at the beginning:

$$P_B = sign_A(h(D_1), K_{Aud} | K_A | K_B, C_A)$$
$$ChainInfo_B = K_{Aud} | K_A | K_B$$

The next field N_A of Pr_A is also created as:

$$N_A = sign_B(h(D_1), K_{Aud} | K_A | K_B, C_A)$$

As can be seen from these two fields, the signatures use the same data fields as agreement. It delineates the transition of responsibility of D_1 from A to B.

B now modifies the document D_1 to D_2.

B preforms operations O_B to modify the document. U_B and O_B can be encrypted using symmetric key K_{S_B}, and K_{S_B} can then be encrypted using K_{Aud} and stored in S_B. We ignore the encryption further in the example, since it remains the same for each record. The remaining fields in Pr_B are now updated. The current signature is of the form:

$$C_B = sign_B(U_B, O_B, h(D_2), S_B)$$

The next field N_B remains empty, because B has not yet passed the document to another user.

B passes the document D_2 to C.

Similar to the previous step, record Pr_C and the fields P_C, $ChainInfo_C$ and N_B are created:

$$Pr_C = <U_C, O_C, h(D_3), ChainInfo_C, S^*_C, C_C, P^+_C, N^*_C>$$
$$P_C = sign_B(h(D_2), K_{Aud}|K_A|K_B|K_C, C_B)$$
$$ChainInfo_C = K_{Aud}|K_A|K_B|K_C$$
$$N_B = sign_C(h(D_2), K_{Aud}|K_A|K_B|K_C, C_B)$$

B passes document D_2 to user D.

Again, the record Pr_D is created with the fields P_D and $ChainInfo_D$:

$$Pr_D = <U_D, O_D, h(D_4), ChainInfo_D, S^*_D, C_D, P^+_D, N^*_D>$$
$$P_D = sign_B(h(D_2), K_{Aud}|K_A|K_B|K_D, C_B)$$
$$ChainInfo_D = K_{Aud}|K_A|K_B|K_D$$

The next field N_B of Pr_B is now updated to include two subfields, to show the two separate provenance chains:

$$N_B = sign_C(h(D_2), K_{Aud}|K_A|K_B|K_C, C_B);$$
$$sign_D(h(D_2), K_{Aud}|K_A|K_B|K_D, C_B)$$

As can be seen, the data in the signatures is identical to that of the previous fields P_C and P_D which shows the separate agreements for the two chains.

Consider that C modifies document D_2 to D_3, and passes it to user E. The fields of record Pr_C are updated and Pr_E is created as:

$$C_C = sign_C(U_C, O_C, h(D_3), S_C)$$
$$N_C = sign_E(h(D_3), K_{Aud}|K_A|K_B|K_C|K_E, C_C)$$
$$Pr_E = <U_E, O_E, h(D_5), ChainInfo_E, S^*_E, C_E, P^+_E, N^*_E>$$
$$P_E = sign_C(h(D_3), K_{Aud}|K_A|K_B|K_C|K_E, C_C)$$

D modifies D_2 to D_4 and creates the fields of the record Pr_D and the current signature:

$$C_D = sign_D(U_D, O_D, h(D_4), S_D)$$

E modifies document D_3 obtained from C to D_5 and creates the fields of the record Pr_E and the current signature:

$$C_E = sign_E(U_E, O_E, h(D_5), S_E)$$

F obtains documents D_5 from E and D_4 from D.

In this case, the record Pr_F is created such that the previous field P_F has two subfields to indicate the two separate provenance chains. The next fields of the two users N_D and N_E are also updated:

$$Pr_F = <U_F, O_F, h(D_6), ChainInfo_F, S_F^*, C_F, P_F^+, N_F^*>$$

$$ChainInfo_F = K_{Aud}|K_A|K_B||K_B\text{–}K_C\text{–}K_E\text{–}K_F|K_B\text{–}K_D\text{–}K_F||$$

$$P_F = sign_D(h(D_4), K_{Aud}|K_A|K_B|K_D|K_F, C_D);$$

$$sign_E(h(D_5), K_{Aud}|K_A|K_B|K_C|K_E|K_F, C_E)$$

$$N_D = sign_F(h(D_4), K_{Aud}|K_A|K_B|K_D|K_F, C_D)$$

$$N_E = sign_F(h(D_5), K_{Aud}|K_A|K_B|K_C|K_E|K_F, C_E)$$

Again, it can be seen that the data used in the signatures for the subfields of P_F and the fields N_D and N_E are the same, indicating the separate agreements for the two chains. In the field $ChainInfo_F$, the sequences of public keys from the two branches is concatenated and the order is indicated by the – and || symbols. The order is also preserved in the two previous signature fields.

4.5 Auditing activity

An auditor performs auditing on the provenance chain to verify that the chain has not been tampered with. The steps followed by an auditor to verify the integrity of the provenance chain are discussed in Algorithm 2. Auditing takes place in reverse

Algorithm 2 Provenance chain auditing steps

1: $Auditor \leftarrow Pr_1|Pr_2|\ldots|Pr_n; D_n$ ▷ Auditor receives the provenance chain and document D_n
2: $H \leftarrow h(D_n)$ ▷ Auditor constructs hash of D_n
3: **Verify:** $Pr_n[h(D_n)] = H$
4: **for all** $Pr_i \in Pr_n|Pr_{n-1}|\ldots|Pr_1$ **do**
5: $W \leftarrow Decrypt_{K_i}(C_i)$ ▷ Verifying the current signature field
6: $X \leftarrow h(U_i, O_i, h(D_i), S_i)$
7: **Verify:** $W = X$
8: **if** $P_i = P_1$ **then** ▷ Previous field of record Pr_1 involves the auditor
9: $Y \leftarrow Decrypt_{K_{Aud}}(Pr_i[P_i])$
10: $Z \leftarrow h(IV, K_{Aud}|K_1, C_1)$
11: **Verify:** $Y = Z$
12: **end if**
13: **for all** $Pr'_j \in Pr_i[P_i]$ **do** ▷ For each previous field
14: $Y \leftarrow Decrypt_{K_j}(Pr_i[P_i^j])$ ▷ Decrypting previous signature field of Pr_i corresponding to Pr'_j
15: $Z \leftarrow Decrypt_{K_i}(Pr'_j[N_j^i])$ ▷ Decrypting next signature field of Pr'_j corresponding to Pr_i
16: **Verify:** $Y = Z$
17: **end for**
18: **end for**

order starting from record Pr_n up till the first record Pr_1. This is different from the verification in the Onion and PKLC schemes. The reverse order is used to be able to verify DAG structures where multiple individual chains may exist.

The auditor first checks if the hash of the current document matches the hash stored in the record Pr_n. If this verification fails, either the document or the record Pr_n has been tampered with. For each record along the chain, the auditor must

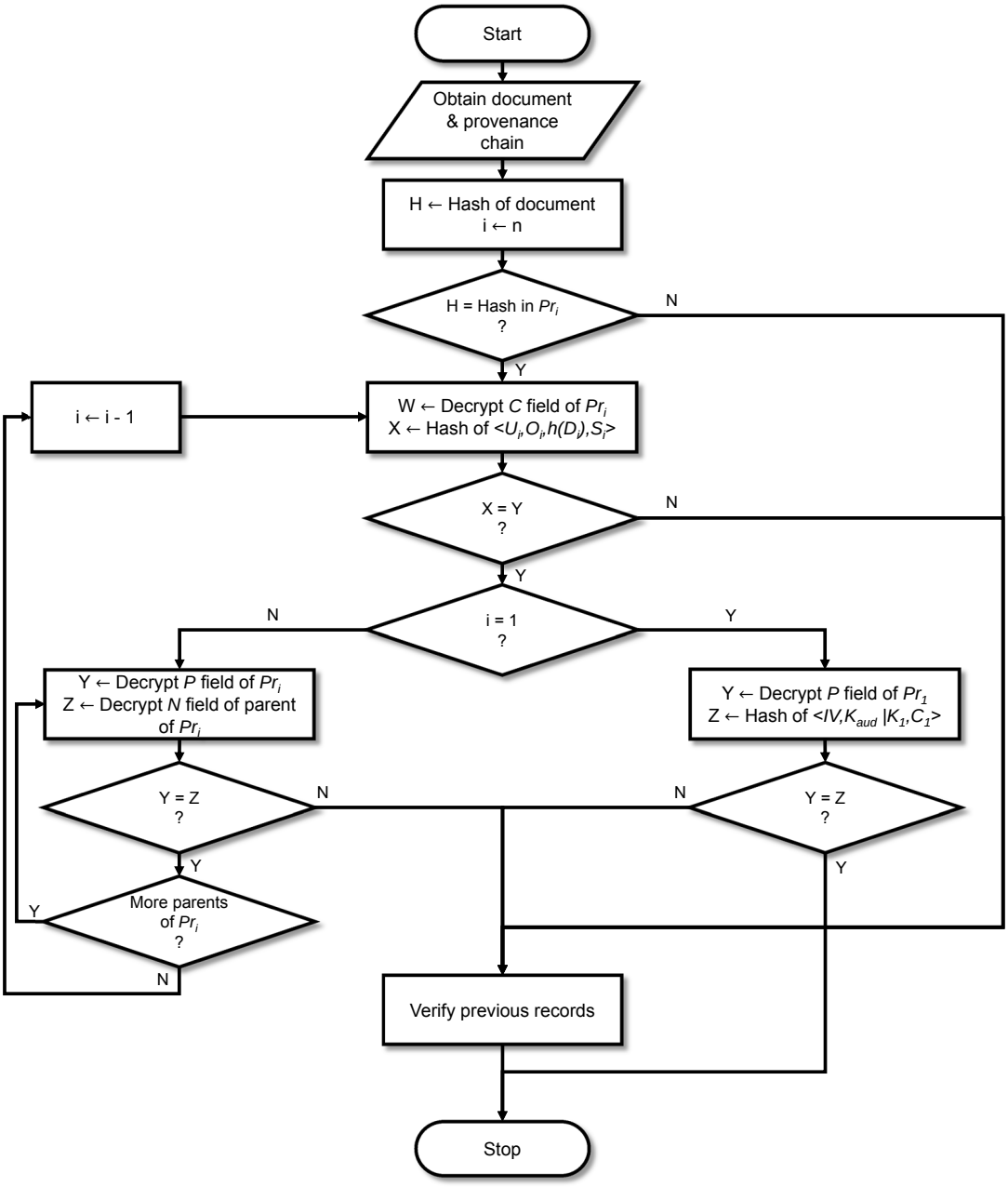

Figure 4.4. Flowchart of a auditor's actions for auditing a provenance chain

check if the current record is intact and that the agreements with the previous and next users are intact. For checking the current record, the auditor computes the

hash of the fields $< U_i, O_i, h(D_i), S_i >$ in the record. Then he checks this hash with the current signature C_i of the record. The agreements between pairs of records are checked by verifying the previous and next signature fields along the chain. To do this, the previous field P_i of every record is checked with the next field N_{i-1} of the previous record. The verification is done by computing the agreement i.e. the hash of the fields $< h(D_{i-1}), ChainInfo_i, C_{i-1} >$ and checking it with the stored signatures in the fields P_i and N_{i-1}.

If the current field verification fails for record Pr_j, there can be two cases:

i) if the agreement between Pr_j and Pr_{j-1} is intact, the record Pr_j has been tampered with, or,

ii) if this agreement is not intact, then the current fields and agreements of previous records must be checked till a record Pr_k is found where the current field verification fails, but agreement is intact. In such a case that record has been tampered with.

Similarly, if the agreement verification between records Pr_j and Pr_{j-1} fails, the auditor must check the current fields and agreements of previous records till a record Pr_k is found where the current field verification fails, but agreement is intact. In such a case that record has been tampered with.

If there is a verification failure at any stage, the auditor (or set of auditors with appropriate symmetric keys of the records) can decrypt the operations fields O_i of the records to reconstruct the document backwards to its original form. This is possible since O_i contains a reversible representation of the operations performed on the document. With this, each of the verifications can be performed again, to identify the exact instance in the history when the document or its record was tampered with. Any user who can verify the initialization vector used in the previous signature field of the first record can successfully perform auditing of the complete chain. Other users can verify the integrity of the chain beginning from the second record. However, only

an auditor can decrypt the symmetric keys stored in the record to further decrypt the U and O fields.

Figure 4.4 shows a flowchart of the steps followed by the auditor for auditing the provenance chain. The steps are similar to our discussion here. However, for the sake of simplicity of the flowchart, when any verification fails, the auditor enters the state 'Verify previous records' in which the verification of the remaining records up along the chain is performed to identify the cause of the verification failure.

In our scheme, each record is related only to its previous and next records. Unlike the Onion scheme, where every record contains the checksum information of the complete preceding chain, our scheme involves agreements with only pairs of records. Although Algorithm 2 involves checking the nodes sequentially from the last to the first, the verification can be conducted concurrently for pairs of records along the chain similar to the PKLC scheme. If all records are intact, this improves the verification process. If the verification fails for a particular pair of records, the preceding chain can then be investigated following the process described.

4.6 Advantages of our scheme

We can see that the provenance records in our scheme are linked together into a chain (actually DAG) structure through the previous and next signature fields which serve as the mutual agreements between records. The scheme is simple to implement and has many advantages over the schemes discussed in Chapter 3. A summary of these advantages is given in Table 4.1.

First, it provides better protection than the Onion scheme against selective removal of provenance records from the chain. In the Onion scheme, records are loosely linked only in the forward direction, and a record does not contain any information about who the next or previous user(s) is. So, if an adversary receives a chain of records $Pr_1|Pr_2|\ldots|Pr_n$, he/she can selectively remove part of the chain $Pr_i|\ldots|Pr_n$, append a new record Pr_i with the signature over the signature of record Pr_{i-1} and

remain undetected. Our scheme overcomes this drawback by introducing next and previous signature fields which cannot be forged. A record Pr_i contains the signatures of users U_{i-1} and U_{i+1}, thus selectively removing a part of the chain as in the case of the Onion scheme will cause the record Pr_{i-1} to still contain the signature of the original user U_i. Every user has proof of his/her actions, as well as the actions of the previous user, and contains information about the next user who the document gets passed to. This case is further explained in Chapter 5.

Second, our scheme does not require a trusted auditor(s) to know the user-key relationship as in the Onion scheme. This maintains the confidentiality of the users, and also makes the auditor a simpler entity. In fact, any user who can verify the initialization vector used in the previous signature field of the first record can successfully perform auditing of the complete chain. Other users can verify the integrity of the chain beginning from the second record. However, only an auditor can decrypt the symmetric keys stored in the record to further decrypt the U and O fields.

Third, our scheme can also support the DAG structure of provenance. As seen in the provenance record structure, each record contains previous P^+ and next N^* signature fields. Multiple fields signed by different users can be incorporated in these as indicated by the $^+$ and * symbols. Thus, our scheme can support DAG structures as part of the record structure itself. While provenance storage is not our concern in this work, we mention that storing a DAG structured provenance as a linear sequence of records (not linearly linked) would require a topological sorting mechanism [54].

The PKLC scheme requires auditors to know the sequence of users in the information flow. Every node also knows the next node to which the data must be passed. This allows the scheme to link the records only using the public keys of the users in the chain. But it also restricts the scheme to the scenario where the sequence of nodes is predetermined. If the sequence of nodes is dynamic, the auditor in the PKLC scheme cannot verify the integrity of the chain. Also, if the PKLC scheme is applied directly to a dynamic scenario, it is susceptible to successful tampering of records without detection. Unlike the PKLC scheme, our scheme can be applied to

Table 4.1
Summary of advantages of the Mutual Agreement Signature scheme over the Onion and PKLC schemes

Scheme	Support for DAG structures	Support for dynamic information sharing	Links between records	Link of a record to previous and/or next record
Onion scheme [29]	✗	✓	Weak because of incremental signatures	✗
PKLC scheme [30]	✓	✗	Only public keys used (weak for dynamic information sharing)	Public keys of previous (optional) and next users
Mutual Agreement Signature Scheme	✓	✓	Strong because of mutual signatures	Signatures of previous and next users on agreements

a dynamic information sharing scenario. This is made possible by the mutual agreements between users which contains enough information for an auditor to verify the sequence of users and integrity of the chain without knowing the sequence beforehand.

In the PKLC scheme, the first record contains the public key of the creator of the document in the previous field. If applied to a general scenario, this weak scheme is susceptible to an owner forgery attack, where an adversary can remove the records and claim to be the creator of the document by making a record with his/her own public key in the previous field. Our scheme prevents this by having the auditor sign the previous field of the first record. It also involves using a unique initialization vector IV that only the auditor knows. This ensures that an adversary cannot claim to be the owner of a document without having the auditor's signature and using a fake IV.

In the Onion scheme and the PKLC scheme, the corresponding current signature field signs over all the fields of the record. In our scheme, however, the current signature field only involves the fields $< U_i, O_i, h(D_i), S_i >$. It does not sign over *ChainInfo*. This is because *ChainInfo* is protected by the P and N fields. The mutual agreements protect the information about the sequence of users. This reduces the overhead of signing over all fields of the record. The current field is, thus, concerned only with the fields specific to that particular record.

We conclude this chapter by briefly summarizing the discussion. The mutual agreement signature scheme is an interactive protocol between users along the chain of custody of the document. We discussed the structure of the provenance chain and record, illustrated it with an example, described the auditing activity and mentioned the advantages of our scheme over existing works. We shall now analyze the security provided by our scheme in the next chapter.

5 SECURITY ANALYSIS

In this Chapter, we discuss how our scheme prevents the attacks from adversaries discussed in Section 2.3. We discuss propositions of attacks from an adversary and give a discussion about how our scheme prevents them.

The following proposition considers the attack on the **confidentiality** of the provenance:

Proposition 1. *An auditor can verify the integrity and availability of the chain but not access any of the confidential information in the records.*

An auditor only requires access to the $<h(D_i), ChainInfo_i, C_i, P_i^+, N_i^*>$ fields of every record in the chain to perform the auditing. The user information U and operations O fields are encrypted using a unique symmetric key K_s generated for the record. Multiple copies of this key are then encrypted each with the key of a different trusted auditor. The S field in the provenance record stores the encrypted versions of the symmetric keys. This encryption ensures that the information in the U and O fields is only accessible to the owner of the provenance record and the auditor(s) he/she trusts. No other entity should be able to access them.

The following propositions consider the attacks on the **integrity** of the provenance records and chain:

Proposition 2. *An adversary cannot claim that a valid provenance chain belongs to another document having different contents.*

Each record Pr_i contains a field $h(D_i)$ that stores the cryptographic hash of the document corresponding to that record. Also, the user U_i signs $h(D_i)$ along with other fields in the current C_i signature field, giving an account of his/her actions. Suppose an adversary U_i takes the provenance chain of document D_1, attaches it to another document D_2 and passes it to user U_j, there can be three scenarios:

i) U_j will verify that the $h(D_2) \neq h(D_1)$ from the last record in the chain. Thus, U_j will know that the provenance chain does not belong to D_2.

ii) If U_i changes $h(D_1)$ in the last record to correspond to $h(D_2)$, but the record does not belong to U_i, it is not possible for U_i to forge the current signature field C of the last record. This will be detected by U_j.

iii) If the last record does belong to U_i, and U_i can change the hash and current signature fields of the last record to correspond to document D_2, it will not be possible for U_i to forge the records in the preceding chain. This will be detected in the auditing activity.

Thus, it is not possible for an adversary to associate a provenance chain with another document.

Proposition 3. *An adversary cannot alter the fields of records from the preceding chain without being detected.*

Every record Pr_i in the chain contains a field for the current signature C_i which establishes that the signer U_i with private key K_i^- is the creator of that record. This signature cannot be forged by another user. Thus, even if an adversary alters the fields of a record Pr_i, the signature of the fields $<U_i, O_i, h(D_i)>$ will not match C_i when auditing is performed. Thus, any alteration of the fields in the preceding chain can be caught in the auditing activity.

Proposition 4. *An adversary cannot add fake records into the beginning or middle of the chain without being detected.*

Each record in the provenance chain has previous P_i and next N_i signature fields. These, along with the *ChainInfo* field, represent the chain of successive custody of the document.

The first record of the owner of the document has the previous field signed by the auditor using the key K_{Aud}^-. Since this signature cannot be forged, an adversary

cannot add fake records into the beginning of the chain since the signature will not be that of the auditor. If the adversary attaches the first record of another chain to the beginning of this chain, the case is similar to Proposition 2.

If fake records are added to the middle of the chain, the previous and next fields must by signed by the appropriate users of the chain to maintain the order of the mutual agreements. Consider the chain contains consecutive records Pr_i and Pr_j. Then N_i is signed by user U_j and P_j is signed by user U_i. If an adversary wants to add a record Pr_k between Pr_i and Pr_j, N_i must be signed by U_k and P_k signed by U_i. Similarly, N_j must be signed by U_k and P_j signed by U_k. This is not possible since the signatures of U_i and U_j cannot be forged.

Proposition 5. *An adversary cannot remove records from the beginning or middle of the chain without being detected.*

Arguments similar to those in Proposition 4 hold for the case when records are removed from the beginning or middle of the chain. If the first record is removed, the new first record must have the previous field signed by the auditor. Since signatures cannot be forged, it is not possible for the new first record to have the signature of the auditor. Suppose we have the records Pr_i, Pr_j and Pr_k as consecutive records in the chain. If an adversary removes the record Pr_j, in the new chain N_i must be signed by user U_k and P_k signed by U_i. Similarly, N_j must be signed by U_k and P_j signed by U_k. This is again not possible since signatures cannot be forged. Thus, it is not possible for the adversary to remove records without being detected in the auditing.

Proposition 6. *An adversary cannot alter the order of records from the preceding chain without being detected.*

This argument is a combination of Propositions 4 and 5. Altering the order of records can be thought of as multiple operations of removal and addition of records into the provenance chain. The P and N signature fields of the altered records will not have the appropriate signatures according to the sequence of users in the chain.

Proposition 7. *An adversary cannot remove all the records of the chain and claim he/she is the owner of the document by creating a new chain.*

The first record of the owner of the document has the previous field signed by the auditor using the key K^{-}_{Aud}. Since this signature cannot be forged, an adversary cannot remove all preceding provenance records and claim to be the owner of the document.

Proposition 8. *Two colluding adversaries already having records in the chain cannot add or remove records of other users between their records.*

Consider a provenance chain $Pr_1|\ldots|Pr_i|\ldots|Pr_j|\ldots|Pr_n$. Two colluding adversaries U_i and U_j may want to remove records between their records in the chain, such that the resulting chain is $Pr_1|\ldots|Pr_i|Pr_j|\ldots|Pr_n$. To be successful, the records Pr_i and Pr_j must have appropriate signatures in the previous and next fields, which is possible in the case of collusion. However, the records must also be updated such that the hashes of the documents at each stage and the operations fields O_i satisfy the reversibility criteria. If O_i is encrypted, this will become difficult to achieve. The case is more difficult when adversaries want to add records of other users between their records. In such a case, the signatures in all the fields must align, which can again be difficult to achieve.

Proposition 9. *An adversary cannot repudiate any records in the provenance chain.*

Each user U_i in the chain must use his/her private key K^{-}_{i} to sign the current field C_i of the provenance record Pr_i. This field serves as the digital signature that no other user can forge. The signature is on the data fields U_i, O_i and $h(D_i)$ which essentially represent the actions performed and results obtained by the user. If the record Pr_i is not the last record, the user U_i must use the same key K^{-}_{i} to sign the previous field P_{i+1} of the next record Pr_{i+1}. If the user U_i is not the owner of the chain, the next field N_{i-1} of the record U_{i-1} must also be signed in this way. The involvement of user U_i in the chain is captured not only in the record Pr_i, but also in the previous and

next records Pr_{i-1} and Pr_{i+1} respectively. Thus, after user U_i has created record Pr_i, he/she cannot repudiate the record.

The following proposition considers the attack on the **availability** of the provenance records and chain:

Proposition 10. *An adversary cannot modify the document without adding the appropriate provenance record in the chain and not being detected.*

The mutual agreement procedure is followed when the document is passed from user U_i to U_{i+1}. Thus, if the user U_{i+1} modifies the document without appending the appropriate record in the chain, the record Pr_i still contains the signature of user U_{i+1}. Thus the involvement of U_{i+1} cannot be repudiated.

Also, every provenance record Pr_i stores the operations O_i performed by the user. This contains a reversible representation of the operations, such that if document D_{i-1} is modified to D_i, D_{i-1} can be obtained from D_i using O_i. Also, record Pr_{i-1} stores the hash $h(D_{i-1})$. If an adversary performs operations on D_i that are not recorded in O_i or modifies D_i and does not add an appropriate record to the chain, the previous document D_{i-1} cannot be recovered such that its hash is the same as $h(D_{i-1})$ stored in record Pr_{i-1}. Simple auditing would not require the document to be reconstructed in reverse; however it can be done to resolve discrepancies.

6 IMPLEMENTATION & EVALUATION

Provenance generation results in an overhead in time as well as space that must be considered apart from the security it provides. In this Chapter, we give an evaluation of the overhead of our scheme and compare it with the Onion and PKLC schemes.

We implemented the three schemes with Java JDK 1.7 to measure and compare their performances. The programs were executed using NetBeans IDE 7.4 on a Windows 7 machine with Intel Core 2 Duo E6550 2.33GHz processor and 4.0 GB main memory. Since the provenance transmission and storing are not our concern in this work, we simulate the users on the same machine and simulate their activities by modifying the document each time it is passed between users. Since our focus is to measure the performance of the provenance generation process and not have it affected by file I/O, we store the provenance records as nodes in a linked list in memory instead of generating an XML document. We use 1024 bit RSA for digital signatures and well as asymmetric encryptions, 128 bit AES for symmetric encryptions, and SHA-1 as our hashing function. We choose four different files 5Kb (text document), 50Kb (text document), 500Kb (JPEG image) and 5Mb (JPEG image) to compare the performance against varying file sizes.

The overhead for creating and auditing the provenance chain can vary depending on the data included in each of the provenance records. We run our experiment under the following settings:

i) Each user appends a small amount of information to the file as an operation on it. This ensures that the file size does not change significantly while its hash value does. Since the actual operations performed on the document are not our concern, this suffices for our experiment.

ii) Each user trusts the same auditor, and thus the symmetric key used to encrypt the U and O fields, is encrypted using the public key of only one auditor.

iii) For the purpose of the experiments, every user operates on the data only once, however, our scheme allows a user to be repeated in the provenance chain.

6.1 Provenance Record Construction

In the first experiment, we compare the performance of the three schemes for the process of creation of a provenance record. We consider two aspects of the record construction: the overall time required to construct a record and, the overhead of individual operations involved in each scheme.

6.1.1 Time to create a record and overhead over other schemes

We measure the time required by each of the schemes to create a single provenance record. We run this experiment 50 times for each of the varying file sizes and report the average computation time. Figure 6.1 shows the results of the experiment. As expected from the experiments in [30], the PKLC scheme performs better than the Onion scheme. Our Mutual Agreement signature scheme takes longer to construct a record.

The Onion and PKLC schemes each uses two hash operations (one for the file and the other for the current signature field). Our scheme on the other hand uses four hash operations (two additional for the previous and next fields). It also uses RSA signatures for each of the agreements. As a result, our scheme has a comparatively larger overhead of 97% over the PKLC scheme and 74% over the Onion scheme. However, as the file size increases, we observe that the overhead reduces and becomes comparable to the performance of the PKLC (7.18% overhead) and Onion schemes (3.22% overhead). Figure 6.2 shows the percentage overhead of our scheme over the Onion and PKLC schemes.

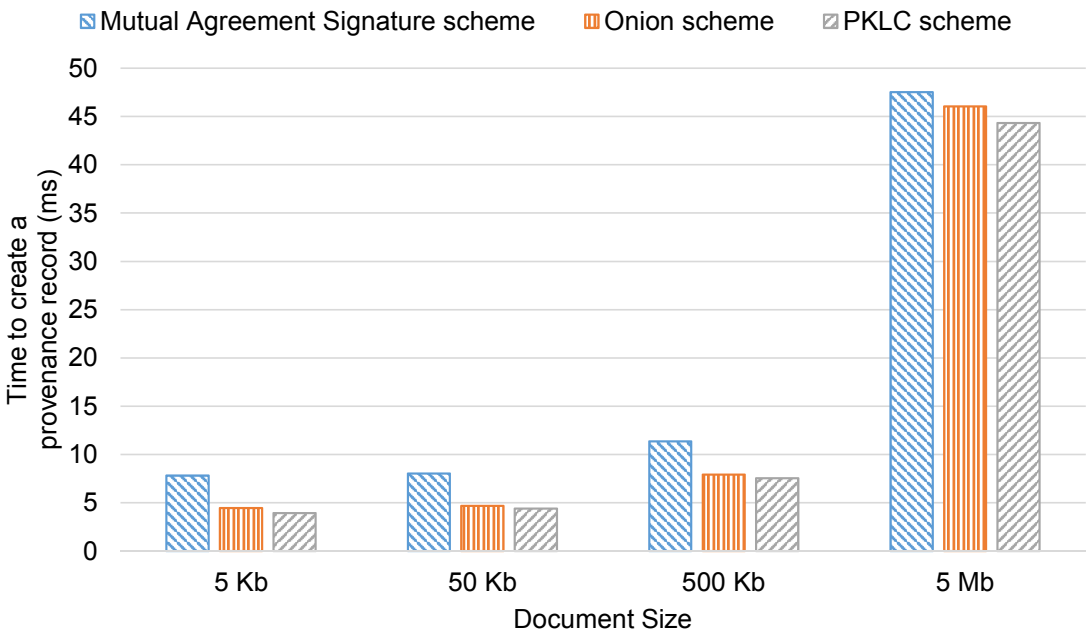

Figure 6.1. Time to create a provenance record in all three schemes

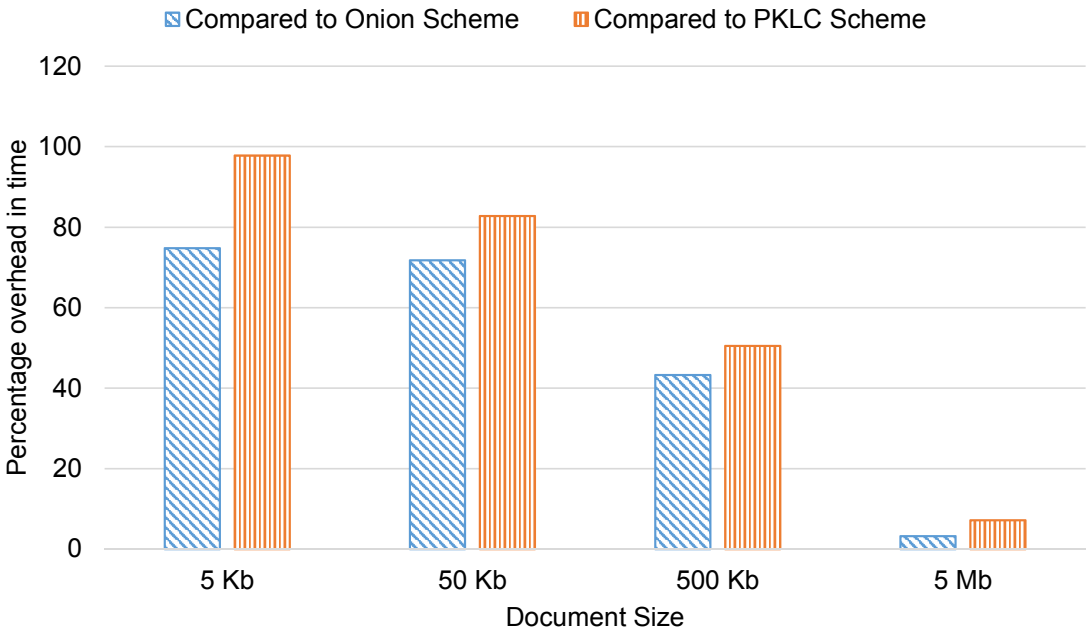

Figure 6.2. Percentage overhead of Mutual Agreement Signature scheme over the Onion and PKLC schemes

6.1.2 Overhead of individual operations of the schemes

We observe that the overhead of our scheme reduces as the file size increases and becomes comparable to the performance of the Onion and PKLC schemes. The time required by each individual operation performed in record creation we measured to understand this. Each of the scheme uses the following operations in record creation:

i) RSA signature: The C, P and N fields each use RSA signatures.

ii) AES encryption: The U and O fields are encrypted with AES using a symmetric key.

iii) RSA encryption: The symmetric key is then encrypted using RSA with the key of the auditor.

iv) File Hash: This is the SHA-1 hash of the document.

v) Field Hash: This is the SHA-1 hash used before the signature can be applied to the C, P and N fields.

vi) Self Time: The process of record creation also involves minor operations other than the cryptographic operations. They are accounted for as self time of the record generation process.

We find that the performances become comparable because, as the file size increases, the overhead of the file hash operation alone dominates over the overhead of the other operations. This is illustrated by the Figures 6.3, 6.4 and 6.5 which indicate the time required by each individual operation for different file size for the three schemes.

From the figures, it is also seen that the RSA signature takes a larger percentage of the total time in our scheme. This is because our scheme uses RSA signatures for the mutual agreements. This operation is not present in the other schemes.

We argue that the security provided by our scheme and support for dynamic information sharing outweighs the overhead of our scheme.

Figure 6.3. Percentage of time for each operation in record creation for the Mutual Agreement Signature scheme

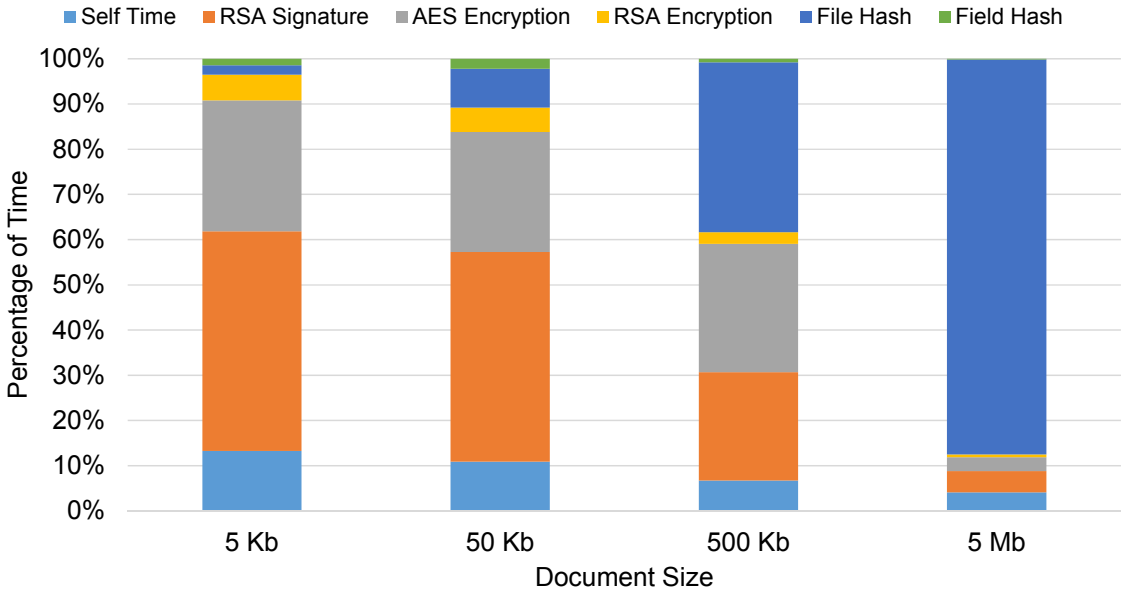

Figure 6.4. Percentage of time for each operation in record creation for the PKLC scheme

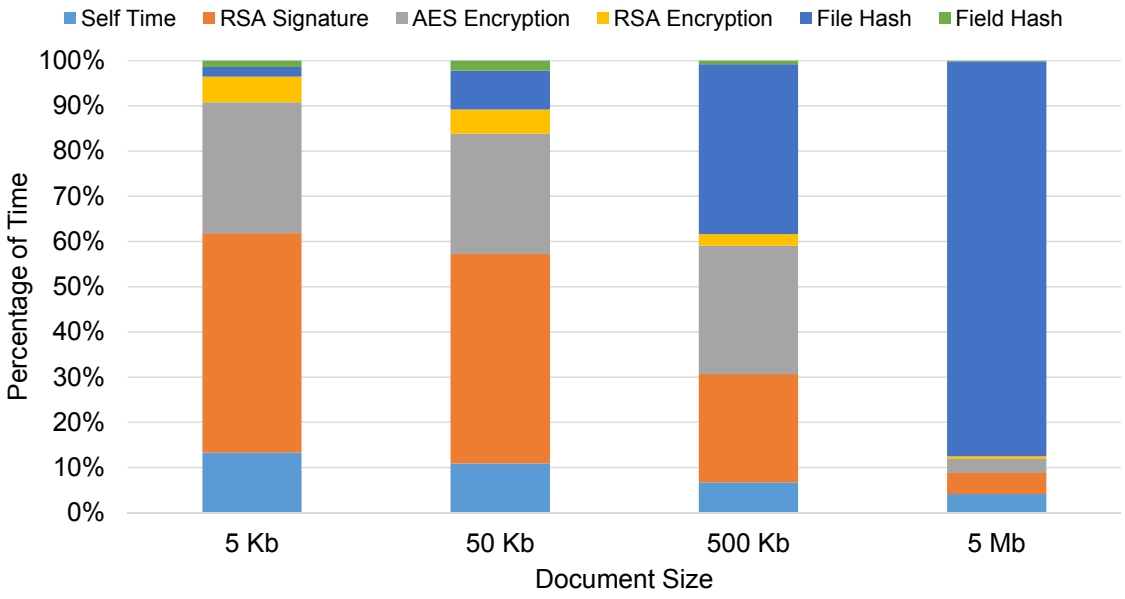

Figure 6.5. Percentage of time for each operation in record creation for Onion scheme

6.2 Provenance Chain Auditing

In the next experiment, we measured the time required for an auditor to audit an entire provenance chain. We run this experiment for files of size 50Kb and 5Mb, each for varying number of records in the chain. Each complete chain verification process is run 10 times and we report the average computation time required. We first generate a linear chain of provenance records by using the same process as in the previous experiment and then perform verification. Since the Onion scheme does not support DAG structures, we experiment with a linear chain to compare the performances of the three schemes.

6.2.1 Concurrent auditing over the chain

In the Onion scheme every record contains the checksum information of the complete preceding chain and the verification must take place sequentially along the chain.

In the Mutual Agreement Signature scheme and the PKLC scheme, each record is related only to its previous and next records. Our scheme involves agreements with only pairs of records. Since it does not require an auditor to go sequentially over the complete chain from the start, the verification can take place concurrently for pairs of records along the chain. We use mutliple auditor threads to verify pairs of records along the chain for our scheme and the PKLC scheme.

Figure 6.6 shows the results of auditing the provenance chain for a file of size 50Kb run on chains with increasing numbers of records. Our scheme takes longer to verify than the PKLC scheme since it employs two more hashes. However, it performs better than the Onion scheme, since we do not need to go sequentially over the chain. For the 50Kb file, the improvement in performance over the Onion scheme becomes significant as the length of the chain increases. The difference grows slower for the 5Mb file as seen in Figure 6.7. This is because the hash of the file again dominates over the time required by other operations. Since the verification for our scheme and the PKLC scheme happens concurrently over different records, it would be expected that the time required remains fairly constant despite the length of the chain. However, we see an increase in the time in both experiments. We attribute this to the overhead of creating and managing threads in Java.

6.2.2 Sequential auditing over the chain

In the previous experiment, we performed the auditing concurrently for pairs of records along the chain for the PKLC scheme and our scheme. In order to compare the performances of the three schemes in the same setting, we run a similar experiment performing auditing sequentially over the chain with only one auditor thread.

Figure 6.8 shows the result of auditing the provenance chain for a file of size 50Kb run sequentially on the chains with increasing number of records. Our scheme takes longer to verify than the Onion and PKLC schemes. This is expected, since our scheme employs two hashes more than the other schemes. This operation dominates

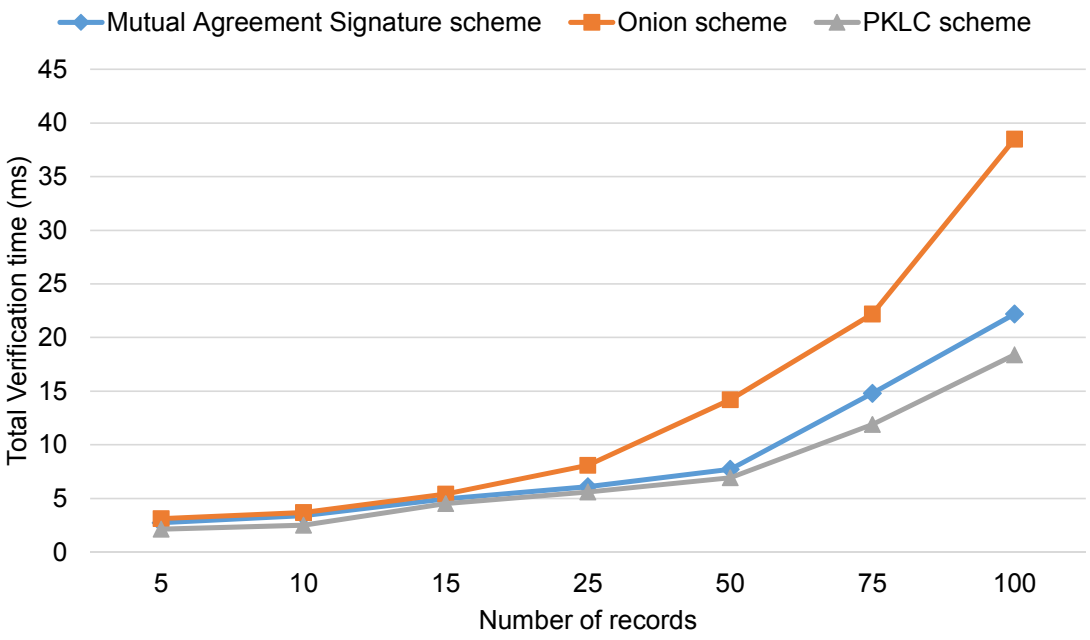

Figure 6.6. Provenance chain verification time in all three schemes for file size 50Kb (using concurrent verification)

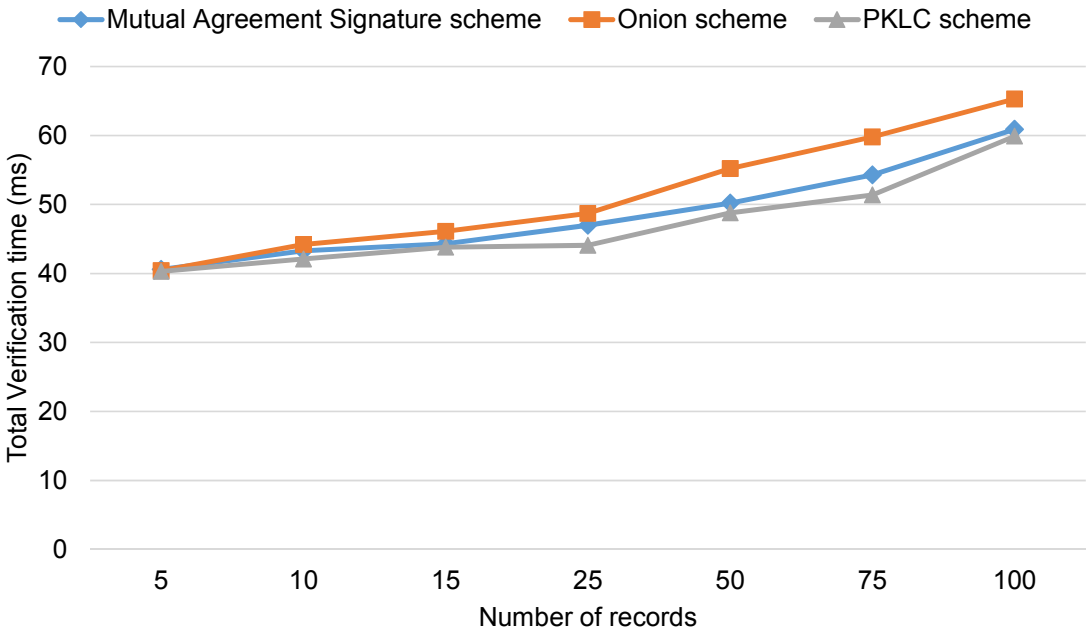

Figure 6.7. Provenance chain verification time in all three schemes for file sizes 5Mb (using concurrent verification)

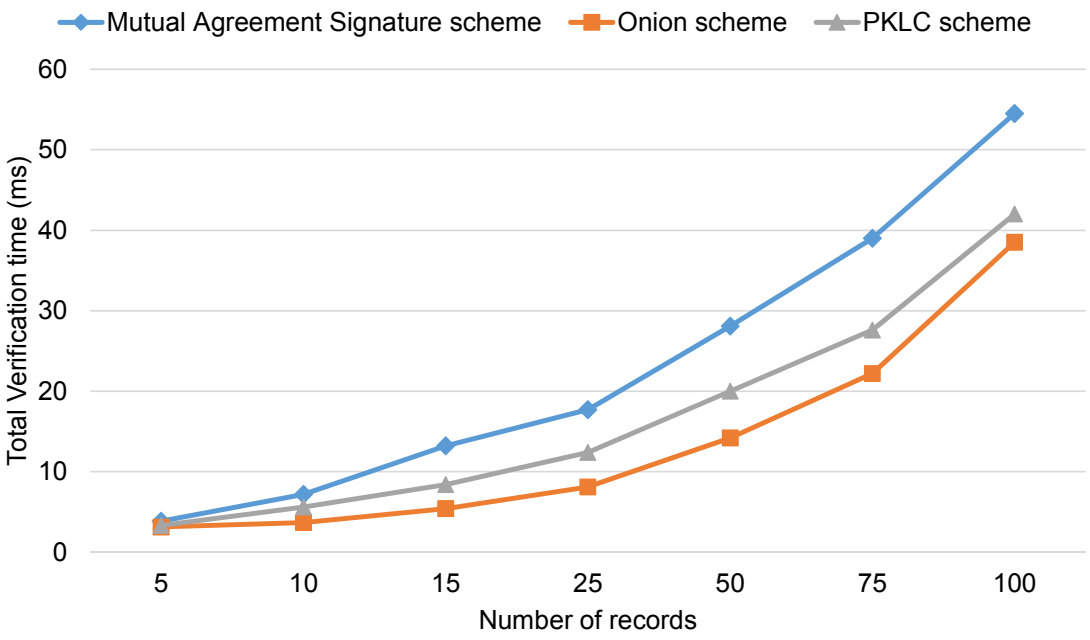

Figure 6.8. Provenance chain verification time in all three schemes for file size 50Kb (using sequential verification)

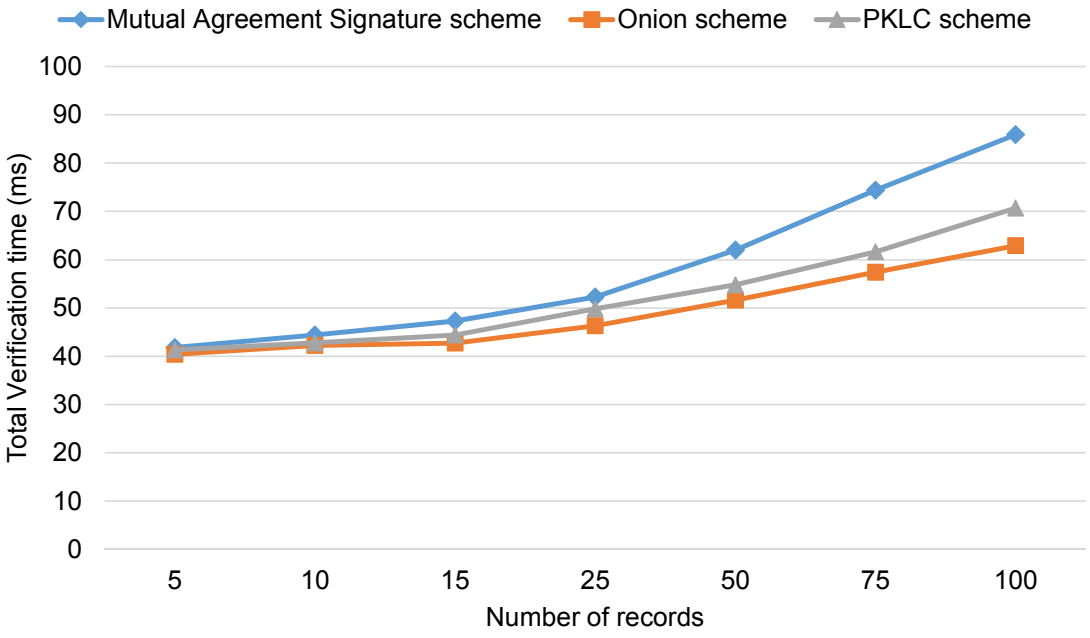

Figure 6.9. Provenance chain verification time in all three schemes for file sizes 5Mb (using sequential verification)

over the other operations while performing auditing. In the previous experiment, the overhead of the extra hash operations is reduced since they are performed concurrently. We also observe that the PKLC scheme also takes longer than the Onion scheme in the similar setting. This is because the PKLC scheme needs to verify the public keys stored in the records in addition to the signature field. In the sequential auditing, this operation introduces the overhead which is not present in the Onion scheme. The difference in the times grows larger as the number of records increases. This is expected since two hash operations per pair of records increases. Figure 6.9 shows the same experiment run for a file of size 5Mb. The different in time grows slower, since the hash of the file again dominates over the time required by the other operations. However, we see an increase in the time in both experiments.

6.3 Argument of security over performance

It can be seen from the experiments and our evaluation that our scheme compromises on the performance as compared to the Onion and PKLC schemes. As the file size increases, the performance for provenance record generation becomes comparable to that of the other schemes. For provenance verification, the performance is better than that of the Onion scheme. The difference in performance is due to the operations performed for creating the previous and next signature fields. These fields are the basis of the mutual agreement scheme. As discussed in Section 4.6, they overcome the shortcomings of the other schemes and provide better security to the provenance chain. We thus argue that the advantages of our scheme i.e. providing a representation for DAG structures of provenance, better security and supporting dynamic information sharing between users, outweighs the performance overhead seen in Figures 6.1 and 6.2 for record construction, and in Figures 6.8 and 6.9 for sequential provenance chain auditing.

We conclude this chapter by briefly summarizing the discussion on the experimentation. The performance of our scheme against that of the Onion and PKLC schemes

was evaluated, for the cases of provenance record construction and chain auditing. Our scheme performs better that the Onion scheme for the case of concurrent chain auditing. We observe an expected performance overhead for record construction and linear chain auditing due to the additional operations involved in our scheme, but argue that the security provided outweighs this overhead.

7 CONCLUSION & FUTURE WORK

Provenance is meta-data that stores the history of an object and it has unique security requirements. Since history of an object cannot be changed, once its provenance is created, the integrity and availability of the provenance records and the confidentiality of user information contained in them must be protected. In this work, we proposed a signature-based mutual agreement scheme that delineates the transition of responsibility of a document when it is passed from one user to the other. We investigated dynamic information sharing and representation of DAG structures of provenance, which are two aspects not thoroughly looked at in recent research. The provenance format and verification process in our scheme was discussed and an analysis of its security assurance was provided. We gave experimental results of the overhead of our scheme for provenance creation and verification. The overhead for provenance creation is a little more than that of the PKLC and Onion schemes, but it provides better security, whereas our scheme performs better than the Onion scheme for chain verification. Section 7.1 briefly discusses some application scenarios for our scheme and Section 7.2 talks about some of the future work related to this research.

7.1 Applications scenarios for our scheme

The mutual agreement signature scheme proposes a structure for the provenance record and chain for a digital object. We use generality while defining the fields in the record in order to support different types of applications. As discussed before, user information and operations on the object are application specific and they may have different representations depending on the domain. However, our scheme can support these since the fields are generally defined to work for any kind of representation. Also, our scheme captures the properties, such as DAG representation,

integrity of individual records and the complete chain, which are common to all forms of provenance. We discuss some of the application scenarios here.

The scheme can be applied to employee review and recommendation systems. A document is passed between different employees, managers, teachers, etc. at different security levels and hierarchy. The document and provenance may have different sensitivity depending on the application and our scheme can support these scenarios. We club all such application scenarios as academic information sharing. Another scenario is a distributed application running instances on multiple nodes such that they incrementally process the data moving towards a sink. This is similar to a wireless sensor network, but encompasses a wider range of applications that can also include industrial pipelined processes. Our scheme can be applied for provenance generation in work-flows of industrial processes and grid based computing applications. In such scenarios, users maybe computer processes, sensor nodes or even industrial mechanized processes, which can have different representations in the provenance records. Wireless sensor networks are typically resource constrained and since our scheme employs computationally expensive operations, it may not be best suited for wireless sensor networks.

7.2 Future Work

Typical provenance schemes similar to our work in this thesis depend on transitive trust among users. A challenge that they face is the detection of collusion of users that are successive in the chain. In our future work we will look into a solution for detecting collusion of successive users. Another aspect is that our scheme requires a trusted auditor to prevent owner forgery cases and to access confidential information of provenance records. One direction of research is to reduce/remove the involvement of the trusted auditor and develop the scheme such that the users are the only entities involved. One possible way is to use Hierarchical Access Control (HAC) schemes such as [55] for efficient key management. In such a scheme, the creator of the document

is at the top of the hierarchy. This can remove the Public Key Infrastructure (PKI) required by most existing provenance schemes. However, it also introduces challenges of maintaining the desired security properties for the records and the chain. We will also work towards reducing the overhead of our scheme even further, by either employing faster algorithms, or by changing the procedure for mutual agreements. In addition, we will implement our scheme in a real world information sharing application, similar to ones discussed in the previous section and carry out a more thorough security and performance analysis.

LIST OF REFERENCES

LIST OF REFERENCES

[1] Yogesh L. Simmhan, Beth Plale, and Dennis Gannon. A survey of data provenance in e-science. *SIGMOD Record*, 34(3):31–36, September 2005.

[2] Luiz M. R. Gadelha Jr and Marta Mattoso. Kairos: An architecture for securing authorship and temporal information of provenance data in grid-enabled workflow management systems. In *Proceedings of the 2008 Fourth IEEE International Conference on eScience*, ESCIENCE '08, pages 597–602, Washington, DC, USA, 2008. IEEE Computer Society.

[3] R. Hasan. Protecting the Past and Present of Data, with Applications in Provenance and Regulatory-Compliant Databases. *Proceedings of the Third SIGMOD PhD Workshop on Innovative Database Research (IDAR 2009)*, 2009.

[4] James Cheney, Stephen Chong, Nate Foster, Margo Seltzer, and Stijn Vansummeren. Provenance: A future history. In *Proceedings of the 24th ACM SIGPLAN Conference Companion on Object Oriented Programming Systems Languages and Applications*, OOPSLA '09, pages 957–964, New York, NY, USA, 2009. ACM.

[5] Shouhuai Xu, Qun Ni, Elisa Bertino, and Ravi Sandhu. A characterization of the problem of secure provenance management. In *Proceedings of the 2009 IEEE International Conference on Intelligence and Security Informatics*, ISI'09, pages 310–314, Piscataway, NJ, USA, 2009. IEEE Press.

[6] Wang chiew Tan. Research problems in data provenance. *IEEE Data Engineering Bulletin*, 27:45–52, 2004.

[7] Rajendra Bose and James Frew. Lineage retrieval for scientific data processing: A survey. *ACM Computing Surveys (CSUR)*, 37(1):1–28, March 2005.

[8] Simon Miles, Sylvia C. Wong, Weijian Fang, Paul Groth, Klaus peter Zauner, and Luc Moreau. Provenance-based validation of e-science experiments. In *In ISWC*, pages 801–815. Springer-Verlag, 2005.

[9] Peter Buneman, Adriane Chapman, and James Cheney. Provenance management in curated databases. In *Proceedings of the 2006 ACM SIGMOD International Conference on Management of Data*, SIGMOD '06, pages 539–550, New York, NY, USA, 2006. ACM.

[10] James Cheney, Laura Chiticariu, and Wang-Chiew Tan. Provenance in databases: Why, how, and where. *Found. Trends databases*, 1(4):379–474, April 2009.

[11] Paolo Missier, Saumen Dey, Khalid Belhajjame, Víctor Cuevas-Vicenttín, and Bertram Ludäscher. D-prov: Extending the prov provenance model with workflow structure. In *Proceedings of the 5th USENIX Workshop on the Theory and Practice of Provenance*, TaPP '13, pages 9:1–9:7, Berkeley, CA, USA, 2013. USENIX Association.

[12] Fernando Chirigati, Dennis Shasha, and Juliana Freire. Reprozip: Using provenance to support computational reproducibility. In *Proceedings of the 5th USENIX Workshop on the Theory and Practice of Provenance*, TaPP '13, pages 1:1–1:4, Berkeley, CA, USA, 2013. USENIX Association.

[13] S. Jensen, B. Plale, M.S. Aktas, Yuan Luo, Peng Chen, and H. Conover. Provenance capture and use in a satellite data processing pipeline. *Geoscience and Remote Sensing, IEEE Transactions on*, 51(11):5090–5097, Nov 2013.

[14] M.R. Huq, P.M.G. Apers, and A. Wombacher. An inference-based framework to manage data provenance in geoscience applications. *Geoscience and Remote Sensing, IEEE Transactions on*, 51(11):5113–5130, Nov 2013.

[15] Nicholas E Taylor and Zachary G Ives. Reconciling while tolerating disagreement in collaborative data sharing. In *Proceedings of the 2006 ACM SIGMOD international conference on Management of data*, pages 13–24. ACM, 2006.

[16] Todd J. Green, Grigoris Karvounarakis, Zachary G. Ives, and Val Tannen. Update exchange with mappings and provenance. In *Proceedings of the 33rd International Conference on Very Large Data Bases*, VLDB '07, pages 675–686. VLDB Endowment, 2007.

[17] Todd J. Green, Grigoris Karvounarakis, Nicholas E. Taylor, Olivier Biton, Zachary G. Ives, and Val Tannen. Orchestra: Facilitating collaborative data sharing. In *Proceedings of the 2007 ACM SIGMOD International Conference on Management of Data*, SIGMOD '07, pages 1131–1133, New York, NY, USA, 2007. ACM.

[18] Zachary G. Ives, Todd J. Green, Grigoris Karvounarakis, Nicholas E. Taylor, Val Tannen, Partha Pratim Talukdar, Marie Jacob, and Fernando Pereira. The orchestra collaborative data sharing system. *SIGMOD Rec.*, 37(3):26–32, September 2008.

[19] John Ockerbloom. Copyright and provenance: Some practical problems. *IEEE Data Eng. Bull.*, 30(4):51–58, 2007.

[20] Ragib Hasan, Radu Sion, and Marianne Winslett. Introducing secure provenance: Problems and challenges. In *Proceedings of the 2007 ACM Workshop on Storage Security and Survivability*, StorageSS '07, pages 13–18, New York, NY, USA, 2007. ACM.

[21] Kiran-Kumar Muniswamy-Reddy, Peter Macko, and Margo Seltzer. Making a cloud provenance-aware. In *First Workshop on on Theory and Practice of Provenance*, TAPP'09, pages 12:1–12:10, Berkeley, CA, USA, 2009. USENIX Association.

[22] Kiran-Kumar Muniswamy-Reddy and Margo Seltzer. Provenance as first class cloud data. *SIGOPS Oper. Syst. Rev.*, 43(4):11–16, January 2010.

[23] 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.

[24] Boris Glavic, Javed Siddique, Periklis Andritsos, and Renée J. Miller. Provenance for data mining. In *Proceedings of the 5th USENIX Workshop on the Theory and Practice of Provenance*, TaPP '13, pages 5:1–5:5, Berkeley, CA, USA, 2013. USENIX Association.

[25] Nathaniel Husted, Sharjeel Qureshi, Dawood Tariq, and Ashish Gehani. Android provenance: Diagnosing device disorders. In *Proceedings of the 5th USENIX Workshop on the Theory and Practice of Provenance*, TaPP '13, pages 4:1–4:7, Berkeley, CA, USA, 2013. USENIX Association.

[26] Boris Glavic, Kyumars Sheykh Esmaili, Peter Michael Fischer, and Nesime Tatbul. Ariadne: Managing fine-grained provenance on data streams. In *Proceedings of the 7th ACM International Conference on Distributed Event-based Systems*, DEBS '13, pages 39–50, New York, NY, USA, 2013. ACM.

[27] S. Sultana, M. Shehab, and E. Bertino. Secure provenance transmission for streaming data. *Knowledge and Data Engineering, IEEE Transactions on*, 25(8):1890–1903, Aug 2013.

[28] Uri Braun, Avraham Shinnar, and Margo Seltzer. Securing provenance. In *Proceedings of the 3rd Conference on Hot Topics in Security*, HOTSEC'08, pages 4:1–4:5, Berkeley, CA, USA, 2008. USENIX Association.

[29] Ragib Hasan, Radu Sion, and Marianne Winslett. Preventing history forgery with secure provenance. *ACM Transactions on Storage (TOS)*, 5(4):12:1–12:43, December 2009.

[30] Xinlei Wang, Kai Zeng, K. Govindan, and P. Mohapatra. Chaining for securing data provenance in distributed information networks. In *Military Communications Conference, 2012 - MILCOM 2012*, pages 1–6, Orlando, FL, USA, Oct 2012.

[31] Paul Groth, Michael Luck, and Luc Moreau. A protocol for recording provenance in service-oriented grids. In *In Proceedings of the 8th International Conference on Principles of Distributed Systems (OPODIS04*, pages 124–139, 2004.

[32] Peter Buneman, Sanjeev Khanna, and Wang Chiew Tan. Why and where: A characterization of data provenance. In *Proceedings of the 8th International Conference on Database Theory*, ICDT '01, pages 316–330, London, UK, UK, 2001. Springer-Verlag.

[33] Kiran-Kumar Muniswamy-Reddy, David A Holland, Uri Braun, and Margo I Seltzer. Provenance-aware storage systems. In *USENIX Annual Technical Conference, General Track*, pages 43–56, 2006.

[34] Adriane P. Chapman, H. V. Jagadish, and Prakash Ramanan. Efficient provenance storage. In *Proceedings of the 2008 ACM SIGMOD International Conference on Management of Data*, SIGMOD '08, pages 993–1006, New York, NY, USA, 2008. ACM.

[35] Ryan K. L. Ko, Peter Jagadpramana, and Bu Sung Lee. Flogger: A file-centric logger for monitoring file access and transfers within cloud computing environments. In *Proceedings of the 2011IEEE 10th International Conference on Trust, Security and Privacy in Computing and Communications*, TRUSTCOM '11, pages 765–771, Washington, DC, USA, 2011. IEEE Computer Society.

[36] Salmin Sultana and Elisa Bertino. A file provenance system. In *Proceedings of the Third ACM Conference on Data and Application Security and Privacy*, CODASPY '13, pages 153–156, New York, NY, USA, 2013. ACM.

[37] Lucian Carata, Ripduman Sohan, Andrew Rice, and Andy Hopper. Ipapi: Designing an improved provenance api. In *Proceedings of the 5th USENIX Workshop on the Theory and Practice of Provenance*, TaPP '13, pages 10:1–10:4, Berkeley, CA, USA, 2013. USENIX Association.

[38] Devdatta Kulkarni. A provenance model for key-value systems. In *Proceedings of the 5th USENIX Workshop on the Theory and Practice of Provenance*, TaPP '13, pages 12:1–12:4, Berkeley, CA, USA, 2013. USENIX Association.

[39] Nikilesh Balakrishnan, Thomas Bytheway, Ripduman Sohan, and Andy Hopper. Opus: A lightweight system for observational provenance in user space. In *Proceedings of the 5th USENIX Workshop on the Theory and Practice of Provenance*, TaPP '13, pages 8:1–8:4, Berkeley, CA, USA, 2013. USENIX Association.

[40] Sherif Akoush, Ripduman Sohan, and Andy Hopper. Hadoopprov: Towards provenance as a first class citizen in mapreduce. In *Proceedings of the 5th USENIX Workshop on the Theory and Practice of Provenance*, TaPP '13, pages 11:1–11:4, Berkeley, CA, USA, 2013. USENIX Association.

[41] Yulai Xie, Kiran-Kumar Muniswamy-Reddy, Dan Feng, Yan Li, and Darrell D. E. Long. Evaluation of a hybrid approach for efficient provenance storage. *Trans. Storage*, 9(4):14:1–14:29, November 2013.

[42] Zachary Hensley, Jibonananda Sanyal, and Joshua New. Provenance in sensor data management. *Commun. ACM*, 57(2):55–62, February 2014.

[43] Victor Tan, Paul Groth, Simon Miles, Sheng Jiang, Steve Munroe, and Luc Moreau. Security issues in a soa-based provenance system. In *In Proceedings of the International Provenance and Annotation Workshop (IPAW06*. Springer-Verlag, 2006.

[44] Muhammad Imran and Helmut Hlavacs. Provenance in the cloud: Why and how? In *CLOUD COMPUTING 2012, The Third International Conference on Cloud Computing, GRIDs, and Virtualization*, pages 106–112, 2012.

[45] 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.

[46] Ryan K. L. Ko, Peter Jagadpramana, Miranda Mowbray, Siani Pearson, Markus Kirchberg, Qianhui Liang, and Bu Sung Lee. Trustcloud: A framework for accountability and trust in cloud computing. In *Proceedings of the 2011 IEEE World Congress on Services*, SERVICES '11, pages 584–588, Washington, DC, USA, 2011. IEEE Computer Society.

[47] Adam Bates, Ben Mood, Masoud Valafar, and Kevin Butler. Towards secure provenance-based access control in cloud environments. In *Proceedings of the Third ACM Conference on Data and Application Security and Privacy*, CODASPY '13, pages 277–284, New York, NY, USA, 2013. ACM.

[48] Uri Braun and Avi Shinnar. A security model for provenance. Technical report, Computer Science Group, Harvard University, 2006.

[49] S. Sultana, G. Ghinita, E. Bertino, and M. Shehab. A lightweight secure provenance scheme for wireless sensor networks. In *Parallel and Distributed Systems (ICPADS), 2012 IEEE 18th International Conference on*, pages 101–108, Dec 2012.

[50] S. Sultana, G. Ghinita, E. Bertino, and M. Shehab. A lightweight secure scheme for detecting provenance forgery and packet drop attacks in wireless sensor networks. *Dependable and Secure Computing, IEEE Transactions on*, PP(99):1–1, 2013.

[51] K. Alharbi and Xiaodong Lin. Pdp: A privacy-preserving data provenance scheme. In *Distributed Computing Systems Workshops (ICDCSW), 2012 32nd International Conference on*, pages 500–505, June 2012.

[52] Hasan, Ragib and Sion, Radu and Winslett, Marianne. The case of the fake picasso: Preventing history forgery with secure provenance. In *Proccedings of the 7th Conference on File and Storage Technologies*, FAST '09, pages 1–14, Berkeley, CA, USA, 2009. USENIX Association.

[53] Amril Syalim, Takashi Nishide, and Kouichi Sakurai. Preserving integrity and confidentiality of a directed acyclic graph model of provenance. In *Proceedings of the 24th Annual IFIP WG 11.3 Working Conference on Data and Applications Security and Privacy*, DBSec'10, pages 311–318, Berlin, Heidelberg, 2010. Springer-Verlag.

[54] Deepak Ajwani, Adan Cosgaya-Lozano, and Norbert Zeh. A topological sorting algorithm for large graphs. *J. Exp. Algorithmics*, 17:3.2:3.1–3.2:3.21, September 2012.

[55] Mikhail J. Atallah, Marina Blanton, Nelly Fazio, and Keith B. Frikken. Dynamic and efficient key management for access hierarchies. *ACM Trans. Inf. Syst. Secur.*, 12(3):18:1–18:43, January 2009.