OTIT: Towards Secure Provenance Modeling for Location Proofs

Rasib Khan, Shams Zawoad, Md Munirul Haque, Ragib Hasan
SECRETLab, Department of Computer and Information Sciences
University of Alabama at Birmingham, AL, USA
{rasib, zawoad, mhaque, ragib}@cis.uab.edu

ABSTRACT

Personal mobile devices and location based services are gaining popularity every day. Since the location based services are often customized based on the location information, it is important to securely generate, preserve, and validate the claim of presence at a given location at a given time as well as location provenance – the history of locations for a mobile device user over a given time period. Location provenance needs to imply secure and chronological ordering of location proofs, which can be successfully verified at a later time. Otherwise, the location based services can be easily spoofed by falsified location history. In this paper, we present OTIT – a model for designing secure location provenance. We formalized the features and characteristics for the domain of secure location provenance schemes, using formal propositional logic and logical proofs. We also present several schemes, which can be used in various modes to provide secure location provenance services. Based on the characteristics defined in OTIT, we have analyzed different schemes to show their adherence to the desired features of secure location provenance. Furthermore, we present experimental results on the performance of the various schemes, in terms of time and storage, to show a comparative applicability analysis. We posit that OTIT will serve as a comprehensive benchmark framework to evaluate the models for secure location provenance.

Keywords

History Preservation, Location Provenance, Security, OTIT

1. INTRODUCTION

Smartphones and other mobile devices have removed the barriers for network oriented services. Fixed physical locations for network attachment are no longer a concern when it comes to deploying distributed systems. Users are no longer confined to a particular geographic location and can now be moving around locations to access various services. Location based services have therefore gained popularity as an application of mobility in networking. Location based services are targeted to provide customization of services based on the current physical location of the users. Service providers perform authentication, authorization, access control, accounting, and similar critical actions for the users in association with the geographical locations of the mobile devices [22].

Validity of location information and the process of validation is always a critical task. High-stake applications require the trustworthiness of location information and related proofs. Motives behind manipulation of such information may vary from trivial personal gains, such as, in social-games like FourSquare [26], to national security issues, as that of spoofing Drones with false location data [19]. Hence, location history and preservation of the corresponding records can be considered very important in various aspects.

History is not preserved by itself[1]. The source of data is always a concern to validate the authenticity of information. Provenance is important for tracing the authenticity of an item back to its source [23]. Securing data and information provenance requires a single stamp from the issuing authority, which can be validated by any requester at a later time [15].

History of locations, or location provenance, is the history of locations traveled by a user. The provenance of location is a crucial requirement in path critical scenarios. A valid claim of travel path needs to be validated only in terms of the location provenance of the travel. The integrity of a product may be highly justified by the supply chain and the intermediate locations which a product travels, as it is transported from the manufacturer to the hands of the final consumer [16]. Location presence and generating proof of presence require the information to be valid and unmodified at a later time, such that it can be used as a secure token of evidence. However, provenance for location is a continuous process. The provenance records require location provenance to be preserved, as the user travels around collecting location proofs. Moreover, unlike the general data items, the sequence, in which the locations are traveled, needs to be preserved in chronological order within the provenance. As a result, provenance for location proofs portrays a greater challenge than that for general data items.

In this paper, we present OTIT[2], a model for secure provenance for chronological preservation of location proofs. OTIT presents a formal model for generating and maintaining a secure and order-preserving chronological chain of the location proofs. We model location provenance as a combination of propositional logic, and their corresponding required properties. Based on our model, we present different approaches for such provenance recording protocol. The paper includes formal proofs for each approach, experimental evaluation, and comparative applicability analysis for each of the approaches.

Contributions: The contributions in this paper can be summarized as follows:

[1]"History is the version of past events that people have decided to agree upon." – *Napoléon Bonaparte*

[2]OTIT (pronounced \'o-θi:θ\) is the Bengali word for 'past' or 'history'.

- We present OTIT, a novel and formal model for chronological order-preserving location proofs in provenance chains. To the best of our knowledge, this is the first work done on secure and order preserving location provenance records.

- We present different approaches, which can be used to design chronological location provenance. Each of the approaches are proved formally using our propositional model for OTIT.

- We provide experimental results to analyze the performance of each of the approaches for preserving chronological location provenance. Our comparative analysis based on the OTIT model can help users in choosing different schemes based on their preferences and priorities.

The rest of the paper is organized as follows. We discuss the desired features of any provenance model and a summary of possible provenance schemes in Section 2. Section 3 presents the security analysis of the schemes based on the selected features. Performance analysis of different provenance models has been presented in Section 4. Section 5 shows a comparative analysis among different models. Related work is presented in Section 6 followed by conclusion in Section 7.

2. SECURE LOCATION PROVENANCE

Securing provenance records can be accomplished in a number of ways. In this section, we first present the threat model for secure location provenance. Subsequently, we present the requirements for any secure and chronological location provenance scheme. We also lay out the different approaches for secure preservation of location provenance, the legacy ones as well as the ones which we have modified for our purpose.

2.1 Threat Model

One of the assets in our model for secure location provenance is the chronological ordering of location proofs. An attacker will always try to alter the sequence of proofs in the provenance chain, and present the tampered provenance record to validate a false timeline for his travel path. We consider that the provenance chain resides with the user at all time. Additionally, it is assumed that an attacker has complete access to all storage and computation resources on his personal device on which he stores the proofs and the corresponding location provenance chain. Therefore, an attacker is able to modify the order of the proofs in the chain at any later time, when presenting the records to a provenance auditing authority. In addition to modifying the provenance chain, the attacker can also tamper with the location provenance record chain or the individual proofs within the provenance records.

Given that a valid provenance chain is presented by a user to an auditing authority, the auditor can try to look at information from the individual proofs for which the user has not intended. Thus, a breach of privacy for the user's information will occur as the auditor is verifying the provenance of location proofs. Moreover, given that the location provenance is a chain of records, the user is unable to protect himself from exposing a subset of proofs from within the chain of proofs.

In an alternative scenario, a malicious user may want to hide an intermediate point in a sequence of location proofs. Even though the source and destination locations are the same, and in order, the attacker is able to hide a temporary off-track movement from the claimed provenance of locations. Therefore, the auditor, validating the claim of chronological location proofs, will be oblivious to the fact that the malicious user is hiding a rerouted path between two given locations. Finally, an attacker may want to daunt a verifying auditor with a huge task of validation in terms of data overload.

2.2 Requirements for Location Provenance

Here, we formalize and present the features and requirements for OTIT, which are necessary for designing any secure location provenance scheme.

Chronological: The location proof records should be ordered according to the sequence of their visits. It is required that, the order in which the location proofs are obtained by the user, should be entered into the location provenance chain in that order. The chronology of the proofs ensures that a malicious user is not able to create false provenance chains for the order of visits. Given that a user visited locations A, B and C in the sequence $A \rightarrow B$ and $B \rightarrow C$, the provenance record chain should hold the corresponding proofs in the given chronology of visits. Considering a location provenance chain, ... + Proof(A) + Proof(B) + Proof(C) + ... should be the order in the given sub-sequence.

Order Preserving: Given the chronology of the location proofs in the provenance chain, the provenance chain of location proofs should preserve the order in which the proofs were entered into the chain. As we assume location provenance in a completely user-centric environment, a user has complete access to the provenance records stored on his personal device. As a result, it is required that the order of proofs, with which the provenance chain is created, is preserved at any point of time. This means that at a later time $(t_i + \delta t_i)$, where t_i is the time at which any proof i has been entered in the provenance chain, the order of the proofs should be unaltered.

Verifiable: The proofs in the provenance and the order of the proof items should be verifiable by a trusted auditor. Users possess location provenance chains and use the chain of records to prove a given sequence of travel. Hence, when an auditor is presented with a provenance chain by the user, the auditor should be able to verify the claim by the user of visiting locations A, B, and C in the order $A \rightarrow B$ and $B \rightarrow C$. A successful verification would validate the claim by the user, and would have to be a false claim otherwise.

Tamper Evident: The information within the location provenance chain should be tamper evident. In contrast to ensuring tamper proof information, we emphasize that the provenance records should reside with the user. We assume that the location provenance is a user-centric scheme. Thus, the user has the flexibility of tampering with the information within the provenance chain. Given a tampered item, the verifying auditor should be able to detect that the provenance chain has been tampered or modified by an unauthorized party.

Privacy Preserved: Given that a location provenance chain is presented to an auditor, it should not reveal any additional information, which is not intended by the user to be revealed. Even though we model location provenance preservation as a user-centric design, the user needs to present the proofs to the verifying auditor for validating the claimed provenance. For the proofs, which are being presented for verifying the provenance chain, it is required that the user has a control on the level of privacy exposure. The overall expectation is, when the provenance chain is verified for specific location proofs, the privacy is preserved for the other location proofs within the chain.

Selective In-Sequence Privacy: A provenance scheme is required to support sub-set verification in terms of privacy and convenience. Location provenance generation is a continuous process and evolves with time. The provenance chain adds the collected proofs in chronological order and builds itself. At any given moment, it is highly

	HC	BC	BF	SH	MH	RC
Chronological	✓	✓	✓	✓	✓	✓
Order Preserving	✓	✓	✓	✓	✓	✓
Verifiable	✓	✓	✓	✓	✓	✓
Tamper Evident	✓	✓	✓	✓	✓	✓
Privacy Preserved	✗	✗	✗	✓	✗	✓
Selective In-Sequence Privacy	✗	✗	✓	✗	✗	✗
Privacy Protected Chronology	✗	✗	✗	✗	✗	✓
Convenience and Derivability	✗	✓	✓	✗	✓	✗

Table 1: Comparison of Location Proof Provenance Approaches: Hash Chains (HC), Block-Hash Chains (BC), Bloom Filter (BF), Shadow Hash Chain (SH), Multi-Link Hashing (MH), and RSA Chaining (RC)

likely that the user will want to prove only a sub-set from the current location provenance chain. Therefore, given a sequence of proofs ... + Proof(A) + Proof(B) + Proof(C) + ..., a user may wish not to reveal the information regarding Proof(*B*), such that the auditor verifying the information should not be able to view the information within the particular proof, or subset of in-sequence proofs.

Privacy Protected Chronology: As it highly likely that a user will only desire to prove a sub-set of the location provenance chain, the provenance scheme should also ensure that the user does not hide away important information from the items within the subset of the provenance chain. Given that the sequence of proofs is ... + Proof(A) + Proof(B) + Proof(C) + ..., the user may selectively hide *Proof(B)*. However, the auditor must be able to view the chronology of the sub-sequence. This is because, in case of privacy protected provenance validation, ... + Proof(A) + Proof(C) + ... is an invalid claim, as it shows the user went from location A $\rightarrow$ C. However, ... + Proof(A) + Proof(?) + Proof(C) + ... is a valid claim with one privacy preserved subset of in-sequence proofs, as it shows the user traveled from location A $\rightarrow$ (?) $\rightarrow$ C, where Proof(?) is a hidden proof.

Convenience and Derivability: As we consider the process of collection of location proofs is a continually evolving process, the provenance chain will go on increasing in length with time. Thus, given that an auditor requires verifying a provenance claim, the process of verification should be convenient. The provenance scheme should ensure that the user does not burden the verifying auditor with a huge load of data. We specify space complexity as a measure of convenience for both the user and the auditor. Therefore, the location provenance should ensure minimal number of items, which are required to be presented by the user to the auditor for verification. A naïve approach requires a sequential access along the whole chain to allow the auditor to derive and verify a subsequence A $\rightarrow$ B $\rightarrow$ C. However, an ideal scheme should allow derivation and verification of sub-sequences in a convenient manner.

2.3 Provenance Approaches

Next, we discuss the approaches that we have considered for secure preservation of location provenance using our model - OTIT. We describe each of the schemes and also present a summarized comparison of the schemes in Table 1.

Hash Chain: Hashing refers to the most primitive form of integrity preservation techniques. Hash chains are an extension of the basic hashing scheme to preserve the order of hashed values. We use the hash of the i^{th} proof and append it to proof $(i+1)$ to create the $(i+1)^{th}$ hash element [15]. For the first proof, hash of element 0 uses an initialization vector to begin the chain creation process.

Block Hash Chain: In our block hash chains, we create sub-segments or blocks of hash chains, each starting with a unique initialization vector. Therefore, for every k^{th} proof, we terminate the chain block, and begin the next chain block using another unique initialization vector. Additionally, we also store a hash chain of the unique initialization vectors for each block, which ensures that the blocks of hash chain cannot be altered in sequence. The user is required to store a total of n/k initialization vectors, where n is the total number of proofs.

Bloom Filter: The Bloom filter is an accumulator based data structure, with a probabilistic membership verification [5]. Our Bloom filter based approach uses individual fixed length Bloom filter for each proof. The individual Bloom filters accumulate the hash values for the proofs and are used to prove implicit chronology of location proofs. For the given Bloom filter for proof i, all the proofs from 0 to i are hashed individually and included in the Bloom filter. Therefore the Bloom filter for proof $(i+1)$ holds the set bits for all the hashes of the proofs from 0 to $(i+1)$.

Shadow Hash Chain: We introduce the concept of a shadow hash chain, which refers to two hash chains together being generated in the same way as for a single hash chain. We generate the first chain by forming the hashes from the plain text proofs. However, another corresponding hash entry is made into the second hash chain, which is a hash generated from encrypting proof $(i+1)$ and padding the hash for encrypted proof i. In this context, the second hash chain is referred to as the shadow hash. The shadow hash chain is useful in proving a sequence of location proofs $(i-1)$, i, and $(i+1)$, without revealing the plain text contents of proof i.

Multi-Link Hash Chain: In multi-linked hashing, each proof entry i in the hash chain is padded with the hash of k other proofs, where k is a subset of of proofs such that $k_m \in [0..(i-1)]$, and $m \ll n$, where n is the total number of proofs, and $|k| = m$.

RSA Chaining: An RSA chain is formed using the concepts of an RSA accumulator [2, 3, 7]. An RSA accumulator is a cryptographic one-way data structure, similar to Bloom filter. The accumulator is based on RSA assumption and provides the functionality of checking the membership of an element in a set. Unlike Bloom filters, RSA accumulators works with zero false negative and false positive probability. In this case, we create a cryptographic accumulator using the hash of the previous proof in the provenance chain and a secret number. The RSA chain helps to prove location provenance without requiring the user to expose any proofs, unless intended.

3. SECURITY ANALYSIS

In this section, we present the security analysis for the aforementioned provenance schemes for location proofs. Our proofs are based on the required properties for location provenance.

Lemma 1: *A location proof is a securely generated data item for user U, which validly verifies the presence of user U at location L_i, where i ∈ {1, 2, ..., n}.*

Lemma 2: *A location provenance chain C is a record of location proofs for locations L_i, where i ∈ {1, 2, .., n }, and presence at each location L is verified using a location proof Proof(L) for that location.*

Therefore, using **Lemma 1** and **Lemma 2**, we can say that if a user U presents a provenance chain C, which has Proof(*L*) as one of the elements, this verifies the claim that the user U was present at location L.

3.1 Security Propositions

Using the above lemmas, we put forward the following security propositions for location provenance.

Proposition 1 - Chronological (P1):
If user U visited locations (L_{i-1}), (L_i), and (L_{i+1}) in order $(L_{i-1}) \rightarrow (L_i) \rightarrow (L_{i+1})$, the provenance chain C enters the location proofs as Proof(L_{i-1}) + Proof(L_i) + Proof(L_{i+1}), which is the order in which they were received by user U.

Proposition 2 - Order Preserving (P2):
If user U visited locations (L_{i-1}), (L_i), and (L_{i+1}) in order $(L_{i-1}) \rightarrow (L_i) \rightarrow (L_{i+1})$, given that Proposition 1 holds true at time t, the provenance chain C preserves the order at time $(t + \delta t)$, where δt is a positive value.

Proposition 3 - Verifiable (P3):
If user U presents an auditor A the location provenance chain C with Proof(L_{i-1}) + Proof(L_i) + Proof(L_{i+1}), and individual proofs - Proof(L_{i-1}), Proof(L_i), and Proof(L_{i+1}), the auditor can successfully verify the claimed order of visits $(L_{i-1}) \rightarrow (L_i) \rightarrow (L_{i+1})$ for user U.

Proposition 4 - Tamper Evident (P4):
If user U presents a tampered location provenance chain C_T and/or tampered individual proofs to the auditor A, the auditor can successfully detect the tampering and the falsely claimed order of visits for user U.

Proposition 5 - Privacy Preserved (P5):
If user U wants to validate his location provenance to an auditor A, no information intended to be hidden is visible to the auditor A.

Proposition 6 - Selective In-Sequence Privacy (P6):
If user U presents the location provenance chain C which is currently holding Proof(L_{i-1}) + Proof(L_i) + Proof(L_{i+1}), the user is able to prove Proof(L_{i-1}), Proof(L_{i+1}), without revealing Proof(L_i) to the auditor A.

Proposition 7 - Privacy Protected Chronology (P7):
If a user U presents to an auditor A the location provenance chain C containing Proof(L_{i-1}) + Proof(L_i) + Proof(L_{i+1}), and hides Proof(L_i), the auditor A is able to validate the provenance chain for the sequence of visits as $(L_{i-1}) \rightarrow (?) \rightarrow (L_{i+1})$.

Proposition 8 - Convenience and Derivability (P8):
If a user U has a provenance chain C with n number of proofs, and wants to prove m number of proofs from the provenance chain C to an auditor A, with the maximum range of proofs as r, the complexity of computation for verification is less than O(r), and greater or equal to O(m).

Given the above propositions (P1 - P8), next we will show the proofs for the different location provenance approaches discussed in Section 2.

3.2 Provenance Proofs

Here we prove/disprove all 8 propositions for each of the schemes mentioned in Section 2.

3.2.1 Hash Chain

For including Proof(L_i) in the provenance chain C, where $i = 0$, we use an initialization vector V, and a hash function *hf(x)*, and $x = Proof(L_0) + V$.

For including Proof(L_i), where $(i > 0) \wedge (i \in \Re)$, we use the hash function *hf(x)*, and $x = Proof(L_i) + hf(Proof(L_{i-1}))$.

Therefore, the order in which user U had received Proof(L_{i-1}), Proof(L_i), and Proof(L_{i+1}), is maintained in order within the location provenance chain C.

$\therefore$ **Proposition *P1* is true.**

Given *P1* is true, if Proof(L_i) is entered in the provenance chain C at time t_i, it implies that Proof(L_{i-1}) was already present in the chain, which was entered at time t_{i-1}. A trusted provenance authority generating the provenance chain for the user ensures the order preservation before the current proof is entered into the chain. Therefore, $t_{i-1} < t_i$, that is, $t_{i-1} + \delta t = t_i$.

$\therefore$ **Proposition *P2* is true.**

A user U intends on proving the location proof provenance $(L_i) \rightarrow (L_{i+m})$, where $m \geq 1$. The user U presents the following items to the auditor: (i) the proofs Proof(L_x), where $i \leq x \leq i + m$, and (ii) the hash value $hf(Proof(L_{i-1}))$.

The auditor A calculates *hf(x)* for Proof(L_x), where $i \leq x \leq i + m$. Once it reaches Proof(L_{i+m}), the auditor compares the calculated value with the value in the hash chain C to be true.

$\therefore$ **Proposition *P3* is true.**

The user U possess the provenance chain C at time *t*. At time $t + \delta t$, tampered location provenance chain C_T is presented to the auditor A. However, the auditor calculates the whole sub-chain of the hashes and then compares the calculated value with the value in the hash chain C to be true. Additionally, the hash values used in the location provenance chain C bears a trusted signature from a provenance authority.

$\therefore$ **Proposition *P4* is true.**

For verification, the user U presents all plain text proofs, without any information hiding schemes applied. Proofs which are not intended to be exposed are enforced to be presented as well. As a result hash chain does not preserve the privacy related propositions.

$\therefore$ **Propositions *P5, P6, P7* are false.**

Additionally, all proofs are required by the auditor A to be able to derive the final hash value contradicting the convenience and derivability issue.

$\therefore$ **Proposition *P8* is false.**

Proof 1: ***Hash Chain*** *holds the following truth values for the propositions* $P1, P2, P3, P4, \neg P5, \neg P6, \neg P7, \neg P8$.

3.2.2 Block Hash Chain

For including Proof(L_i) in the provenance chain C, where $i\%b = 0$ and b is our block size, we use an initialization vector (V_j), and a hash function *hf(x)*, and $x = Proof(L_i) + V_j$, where j $\in$ {1, 2, .., n/b }.

For including Proof(L_i), where $(i\%b \neq 0) \wedge (i \in \Re)$, we use the hash function *hf(x)*, and $x = Proof(L_i) + hf(Proof(L_{i-1}))$.

For maintaining chronology of initialization vectors (V_j), we maintain a hash chain C_V of initialization vectors, where j $\in$ {1, 2, .., n/b }. We use the hash function *hf(x)*, and $x = V_j + hf(V_{j-1})$.

Therefore, the order in which user U had received Proof(L_{i-1}), Proof(L_i), and Proof(L_{i+1}), is maintained within the location provenance chain C within each block b_i. The order of the blocks are maintained using hash chain C_V. Provenance proof for hash chains is already proved true in Proof 1.

$\therefore$ **Proposition *P1* is true.**

The chronological order preservation of location proofs in the provenance chain functions in a similar manner as to that in hash chains.

Proposition *P2* is true, $\because$ already true in Proof 1.

A user U intends on proving the location proof provenance $(L_i) \rightarrow (L_{i+m})$, where $m \geq 1$.

In the first case, (L_i) and (L_{i+m}) are within the same block. In this scenario, the verification proceeds the same way as in Proof 1.

In the second case, (L_i) and (L_{i+m}) are in two separate blocks *j* and *k*, where each block is of size *b*, and $j < k$. The user U presents the following items to the auditor: (i) the j^{th} block of the

provenance chain C, where the jth block holds (L_i), (ii) the proofs Proof(L_x), where $j * b \leq x \leq i$, (iii) the kth block of the provenance chain C, where the kth block holds (L_{i+m}), (iv) the proofs Proof(L_y), where $k * b \leq y \leq i + m$, (v) the block initialization vectors V_z, where $j \leq z \leq k$, and (vi) the hash value $hf(V_{j-1})$.

The auditor A calculates *hf(x)* for Proof(L_x), where $j * b \leq x \leq i$, till it reaches Proof(L_i). The auditor A then calculates *hf(x)* for Proof(L_y), where $k * b \leq x \leq i + m$, till it reaches Proof(L_{i+m}). The auditor compares the calculated values with the value in the hash chain C to be true.

The auditor A then verifies the initialization vector hash chain C_V, using $hf(V_{j-1})$, and initialization vectors V_j to V_k. The auditor compares the value with the presented hash chain C_V to be true.

$\therefore$ **Proposition *P3* is true, given *P3* is true in Proof 1.**

Proposition *P4* is true, $\because$ already true in Proof 1.

Similar to hash chains, the block hash chain does not incorporate any information encapsulation. Although the block hash chain does not require all items within Proof(L_x) and Proof(L_y), where $x < y$, it does not allow selectively hiding proofs. Neither does it allow in-sequence privacy, as it does not allow the user U to choose the item for which to preserve the privacy in chronology.

$\therefore$ **Propositions *P5, P6, P7* are false**.

As shown above, in the worst case, the user U only presents a sub-set of the proofs between Proof(L_x) and Proof(L_y).

$\therefore$ **Proposition *P8* is true**.

Proof 2: ***Block Hash Chain*** *holds the following truth values for the propositions* $P1, P2, P3, P4, \neg P5, \neg P6, \neg P7, P8$.

3.2.3 Bloom Filters

We use Bloom filters for implicitly maintaining location proof entries in the provenance chain C. For every location proof Proof(L_i), there is a corresponding Bloom filter BF_i. The Bloom filter accumulator BF_i inserts the signed location proofs as a function *FuncBEnt()*, such that $BF_i = \forall_x FuncBEnt[Proof(L_x)]$, where $x \in \{0 \dots i\}$. Therefore, by induction, BF_{i+1} for Proof(L_{i+1}) is created such that, $BF_{i+1} = \forall_x FuncBEnt[Proof(L_x)]$, where $x \in \{0 \dots i+1\}$.

$\therefore$ **Proposition *P1* is true**.

Given *P1* is true, if the provenance Bloom filter BF_i is created for Proof(L_i) at time t_i, it implies that BF_{i-1} which existed for Proof(L_{i-1}) was created at time t_{i-1}. Therefore, $t_{i-1} < t_i$, that is, $t_{i-1} + \delta t = t_i$.

$\therefore$ **Proposition *P2* is true**.

A user U intends on proving the location proof provenance $(L_i) \rightarrow (L_{i+m})$, where $m \geq 1$. The user U presents the following items to the auditor: (i) the proofs Proof(L_i) and Proof(L_{i+m}), and (ii) the corresponding Bloom filters BF_i and BF_{i+m}.

The auditor A performs a membership check on the Bloom filter, using function *FuncBChk()*, which returns either *True* or *False*. Membership check for Proof(L_i) in both BF_i and BF_{i+m} using *FuncBChk()* should return *True*. Membership check for Proof(L_{i+m}) in both BF_i and BF_{i+m} using *FuncBChk()* should return *False* and *True* respectively. Therefore, it is verified that $BF_i \subset BF_{i+m}$, and $(L_i) \rightarrow (L_{i+m})$.

$\therefore$ **Proposition *P3* is true**.

At time *t*, the user U possesses the Bloom filters BF_k, where $0 \leq k \leq t$. At time $t + \delta t$, tampered Bloom filters BF_i^T and BF_j^T are presented to the auditor A in correspondence with Proof(L_i) and Proof(L_j), where $i < j$. However, the auditor uses both the proofs and the Bloom filters with the membership check function *FuncBChk()* to verify $BF_i \subset BF_{i+m}$. Additionally, the proofs used for creating the Bloom filters for function *FuncBEnt()* are signed by a trusted provenance authority.

$\therefore$ **Proposition *P4* is true**.

Our Bloom filter approach does not include any encryption for the proofs. As a result, the user U does not have any option for information hiding within the location proofs.

$\therefore$ **Proposition *P5* is false**.

User U wants to prove $(L_{i-1}) \rightarrow (L_{i+1})$ without revealing (L_i). Therefore, the user U presents Proof(L_{i-1}), Proof(L_{i+1}), and the corresponding Bloom filters BF_{i-1} and BF_{i+1} to the auditor A. As already shown above, $BF_{i-1} \subset BF_{i+1}$. Therefore, the auditor A can verify the claim by user U.

$\therefore$ **Proposition *P6* is true**.

User U wants to prove $(L_{i-1}) \rightarrow$ (?) $\rightarrow (L_{i+1})$. Therefore, the auditor checks only BF_{i-1} and BF_{i+1}, and verifies $BF_{i-1} \subset BF_{i+1}$, without the knowledge that there is another location in between (L_{i-1}) and (L_{i+1}).

$\therefore$ **Proposition *P7* is false**.

As shown above, given that there are n proofs and Bloom filters in the provenance chain C, the user U only presents m proofs and Bloom filters. Here, m is number of locations for which user U wants to receive validation from the auditor A and $m << n$.

$\therefore$ **Proposition *P8* is true**.

Proof 3: ***Bloom Filter*** *holds the following truth values for the propositions* $P1, P2, P3, P4, \neg P5, P6, \neg P7, P8$.

3.2.4 Shadow Hash Chain

A shadow hash chain works in the same way for hash chains. However, in addition to the chain of hash values for Proof(L_i), where $(i \neq 0) \wedge (i \in \Re)$, there is a secondary chain of encrypted hash values of the proofs Enc[K$_i$, Proof(L_i)], where $Enc()$ is an encryption function, and K$_i$ is the unique encryption key for Proof(L_i).

Proposition *P1, P2, P3, P4* is true, $\because$ already true in Proof 1.

The user U wants to prove $(L_i) \rightarrow (L_{i+1})$ to auditor A, without revealing the information within Proof(L_{i+1}). Therefore, along with the segment of the hash chain, the user U presents the unique key K$_i$, Enc[K$_i$, Proof(L_i)], and Enc[K$_i$, Proof(L_{i+1})]. The auditor A thus calculates the encrypted proof hash chain in the same way as before, without Proof(L_{i+1}) being visible. Verification by preserving sub-chain privacy can also be executed by presenting three sub-chains: (i) $(L_i) \rightarrow (L_{i+j-1})$ from the regular hash chain, (ii) $(L_{i+j}) \rightarrow (L_{i+k-1})$ from the encrypted shadow hash chain, and (iii) $(L_{i+k}) \rightarrow (L_{i+m})$, where $0 \leq i < j < k < m \leq n$.

$\therefore$ **Proposition *P5* is true**.

The auditor A requires the shadow hash chain to be generated sequentially, and does not allow the user U to selectively hide any proof from within a sequence. The chain requires all the proofs or the encrypted proofs to be presented to the auditor A.

$\therefore$ **Propositions *P6, P7* are false**.

Given that *P1* is true, and *P6, P7* are false, the user U proving $(L_i) \rightarrow (L_{i+r})$ requires to present r items for verification.

$\therefore$ **Proposition *P8* is false.**

As an extension to shadow hash chains, we can apply the block based approach to the shadow hash chain scheme, as shown in block hash chains. In this case, we will use the initialization vector V_k for each block k of the hash chain, as well as the encrypted hash chain. Therefore, we are able to achieve the complexity of computation less than *O(n)*.

Proposition *P8* is true for blocks, $\because$ already true in Proof 2.

Proof 4a: ***Shadow Hash Chain*** *holds the following truth values for the propositions* $P1, P2, P3, P4, P5, \neg P6, \neg P7, \neg P8$.

Proof 4b: ***Block Shadow Hash Chain*** *holds the following truth values for the propositions* $P1, P2, P3, P4, P5, \neg P6, \neg P7, P8$.

3.2.5 Multi-Link Hash Chain

In our scheme for multi-link hash chains, we use two-link (or more) hashing for each proof in the provenance chain C.

For including Proof(L_i) in the hash chain, we use the hash function *hf(x)*, and $x = Proof(L_i) + V_i + V_{i+s-1}$, where $0 \leq i \leq s-1$, and s is the link size. For including Proof(L_i), where $s \leq i \leq n$, we use the hash function *hf(x)*, and $x = Proof(L_i) + hf(Proof(L_{i-1})) + hf(Proof(L_{i-s}))$. The other properties and operations are the same as hash chains.

Proposition *P1, P2* is true, $\because$ already true in Proof 1.

If user U wants to prove $(L_i) \rightarrow (L_{i+m})$ to auditor A, where $m \leq s$, the convenience and derivability remains the same as in hash chains. When $m \leq s$, user U presents the following to the auditor A: (i) Proof(L_i) and Proof(L_{i+m}), (ii) immediately preceding hash values $hf(Proof(L_{i-1}))$, $hf(Proof(L_{i+m-1}))$, (iii) the linker hash value for ith position $hf(Proof(L_{i-s}))$, and (iv) the linker hash values for (i+m)th position $hf(Proof(L_{(q*r)-s}))$, and the proofs Proof(L_{q*r})), where $q = ((i+m)\%s)$, and *r* is the number of leaps of size s, from $i \rightarrow (i+m)$.

Example: *User U wants to prove* $(L_6) \rightarrow (L_{13})$ *to an auditor A.*

For proving $(L_6) \rightarrow (L_{13})$, let us assume that the user U maintains a multi-link hash chain, with link length $s = 5$, and presents the following items to the auditor for verification of the claim:

Proofs: Proof(L_6), Proof(L_7), Proof(L_8), and Proof(L_{13}).

Hash values:$hf(Proof(L_1))$, $hf(Proof(L_2))$, $hf(Proof(L_3))$, $hf(Proof(L_5))$, and $hf(Proof(L_{12}))$.

For verification, the auditor A calculates $hf(Proof(L_6))$ using Proof(L_6), $hf(Proof(L_5))$, and $hf(Proof(L_1))$. The auditor then calculates $hf(Proof(L_7))$ using Proof(L_7), $hf(Proof(L_2))$, and the previously calculated $hf(Proof(L_6))$. Next, the hash $hf(Proof(L_8))$ is calculated using Proof(L_8), $hf(Proof(L_3))$, and the previously calculated $hf(Proof(L_7))$. Finally, the auditor calculates $hf(Proof(L_{13}))$ using Proof(L_{13}), $hf(Proof(L_{12}))$, and the previously calculated $hf(Proof(L_8))$.

$\therefore$ **Proposition *P3* is true.**

Proposition *P4* is true, $\because$ already true in Proof 1.

Propositions *P5, P6, P7* are false, $\because$ already false in Proof 2.

As shown in the example, for proving $(L_i) \rightarrow (L_{i+m})$, the user U presents a subset of the proofs from the range $i \rightarrow (i+m)$.

$\therefore$ **Proposition *P8* is true.**

Proof 5: ***Multi-Link Hash Chain*** *holds the following truth values for the propositions* $P1, P2, P3, P4, \neg P5, \neg P6, \neg P7, P8$.

3.2.6 RSA Chaining

We use RSA accumulators to design our RSA chains for location provenance. For every RSA provenance chain, we require private values P and Q, which are large prime numbers. The first public value is N, where N = P*Q, and the second public value is a large random number which is the initial seed R.

An accumulator C_i for the RSA chain is created using the function *FuncRSACreate()*, such that $C_i \in \forall_i$*FuncRSACreate(x)*. Here, *x*=[$R^{hf(Proof(L_i))}$ % N], for i=0; and *x*=[$C_{i-1}^{H(Proof(L_i))}$ % N], for $(i \in \Re) \wedge (i > 0)$, and $hf(Proof(L_i))$ is the numerical hash value for Proof(L_i).

$\therefore$ **Proposition *P1* is true.**

Given *P1* is true, C_i is created for the Proof(L_i) at time t_i. Therefore, it implies that either it is the first location proof, for which we have the initial public seed R, which is the trivial case. Otherwise, we know that C_{i-1} was created at t_{i-1}, in which case $t_{i-1} < t_i$, that is $t_{i-1} + \delta t = t$.

$\therefore$ **Proposition *P2* is true.**

The user U wants to prove the location provenance $L_i \rightarrow L_{i+m}$. He presents the auditor A the following items: (i) the RSA chain accumulators C_i and C_{i+m}, (ii) the hash values $hf(Proof(L_r))$, where $(i-1) \leq r \leq (i+m)$, (iii) the proofs Proof(L_i) and Proof(L_{i+m}), and (iv) the public value N. The auditor A uses the presented items to create C_{i+m} from C_i, and compares the validity with the one presented by the user U.

$\therefore$ **Proposition *P3* is true.**

At time *t*, user U possesses the RSA chain accumulators C_k, where $0 \leq k \leq t$. The tampered accumulators C_i^T and C_j^T are presented to the auditor A by the user U, which correspond to the locations L_i and L_j, and $i < j$. However, the auditor only accepts the claim of location provenance if and only if the sequential derivation of the accumulator from C_i^T to C_j^T is successful. Additionally, the hash values for $hf(Proof(L_r))$, where $(i-1) \leq r \leq (j)$, are signed by the trusted provenance authority, need to be validated as well.

$\therefore$ **Proposition *P4* is true.**

The auditor A verifies the location provenance using the hash values for all proofs, the RSA chain accumulators, and the visible proofs for only the intended items. The user U never presents anything which is not intended to be revealed to the auditor A.

$\therefore$ **Proposition *P5* is true.**

User U intends on proving the location provenance $L_{i-1} \rightarrow L_i \rightarrow L_{i+1}$, without revealing location L_i. He presents the hashes $hf(Proof(L_{i-1}))$, $hf(Proof(L_i))$, and $hf(Proof(L_{i+1}))$, and the plain text proofs Proof(L_{i-1}) and Proof(L_{i+1}).

$\therefore$ **Proposition *P6* is false.**

However, no information apart from the hash regarding L_i is exposed, which is a one-way function, and does not effect the privacy. As a result, the user U is able to prove $L_{i-1} \rightarrow (?) \rightarrow L_{i+1}$.

$\therefore$ **Proposition *P7* is true.**

For verifying $L_i \rightarrow L_{i+m}$, the auditor A requires all the hash values of proofs $hf(Proof(L_r))$, where $(i-1) \leq r \leq (i+m)$.

$\therefore$ **Proposition *P8* is false.**

Proof 6: ***Multi-Link Hash Chain*** *holds the following truth values for the propositions* $P1, P2, P3, P4, P5, P6, P7, \neg P8$.

4. EVALUATION

In this section, we present the performance analysis of the proposed schemes based on storage and time requirement for different operations on provenance chain.

4.1 Experimental Setup

Location proof database: Since we have not found any database containing location proofs directly, we have modified the Foursquare database [12] for our purpose. Foursquare database contains 2073740 check-ins of 18107 users from March 2010 to January 2013. Each entry of this database contains user ID, latitude, longitude, time, and location ID. We concatenate all the fields and hash the concatenation using MD5 hash algorithm generating a 32 byte string, which we use as a location proof entry.

Criteria: We have evaluated the performance of each of the schemes based on the following common operations done on location proofs.

- *Provenance generation time:* This refers to the time required to generate provenance information from the location proofs and insert them in the provenance chain. To evaluate the performance of the schemes, we generate provenance for different number of proofs e.g. 500, 1000, etc. and measure their respective timing requirement.

- *Sequential chain verification time:* This is the time required to verify all the location proofs in the provenance chain in sequential manner.

- *Sparse chain verification time:* This is the time required to verify the sequence (who comes first) of any two randomly selected location proofs of the provenance chain. We have termed the first proof as the source proof and the second proof as the destination proof in the rest of the paper. It should be noted that, the sparse verification time increases with the increase of the gap between these randomly selected proofs. This issue creates a problem since for one scheme, the gap between random selections might be too small compared to other schemes and thus creating drastic difference in the performance. In order to handle this scenario, we have always picked proofs with fixed gap size for all the schemes. For example, we have measured the sparse verification time for the 1^{st} and 100^{th} proof, 1^{st} and 200^{th} proof and so on for all the schemes.

System configuration: We implemented the schemes on a Dell laptop, running Debian 3.2.46-1 on Intel Core 2 Duo CPU (6MB Cache, 2.66GHz) with 4GB of RAM. All of the schemes are developed on OpenJDK (version:1.6.0_27). We used SHA-2 (SHA-256) hash function for hashing and RSA (1024 bit) for encryption.

4.2 Block Hash Chain

Figure 1 presents the performance analysis of block hash chain scheme for different block sizes.

Provenance Generation Time: From Figure 1a, we notice that the provenance generation time increases linearly with the increase in number of proofs, and performance increases with the increase in block size. For smaller block size, we will have more number of blocks, i.e., we need to maintain a longer chain of initialization vectors (V) for smaller block size. Hence, for bigger block size, we get better performance for provenance generation.

Sequential Verification Time: Figure 1b shows the time requirement for sequential chain verification. We notice that, the verification time increases linearly with the number of proofs irrespective of the block size. Moreover, the graph shows that the timing requirements vary a little with the increased block size. For sequential verification, we need to verify the provenance of each proof in relation with its previous proof inside the same block. The provenance information of the first element of a block is verified using the initialization vector of that block. Hence, for the first element of a block, we do not need to retrieve the hash chain of its previous element. Retrieving initialization vector of a block is slightly faster than retrieving the hash chain of the previous element. For this reason, we get slightly better performance for smaller block size.

Sparse Verification Time: Figure 1c presents the performance analysis of verifying two provenance entries, where the location proofs are separated by varying distances. In block hash chain scheme, first we need to verify the presence of the selected proofs in their respective blocks. Once the blocks are identified, we need to check whether the initialization vector of the source proof (V_s) comes before that of the destination proof (V_d). In order to do so, we traverse through the provenance chain of the initialization vectors starting with V_s and try to reach V_d. We get better performance for larger block size since it ensures shorter length for the provenance chain of initialization vectors.

Space Requirement: Provenance information is stored as a signature of location authority on the hash. Hence, for each provenance information, we require 128 bytes (for RSA 1024). As we are maintaining two provenance chains (location proofs and initialization vectors), *N* number of proofs require N*128 + (N/Block Size)*128 bytes. The space requirement is for one proof is therefore:

$$128 * (1 + 1/BlockSize) \quad (1)$$

According to equation 1, space requirement for block size 5, 25, and 50 are 153.6, 133.12, and 130.56 bytes respectively.

4.3 Bloom Filter

Figure 2 presents the performance analysis of Bloom filter based scheme for two different filter sizes: 1000 and 2000 byte.

Provenance Generation Time: From Figure 2a, we notice that the provenance generation time increases linearly with the increase in number of proofs. We can generate approximately 12 provenance entries per second using 1000 byte Bloom filter. Whereas, using 2000 byte Bloom filter, we can generate roughly 3 provenance entries per second. However, the smaller sized Bloom filter suffers from higher false positive probability. For example, 1000 byte Bloom filter generates 2% false positive results considering 1000 proofs. Whereas, using 2000 byte Bloom filter, we can reduce the false positive probability to 0.04% for the same number of proofs [5].

Sequential Verification Time: Figure 2b shows the time requirement for sequential chain verification. We notice that, for both the Bloom filters, time increases linearly with the number of proofs. Using 1000 byte Bloom filter, we can verify a provenance chain of length $\approx$ 3758 in one second and for the 2000 byte Bloom, the corresponding chain length is approximately 2685. The difference in sequential verification timing requirement, for the different sized Bloom filters, is significantly less compared to that of provenance generation.

Sparse Verification Time: Figure 2c presents the performance analysis of sparse verification for two different sized Bloom filters. Both the Bloom filters take nearly constant time for this verification. The reason is that, we always need to run only one subset checking operation for the two provenance entries. Because of the larger bit array, we get slightly higher time for the larger Bloom filter.

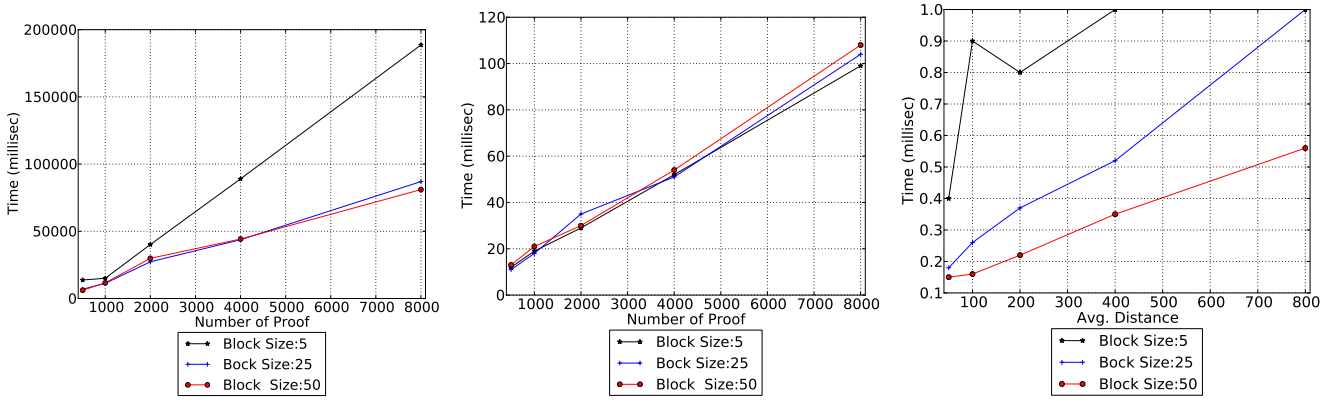
Space Requirement: As location authority will sign the Bloom filter and we preserve only the signature, Bloom Filter scheme requires 128 bytes (for RSA 1024) to preserve the provenance of one proof. Noticeably, the storage requirement does not change with the size of the Bloom filter.

4.4 Multi-Link Hash Chain

Figure 3 presents the performance analysis of multi-link hash chain scheme for different link sizes.

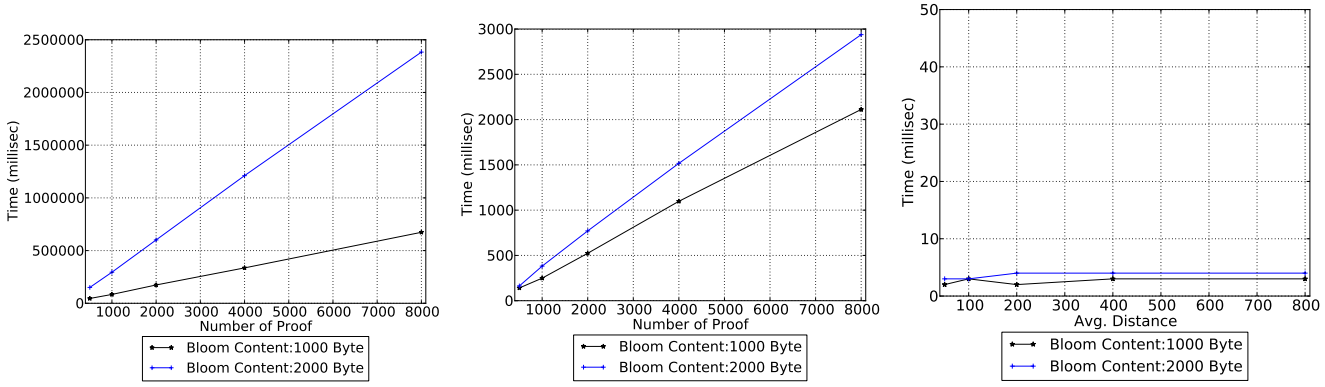
Provenance Generation Time: From Figure 3a, we notice that the provenance generation time increases linearly with the increase in number of proofs for each of the link sizes. We did not get much variation in time for different sized links. we get the best performance for the link with highest size. As we need lesser linking operations for larger link size, the provenance generation time decreases with the increase in link size.

Sequential Verification Time: Figure 3b shows the time requirement for sequential chain verification. For each of the link sizes, time increases linearly with the number of proofs, and the required times are very close for different link sizes. For sequential verification, we need to verify the provenance of each proof with its previous proof and the proof with whom it is linked. As larger link size requires lesser number of links, number of operations that



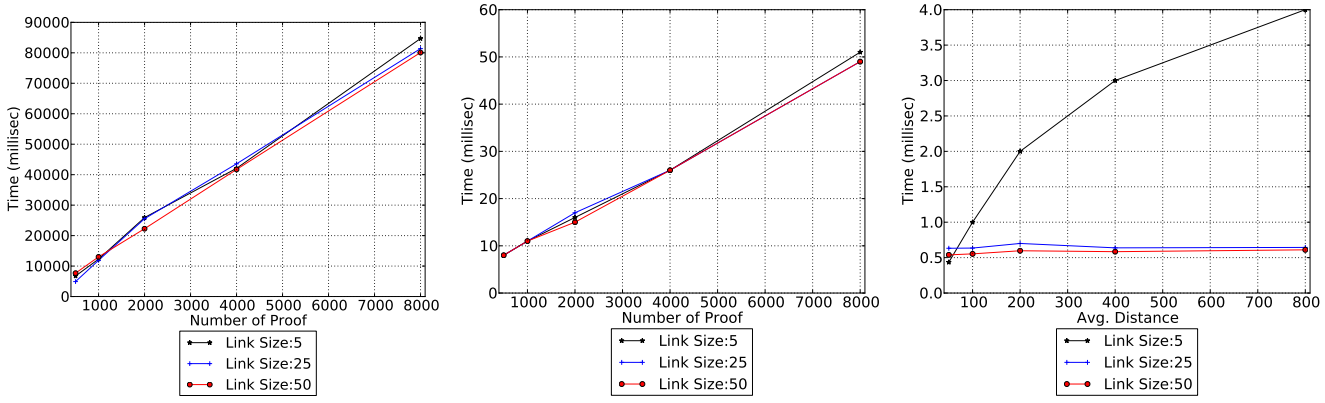
(a) Provenance Generation (b) Sequential Verification (c) Sparse Verification

Figure 1: Time Required in Different Operations for Block Hash Chain



(a) Provenance Generation (b) Sequential Verification (c) Sparse Verification

Figure 2: Time Required in Different Operations for Bloom Filter



(a) Provenance Generation (b) Sequential Verification (c) Sparse Verification

Figure 3: Time Required in Different Operations for Multi-link Chain

check the provenance with the linked proof will be reduced. Hence, higher link size gives better performance.

Sparse Verification Time: Figure 1c presents the performance analysis of sparse verification for different size of links. As we can quickly reach from the source proof to the destination proof using larger links, we observe better performance with increased link size. However, if the distance between the source and destination proof is less than the link size, we cannot utilize the benefit of using multi-link chain. Hence, we get better performance using smaller

sized links, where the distance between the source and destination proof is smaller.

Space Requirement: Space requirement does not vary with link size for this scheme, as we do not need any extra storage for the links. Location authority signs hash value and we preserve only the signature of the hash. Hence, multi-link hash chain scheme requires 128 bytes (for RSA 1024) to preserve the provenance of one proof.

4.5 RSA Chaining

Figure 3 presents the performance analysis of RSA chaining scheme for two different sized private keys: 128 and 256 bits.

Provenance Generation Time: From Figure 4a, we notice that the provenance generation time increases linearly with the increase in number of proofs for both of the key sizes. For larger key size, the performance decreases significantly. Time to generate one provenance entry using 128 bits key is approximately 628 milliseconds. Whereas, using the 256 bits key, time required to generate one proof is about 2620 milliseconds. However, the larger the key size, the higher the security against chosen cypher-text attack.

Sequential Verification Time: Figure 4b shows the time requirement for sequential chain verification using RSA chaining scheme. For each of the key sizes, time increases linearly with the number of proofs. Time required to verify the provenance of two consecutive proofs using 128 bits key is approximately 398 milliseconds, and using 256 bits key, it is around 1290 milliseconds.

Sparse Verification Time: Figure 4c presents the performance analysis of sparse verification for different key sizes. In RSA chaining scheme, we need to traverse the whole provenance chain from the source proof to the destination proof. Hence, for RSA chaining scheme, sparse verification time increases linearly with the distance between source and destination proofs.

Space Requirement: For RSA chaining scheme, space requirement to preserve the provenance of one proof is constant. The accumulator value is signed by the location authority and stored as the provenance entry. Hence, the space requirement for RSA chaining scheme is 128 bytes (for RSA 1024).

4.6 Hash Chain and Shadow Hash Chain

As we used a fixed hash algorithm for all the schemes, there is no criteria for hash chain and shadow hash chain which can be varied to compare the performance of these schemes. Moreover, the performance of hash chain and shadow hash chain should be similar as in shadow hash chain, we actually just use two hash chains: one using regular proof and the other using encrypted proof. However, the space requirement is higher in shadow hash chain compared to hash chain. In hash chain, we require 128 bytes (for RSA 1024) for the provenance entry of one proof. However in shadow hash chain, we require (2*128 + Key Size) bytes for the provenance entry of one proof.

5. DISCUSSION

To compare the performance of different schemes, we select the best performing criteria of each scheme. As discussed in Section 4, we get the best performance for block hash chain with block size 50, multi-link hash chain with link size 50, Bloom filter with filter size 1000 byte, and RSA chaining with key size 128. As the performance of shadow hash chain is similar to that of hash chain scheme, we have just included hash chain in the comparison.

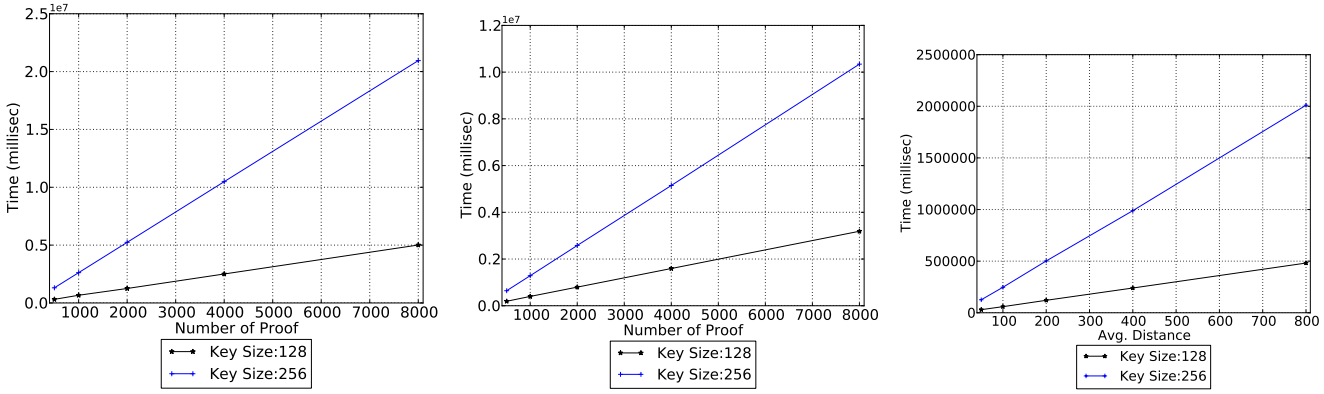
But when we tried to compare the required time for all three operations (insertion, sequential verification, and sparse verification) by all the schemes, we found that the gap between the required time by the best and worst scheme is too big to be appropriately represented in graphs. Since RSA chaining requires the highest time for each of the three operations, we chose the required time of RSA chaining as 100% and represented the time required by other schemes as a percentage of the time required by RSA chaining. For better visualization, we represented the data as the Log_{10}(Percentage of Time) in Figure 5. However, for sequential verification and sparse verification, the time required by other schemes is too small compared to that of RSA chaining and thus resulting in negative values as shown in Figure 5b and 5c. For sparse verification, we have randomly picked two proofs, which are apart by a distance of 136. According to four square database [12], on an average each user traverses 136 locations in one year. Hence, for duration of one year, average distance between the source and destination proof can be safely assumed to be 136.

Figure 5a presents the comparison of provenance entry generation time for one location proof by different schemes. We observe that for provenance generation, multi-link hash chain provides the best performance. Hash-chain and block hash chain provides nearly the similar performance compared to multi-link. Figure 5b compares the time to sequentially verify a provenance chain containing 136 entries. For both sequential and sparse verification, block hash chain scheme provides the best performance as shown in Figure 5b and Figure 5c. Considering all the three operations, we can say that block hash chain scheme performs better than all other schemes and RSA chain requires the highest time for all the cases.

However, computational performance is not always the selection criteria for choosing a provenance scheme. RSA chain complies with maximum number of criteria as mentioned in Table 1. On the other hand, block hash chain does not comply with any privacy requirement. Hence, for sensitive applications, where users' privacy is critical, we can employ the RSA chain scheme, which gives a reasonable performance with highest privacy protection.

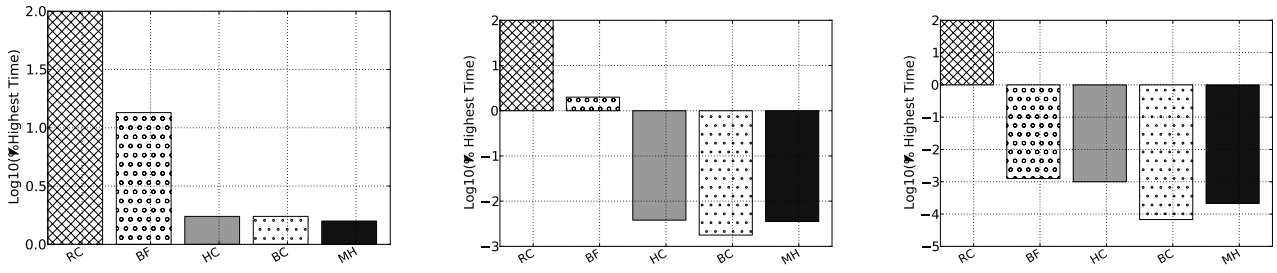
6. RELATED WORK

All location tracking and reporting mechanisms require a reliable and tamper proof architecture to preserve the integrity of the location data. Traditional Global Positioning Systems (GPS) [10] are effective in general purpose location reporting. However, it is not a suitable option in terms of security and indoor tracking. Recent papers have proposed a combination of GPS signals with other mechanisms, such as, cellular tower triangulations and identification of the access network channel. Gabber *et al.* [11] utilized multi-channel information to verify the location. They used the Caller-ID feature, GPS, cellular telephony, and satellite ranging, in a combined approach to determine the movement and location of user devices. Unfortunately, malicious entities can bypass such combinatorial schemes [18, 22]. Additionally, GPS signatures [9] are not useful since they are open to spoofing attacks [18]. Secure and unforgeable location proof was discussed by Waters *et al.* [28], which introduced the idea of collecting location proofs from a location manager. A secure geo-tagging service has been proposed by Lenders *et al.* [17], which allowed the verification of the location and timestamp for user-generated content. Zhu *et al.* proposed APPLAUS, a collusion resistant location proof updating system [30]. The collusion resistance is ensured using betweenness ranking-based and correlation clustering-based approaches. Wang *et al.* proposed STAMP, a similar approach for providing spatial-temporal provenance assurance for mobile users [27]. But none of these schemes consider preserving secure location provenance. Without secure location history, discrete location proofs cannot be used in real-life scenario.



(a) Provenance Generation (b) Sequential Verification (c) Sparse Verification

Figure 4: Time Required in Different Operations for RSA Chaining



(a) Provenance Generation (b) Sequential Verification (c) Sparse Verification

Figure 5: Comparison among Different Schemes

However, provenance has been explored in diverse areas of computer science. Provenance has been proposed for file system [21], database system [4, 6, 29], grid and distributed system [8, 24, 25]. Hasan et al. first introduced secure data provenance [14] and later they proposed a solution to ensure secure data provenance [15]. However, these schemes do not provide solution for secure location provenance.

Ananthanarayanan et al. discussed a framework for collecting and storing sequence of user locations [1]. In their StarTrack system, sequence of a user's location and time entries are stored in tracks which can later be shared, compared, clustered, and queried. While tracks are similar to location provenance chains, security issues are not considered here making tracks vulnerable to attacks by malicious users. Zugenmaier et al. introduced the notion of location stamps [31] for cell phones. The stamps provide proof about the location of the user at a certain time. Based on this idea, Gonzalez-Tablas et al. developed the notion of Path-stamps [13], where the location history of the user is considered. Here, a sequence of location stamps, i.e., location proofs, is combined by creating a hash chain. The Path-stamps protocol requires each user to possess a specialized hardware that is used to authenticate the user. This requirement makes the system difficult and expensive to deploy. Path-stamps protocol does not support publication of partial path. Therefore, users must reveal their entire path to auditors even when they are proving a subset of their path. In contrast, some of our proposed schemes allow users to have privacy by enabling them to prove any arbitrary subset of their location history. Finally, Interaction-based missed connection services have been described by Manweiler et al. [20]. Here, two mutual strangers can use the SMILE protocol to establish shared knowledge, which can later be used to prove that they have met before. However, none of these works define the criteria for secure location provenance scheme or measure the performance based on the required characteristics. In our work, we provide a complete guideline for generating secure location provenance, analyzed the performance of current schemes, and discuss their applicability.

7. CONCLUSION

As location based services become popular, collecting and securely maintaining location provenance also becomes very important. Secure generation of location proofs and their applications have the promise to revolutionize many domains, such as, in supply chains and digital forensics. In this paper, we presented OTIT, a model for formalizing the characteristics of location provenance under a single domain and map the current models based on these formal propositions. Simulated results give a clear idea about the performance of different schemes created using our model. We believe that our proposed framework can serve as a benchmarking tool for designing any secure location proof generation scheme. Currently we are working on building further schemes and a generic framework for secure and automated provenance generation. Our future work includes designing an off-the-shelf model for creating secure location provenance under a domain specific language driven definition of any arbitrary location proof generation scheme.

Acknowledgement

This research was supported by a Google Faculty Research Award and the Department of Homeland Security Grant #FA8750-12-2-0254.

References

[1] G. Ananthanarayanan, M. Haridasan, I. Mohomed, D. Terry, and C. A. Thekkath. Startrack: a framework for enabling track-based applications. In *Proceedings of the 7th international conference on Mobile systems, applications, and services*, pages 207–220. ACM, 2009.

[2] N. BariĆ and B. Pfitzmann. Collision-free accumulators and fail-stop signature schemes without trees. In W. Fumy, editor, *Advances in Cryptology - EUROCRYPT 1997*, volume 1233 of *Lecture Notes in Computer Science*, pages 480–494. Springer Berlin Heidelberg, 1997.

[3] J. Benaloh and M. Mare. One-way accumulators: A decentralized alternative to digital signatures. In T. Helleseth, editor, *Advances in Cryptology - EUROCRYPT 1993*, volume 765 of *Lecture Notes in Computer Science*, pages 274–285. Springer Berlin Heidelberg, 1994.

[4] D. Bhagwat, L. Chiticariu, W.-C. Tan, and G. Vijayvargiya. An annotation management system for relational databases. *The VLDB Journal*, 14:373–396, 2005.

[5] B. H. Bloom. Space/time trade-offs in hash coding with allowable errors. *Communication of the ACM*, 13(7):422–426, 1970.

[6] P. Buneman, S. Khanna, and T. Wang-Chiew. Why and where: A characterization of data provenance. In J. Bussche and V. Vianu, editors, *Database Theory ? ICDT 2001*, volume 1973 of *Lecture Notes in Computer Science*, pages 316–330. Springer Berlin Heidelberg, 2001.

[7] J. Camenisch and A. Lysyanskaya. Dynamic accumulators and application to efficient revocation of anonymous credentials. In M. Yung, editor, *Advances in Cryptology - CRYPTO 2002*, volume 2442 of *Lecture Notes in Computer Science*, pages 61–76. Springer Berlin Heidelberg, 2002.

[8] D. Crawl and I. Altintas. A provenance-based fault tolerance mechanism for scientific workflows. In J. Freire, D. Koop, and L. Moreau, editors, *Provenance and Annotation of Data and Processes*, volume 5272 of *Lecture Notes in Computer Science*, pages 152–159. Springer Berlin Heidelberg, 2008.

[9] D. E. Denning and P. F. MacDoran. Location-based authentication: Grounding cyberspace for better security. *Computer Fraud & Security*, 1996(2):12–16, 1996.

[10] P. Enge and P. Misra. Special issue on global positioning system. *Proceedings of the IEEE*, 87(1):3 –15, jan. 1999.

[11] E. Gabber and A. Wool. How to prove where you are: tracking the location of customer equipment. In *Proceedings of the 5th ACM conference on Computer and communications security (CCS)*, pages 142–149. ACM, 1998.

[12] H. Gao, J. Tang, and H. Liu. Exploring social-historical ties on location-based social networks. In *Proceedings of the 6th International AAAI Conference on Weblogs and Social Media*, 2012.

[13] A. I. González-Tablas, B. Ramos, and A. Ribagorda. Path-stamps: A proposal for enhancing security of location tracking applications. In *Ubiquitous Mobile Information and Collaboration Systems Workshop*. Citeseer, 2003.

[14] R. Hasan, R. Sion, and M. Winslett. Introducing secure provenance: problems and challenges. In *Proceedings of ACM StorageSS*, pages 13–18, 2007.

[15] R. Hasan, R. Sion, and M. Winslett. The case of the fake Picasso: Preventing history forgery with secure provenance. In *Proceedings of the 7th USENIX Conference on File and Storage Technologies (FASTŠ09)*, pages 1–12. USENIX Association, 2009.

[16] R. Khan, M. M. Haque, and R. Hasan. A secure location proof generation scheme for supply chain integrity preservation. In *Proceedings of The 2013 IEEE International Conference on Technologies for Homeland Security*, HST '13, pages 446–450, Waltham, MA, USA, 2013.

[17] V. Lenders, E. Koukoumidis, P. Zhang, and M. Martonosi. Location-based trust for mobile user-generated content: applications, challenges and implementations. In *Proceedings of HotMobile*, pages 60–64. ACM, 2008.

[18] W. Luo and U. Hengartner. Proving your location without giving up your privacy. In *Proceedings of HotMobile*, pages 7–12, 2010.

[19] I. Maduako. Wanna hack a drone? possible with geo-location spoofing! Online at `http://geoawesomeness.com/?p=893`, 26th July 2012.

[20] J. Manweiler, R. Scudellari, and L. P. Cox. SMILE: encounter-based trust for mobile social services. In *CCS '09: Proceedings of the 16th ACM conference on Computer and communications security*, pages 246–255. ACM, 2009.

[21] K. Muniswamy-Reddy, D. Holland, U. Braun, and M. Seltzer. Provenance-aware storage systems. In *Proceedings of the 2006 USENIX Annual Technical Conference*, pages 43–56, 2006.

[22] S. Saroiu and A. Wolman. Enabling new mobile applications with location proofs. In *Proceedings of HotMobile*, pages 1–6, 2009.

[23] Y. L. Simmhan, B. Plale, and D. Gannon. A survey of data provenance in e-science. *SIGMOD Rec.*, 34(3):31–36, September 2005.

[24] I. Souilah, A. Francalanza, and V. Sassone. A formal model of provenance in distributed systems. *TAPP*, 9:1–11, 2009.

[25] P. Townend, P. Groth, and J. Xu. A provenance-aware weighted fault tolerance scheme for service-based applications. In *Object-Oriented Real-Time Distributed Computing, 2005. ISORC 2005. Eighth IEEE International Symposium on*, pages 258 – 266, may 2005.

[26] J. VanGrove. Foursquare cracks down on cheaters. Online at `http://mashable.com/2010/04/07/foursquare-cheaters/`, April 2010.

[27] X. Wang, J. Zhu, A. Pande, A. Raghuramu, P. Mohapatra, T. Abdelzaher, and R. Ganti. STAMP: Ad Hoc Spatial-Temporal Provenance Assurance for Mobile Users. In *Proceedings of The 21st IEEE International Conference on Network Protocols*, ICNP '13, Gottingen, Germany, October 2013.

[28] B. R. Waters and E. W. Felten. Secure, private proofs of location. Technical report TR-667-03, Princeton University, January 2003.

[29] J. Widom. Trio: A system for integrated management of data, accuracy, and lineage. Technical Report 2004-40, Stanford InfoLab, August 2004.

[30] Z. Zhu and G. Cao. Toward privacy preserving and collusion resistance in a location proof updating system. *IEEE Transactions on Mobile Computing*, 12(1):51–64, 2013.

[31] A. Zugenmaier, M. Kreutzer, and M. Kabatnik. Enhancing applications with approved location stamps. In *Proceedings of IEEE Intelligent Network Workshop, 2001*, page 140. IEEE, 2001.