CLIP: Continuous Location Integrity and Provenance for Mobile Phones

Chen Lyu∗, Amit Pande†, Xinlei (Oscar) Wang‡, Jindan Zhu‡, Dawu Gu∗, Prasant Mohapatra†
∗Shanghai Jiao Tong University, Shanghai, China
Email: {chen lv, dwgu}@sjtu.edu.cn
†University of California, Davis, CA
Email: {pande, xlwang, jdzhu, pmohapatra}@ucdavis.edu

Abstract—Many location-based services require a mobile user to continuously prove his location. In absence of a secure mechanism, malicious users may lie about their locations to get these services. Mobility trace, a sequence of past mobility points, provides evidence for the user’s locations. In this paper, we propose a Continuous Location Integrity and Provenance (CLIP) Scheme to provide authentication for mobility trait, and protect users’ privacy. CLIP uses low-power inertial accelerometer sensor with a light-weight entropy-based commitment mechanism and is able to authenticate the user’s mobility trace without any cost of trusted hardware. CLIP maintains the user’s privacy, allowing the user to submit a portion of his mobility trace with which the commitment can be also verified. Wireless Access Points (APs) or co-located mobile devices are used to generate the location proofs. We also propose a light-weight spatial-temporal trust model to detect fake location proofs from collusion attacks. The prototype implementation on Android demonstrates that CLIP requires low computational and storage resources. Our extensive simulations show that the spatial-temporal trust model can achieve high (> 0.9) detection accuracy against collusion attacks.

I. INTRODUCTION

Mobile phones have become immensely popular in recent years. Location based services have attracted users’ interests with many popular applications running in the mobile phones. These applications provide access control, authentication, advertisements and other important functions based on a mobile user’s location. In addition, there is a growing trend of continuous location awareness in mobile applications, such as Geo-fencing [1] and location-based rewards. For example, companies may want to encourage employees to increase their physical activities by participating in biking, walking and running. The employees must be able to show their past trajectories to the companies. This notion is being used to reward and encourage employees [2] as well as ensue positive competition among companies [3].

The mentioned applications require mobile users to prove successive location points that they visited in past. To achieve this purpose, users will submit a series of location proofs using the data drawn from smartphone sensors, such as GPS [4], accelerometer or magnetometer to claim their presences to a third-party verifier. For convenience, in this work, we define a sequence of past location points at corresponding time points as a mobility trace of the user.

Using GPS for continuously recording the mobility trace will drain the battery of the smartphone quickly. We need to consider other energy-efficient sensors. A lot of prior works have studied on indoor [5]–[7] and outdoor [8]–[10] localization with low power sensors. However, the sensor data can be easily faked by malicious users. Therefore, a secure authentication scheme is required to provide the integrity property. Previous works have mainly deployed two mechanisms to authenticate the sensor data drawn from mobile devices. First, they use co-located wireless infrastructure (i.e., APs) to sign users’ GPS data for location proofs [11], [12]. However, this mechanism becomes infeasible in the scenarios lacking trusted wireless infrastructure. Second, trusted hardware, such as a Trusted Platform Module (TPM) [13], [14], is introduced to sign the sensor data to generate location proofs in users’ devices. Besides the extra cost of hardware, this scheme introduces additional computational and delay overhead for operating in trusted computing environments [15]. Furthermore, it’s not flexible to provide differential location privacy levels controlled by the user due to rigid-cryptographic mechanisms used in TPMs [16], [17].

A digitally signed proof of user’s location at some particular time is denoted as a Secure Location Proof (SLP). In this paper, we present the design and evaluation of a scheme, CLIP, to provide secure, efficient and Continuous Location Integrity and Provenance. It mostly relies on common low power sensors like accelerometer, and leverages existing resources, such as public road maps, available wireless APs or co-located mobile devices, for continuous location verification. CLIP aims at providing the integrity of users’ mobility traces. Without using any special hardware, it constructs a commitment of a user’s mobility trace with hash chains or chained Merkle Hash Trees (MHTs) [18], based on the pre-known trajectory or the only information of the user’s average mobility speed. Moreover, CLIP protects users’ location privacy. A mobile user is allowed to submit the tailored sensor data to provide differential location privacy granularity for verification. No more location information could be learned by a verifier than the user intends to reveal. To target more applications, our CLIP is distributed. Co-located mobile devices or available wireless APs generate SLPs for the mobility trace. We also propose a spatial-temporal trust model to resist collusion attacks that fake SLPs are generated for fake spatial-temporal information.

The main features of CLIP are as follows:

- An entropy-based commitment mechanism is proposed to provide the integrity of the mobility trace without any cost of trusted hardware. After generating the commitment, even the mobile user himself cannot fabricate or modify his mobility trace.
- CLIP mostly relies on energy-efficient sensors such as accelerometer to record a user’s mobility trace. Moreover, the entropy-based commitment mechanism is light-weight as it is built on symmetric cryptography, e.g., hash functions.

- CLIP protects users’ location privacy. Mobile users are given the control over the differential location granularity level that is shown to the verifier.

- CLIP defends against collusion attacks. A spatial-temporal trust model is made use of to detect users generating forged SLPs with co-located mobile devices.

To the best of our knowledge, this is the first work that provides a secure protocol for continuous location proofs. Our simulations show that our spatial-temporal trust model can achieve over 90% detection accuracy even under high probability of collusion attacks. Our Android implementation shows that CLIP is an efficient protocol with fairly low computational and storage cost.

The rest of the paper is organized as follows. Section II discusses related work. Section III gives the system model and security model. In Section IV, we describe the details of our CLIP protocol. Section V provides a security analysis of CLIP. In Section VI, we present experimental results. Finally, a discussion is outlined in Section VII.

II. RELATED WORK

Several location verification schemes have been proposed to prove the presence of a mobile user at a particular place and time. These works can be mainly divided into two categories depending on the network architecture considered: centralized [11], [12] and distributed [19]–[21].

Saroiu et al. [11] propose a centralized protocol to create SLPs by trusted wireless APs. However, their scheme does not consider how to protect users’ privacy while submitting the SLPs. To provide privacy protection, Luo et al. [12] make use of hash function and public key cryptography in their location proof architecture named VeriPlace. VeriPlace can only detect location spoofing when the two locations claimed in location proofs are impossible in space-time domain for an individual.

Distributed protocols rely on colocated users and not only on some trusted centralized infrastructure. Zhu et al. [19] propose APPLAUS. To protect the location privacy, mobile users periodically change their pseudonyms. To further prevent colluding attackers from generating bogus location proofs, APPLAUS uses betweenness ranking and correlation clustering-based approaches. However, periodically changing pseudonyms introduces high storage and management overhead for Certificate Authority in APPLAUS. Moreover, their detection approaches perform well only when the probability of collusion attack is no more than 0.1.

Wang et al. [20] propose STAMP, where users can obtain SLPs for different levels of location granularity. To defend against collusion attacks, they propose an entropy-based trust evaluation approach based on the recorded location proof transaction between two users. However, as their SLPs are constructed for one location point, their scheme cannot be applied to continuous location verification. Besides these works, Hasan et al. [21] generate the secure location provenance with hash chains or Bloom filters based on the wireless communication with wireless AP and co-located users. The OTIT protocol [22] presents a secure model to design order preserving location chains with formal propositional logical proofs. Both schemes mainly use cryptography mechanisms to defend against forgery of location chronology. However, since there are no detection mechanisms in these schemes, location information could be forged by malicious users.

None of these works provision for a secure scheme for continuous location provenance.

III. SYSTEM MODEL AND SECURITY MODEL

It may not be possible to obtain SLPs from wireless APs in all the scenarios such as walking outdoors. Thus, we consider a distributed architecture, i.e., mobile users obtaining SLPs from nearby peers or available AP.

A. System Model

The architecture of CLIP is illustrated in Fig. 1. We have five types of parties according to their roles:

- **Prover**: A prover is one mobile user with a mobile device, who tries to generate a secure mobility trace, and obtain SLPs of the mobility trace during his movement.

- **Wireless AP**: A wireless AP is trusted and used to initiate CLIP, which can be chosen by the prover. It will sign the commitment of the mobility trace for the prover, and may act as a witness.

- **Witness**: A witness is one nearby mobile user or stationary wireless AP, who agrees to create a SLP when he receives the prover’s request.

- **Verifier**: A verifier is the entity to whom the prover intends to claim his presences at continuous location points at specific time points.

- **Certificate Authority (CA)**: The CA is one semi-trusted server which manages cryptographic credentials and verifies signatures for other entities.

A prover will communicate with available wireless AP by Wi-Fi, and with co-located mobile devices over Bluetooth or Wi-Fi. Sensor data and secure location proofs will be sent to the verifier by Internet from provers. Each mobile user acts as a
prover/witness just relying on his role at the moment. Verifiers are considered to use Internet to communicate with CA. Prior work in indoor and outdoor navigation [5]–[10] has lead to algorithms for translating sensor readings into movements in physical plane, and it can be incorporated as is, to our work.

B. Security Requirements

In this paper, we want to provide security assurance for continuous location verification. To achieve this goal, we need to construct secure provenance with both users’ mobility traces based on energy-efficient sensors, and SLPs generated by witnesses. Therefore, we need to ensure important security properties of mobility traces as well as SLPs.

Security of mobility trace: 1. Authentication: The verifier must be able to authenticate or verify the truth of set of locations claimed by the user. Malicious users should not be allowed to fake or modify the mobility trace.

2. Privacy: The prover should have his own choice to reveal a coarse location granularity level to the verifier. Moreover, the prover’s mobility trace should not be disclosed to CA when CA verifies the prover’s identity.

Security of SLPs: 1. Anonymity: It will put threats to a mobile user that giving the identity information to an untrusted entity. A prover’s identity should be able to be hidden from a witness. As a witness who is willing to generate a SLP is in proximity with a prover, the witness must be able to hide his identity from the prover.

2. Non-transferability: This property requires that a prover cannot occupy legitimate SLPs for another prover.

3. Non-repudiation: It allows a prover to prove to a third entity that one witness is accountable for generating the SLP. Without non-repudiation, a malicious witness can claim that another entity created the SLP.

C. Threat Model

We assume a user’s identity is connected to a pair of his public and private keys, which are created when the user registers with CA. Users will never disclose their private keys to others. A malicious prover may try to create a fake mobility trace and fake SLPs without physically existing at these location points. It includes generating forged mobility trace by himself, tampering with the location or time information of his proofs, and stealing another user’s SLPs. In addition, a malicious prover may also seek to acquire a witness’s identity during the process of SLPs collection.

A malicious witness may attempt to obtain a prover’s identity and repudiate a SLP generated by him. A verifier may want to obtain more location information than the prover wants to reveal. A wireless AP is considered to be secure to initiate CLIP, and trusted when it is available to act as a witness. We consider CA to be semi-trusted. It means that it is trusted in terms of honestly executing its tasks, i.e., credential management and signature verification. However, we assume CA intends to compromise users’ location privacy. Wormhole attacks [23], attacks via communication links (e.g., jamming attacks), and DoS attacks are out of the scope of this paper.

IV. THE CLIP SCHEME

A. Preliminaries

CLIP uses commitments to construct the secure authentication scheme and ensure the prover’s privacy. In a commitment scheme, one entity is able to commit to a message while hiding it to others, and later disclose the committed value.

In CLIP, the prover commits to a mobility trace using either a hash chain or chained MHTs depending on knowledge of mobility path. We present the detailed process of commitment generation in Section IV-C.

We denote a commitment to the prover’s identity $ID_p$ as $C(ID_{p}, r_p)$, where $r_p$ is a nonce for randomizing the commitment. The commitment could be verified after both $ID_p$ and $r_p$ are disclosed by the prover. There are many proposed commitment schemes [24], [25], and CLIP works with all of them. All cryptographic notations we use are listed in Table I.

B. Protocol Overview

CLIP relies on energy-efficient sensors to collect the location data of a mobility trace. To securely verify continuous location points, CLIP includes four phases, as shown in Fig. 2. To formulate this problem, we assume that the prover begins at $L_0$ at time $T_0$. At a high level, each prover in CLIP divides its timeline into a sequence of time events, and we call them mobility intervals. A mobility trace will contain $n$ location points for $n$ mobility intervals, where $n ≥ 1$.

1) Commitment Generation: At the beginning of location $L_0$, a prover chooses his destination, and predicts his mobility trace of next $n$ mobility intervals based on public road maps, means of transportation (e.g., vehicle or walking) or gives a probabilistic estimate of locations for next $n$ mobility intervals. The prover generates a commitment on the predicted mobility trace, and submits the commitment to a nearby wireless AP, who will sign it to ensure that no users can fake or modify the mobility trace any more. Since the prover can choose an arbitrary begin time, the assumption of presence of a trusted wireless AP is justified.

2) SLPs Collection: After generating the commitment, the prover starts to move as estimated. During a mobility interval or some mobility intervals, the prover collects SLPs from peers or available APs. Therefore, the prover finally stores $m$ SLPs of the full mobility trace.

3) Signature Generation: To protect the integrity of the mobility trace, the prover will generate the signatures of sensor data for $n$ mobility intervals. During this process, the prover may want to tailor his sensor data for location privacy protection, i.e., only revealing a coarse location point. We discuss how the prover addresses this issue in Section IV-G.
4) Continuous Location Verification: In this phase, the prover is ready to choose sensor data which are necessary for continuous location verification. He then submits the commitment, chosen/tailored sensor data, relevant signatures, and selected SLPs to the verifier. Upon receiving the verification message, the verifier first communicates with CA to verify the identity of AP, prover and witness. It then verifies and evaluates the location points provided by the prover.

C. Commitment Generation

We begin by assuming that a prover stands at \( L_0 \), near a trusted wireless AP. First, we consider a simple case that the prover knows the clear mobility trajectory for next \( n \) intervals according to public road maps, which could run an application of navigation in his mobile phone. Next, we consider the case when prover does not get the information of trajectory, and just decides his average mobility speed based on the means of transportation and possible directions.

1) Pre-known Mobility Trace: The prover generates a claim or commitment to the trajectory \( L = L_1, \ldots, L_n \) he plans to take. As shown in Fig. 3, the prover first generates the hash of each location point: \( F_i = H(L_i | R_i | T_i), 0 < i \leq n \), where \( R_i \) is a random value, and \( T_i \) is the expected time of \( L_i \) in the predicted mobility trajectory. Then, the prover generates hash chains as follows: \( S_n = H(F_n) \), and

\[
S_i = H(F_i | S_{i+1}), 1 \leq i \leq n - 1. \tag{1}
\]

Finally, the prover publishes his commitment for the \( n \) mobility intervals:

\[
C_{MT} = H(S_1). \tag{2}
\]

2) Un-known Mobility Trace: When the trajectory is unknown, we try to create an estimated path with set of probabilities. For example, a user biking at 10 mph will have specific probabilities of physical movement in a time epoch or mobility interval. His physical movement is limited to the length of interval and speed. Thus, it is possible to generate a movement prediction table (\( PT_i \)) where each entry presents one possible movement between the mobility interval \( i - 1 \) and \( i \).

To compress the amount of the movement information, the prover will use a local coordinate to express his future positions [26]. We set the origin of the local coordinate at the beginning position \( L_0 \). A pair of orthogonal vectors (i.e., \( \Delta x \) and \( \Delta y \)) are also chosen, the scalar of which can be defined based on a desired level of positioning accuracy. For example, \( |\Delta x| \) and \( |\Delta y| \) can be decided to 2 meters for accurate localization in indoor [5] and some outdoor environments [4], [9], [26]. The prover’s future location \( L_i \) in the mobility interval \( i \) can be presented as: \( L_i = L_0 + a_i \Delta x + b_i \Delta y \), where \( a_i \) and \( b_i \) are rounded to integers. Therefore, every movement from the interval \( i - 1 \) to the interval \( i \) is shown as:

\[
D_i = L_i - L_{i-1} = (a_i - a_{i-1}) \Delta x + (b_i - b_{i-1}) \Delta y. \tag{3}
\]

which can be given by a pair of integers \((a_i - a_{i-1}, b_i - b_{i-1})\) for short. \( PT_i \) collects all the possible results of \( D_i \), which changes with time, location and mobility speed. By combining several prediction tables, the prover could model his future location not only in this mobility interval but also in many mobility intervals.

With movement prediction tables, the prover generates a commitment \( C_{MT} \) using the structure of chained Merkle Hash Trees from \( T_0 \) to \( T_n \). A MHT is a binary tree structure in which each leaf node is assigned a value and an inner node is assigned the hash of its two children. As shown in Fig. 4, for an entry \( D_{ik} \) in the movement prediction table, there is a leaf labeled as \( h_{ik} = H(D_{ik} | R_{ik} | T_i) \) in the MHT, where \( R_{ik} \) is a random value.

The chained MHTs contain \( n \) MHTs linked chronologically, and the chain’s anchor is the commitment \( C_{MT} \). We present the resulting tree structure in Fig. 4. The root of a single MHT is denoted as \( S_i \), and its two children as \( l_i \) and \( f_i \). In the chained structure,

\[
S_i = H(l_i | f_i | S_{i+1}), 1 \leq i \leq n - 1. \tag{4}
\]

Finally, the commitment is computed as:

\[
C_{MT} = H(S_1). \tag{5}
\]

After generating the commitment, the prover starts to encrypt a message \( MI = \{L_0 | T_0 | C_{MT} \} \) with a key marked as \( r_0 \). Then, the prover sends a message \( MQ | r_0 \) to a nearby AP (denoted as \( AP_0 \)):

\[
MQ = C(ID_p, r_p) | E(MI, r_0) | E_{SK_p}(H(E(MI, r_0))),
\]

where, \( ID_p \) is the prover’s identity, \( r_p \) is a random nonce. For an un-known mobility trace, prover also adds the information of local coordinates, \( \Delta x \) and \( \Delta y \), into \( MI \).

Upon receiving the message \( MQ \), \( AP_0 \) checks the prover’s initial location \( L_0 \) and time \( T_0 \). The physical proximity of prover can be verified by wireless AP (or witness in later sections) with distance bounding protocols [20], [23], [27]. For brevity, we do not consider it in this paper. If \( AP_0 \) decides to accept the prover’s request, it signs \( MQ \) with its private key.
Determines the Movement prediction table $PT_l$, and then construct chained MHTs. In a MHT, each leaf node is associated with one entry in $PT_l$, and each inner node is calculated by hashing the two children.

$SK_{CA_{j,C_{0}}}$, encrypts it with the public key of $CA$, and sends back a message $MR$ to the prover:

$$MR = E_{PK_{CA}}(ID_{CA_{j,C_{0}}},MQ|E_{SK_{CA_{j,C_{0}}}}(H(MQ))).$$  

(6)

D. SLPs Collection

After receiving $MR$, the prover begins to move from location $L_0$ to location $L_n$. The sensors in his phone will collect the location parameters, mainly the acceleration readings (denoted as $\{A_p\}_{T_0 - T_0}$).

Determine the Movement Prediction Table

During the movement, the prover starts the process of SLPs collection from $L_0$ to $L_n$. The sensors in his phone will collect the location parameters, mainly the acceleration readings (denoted as $\{A_p\}_{T_0 - T_0}$).

$$S_{\Delta} = H(h[f_i]S_{\Delta})$$

(5)

As the sibling nodes on the path from this leaf node to the root of the MHT.

When a witness receives a $SQ_j$, he makes a decision if he accepts the request. If the witness accepts it, he creates a SLP response message (denoted as $SR_{j,w}$), including the encrypted location proof (denoted as $IP_{j,w}$):

$$SR_{j,w} = E_{PK_{C_{j,w}}}(ID_{j,w}|IP_{j,w}|E_{SK_{C_{j,w}}}(H(P_{j,w}))),$$  

(7)

$$P_{j,w} = IP_{j,w}|SQ_j = E(L_j,r_{j,w})|C(ID_{j,w},r_{j,w})|T_j,$$  

(8)

where $ID_{j,w}$ is the identity of the witness, $L_j$ is the current location, $r_{j,w}$ is a random nonce used to encrypt $L_j$.

A SLP is then created with a $SR$ and a random nonce. For the $j$th proof collection interval, we assume that there are $w$ SLPs collected from $w$ witnesses:

$$SLP_j = (SR_{j,1},r_{j,1}, \cdots, SR_{j,w},r_{j,w}).$$  

(9)

Finally, the prover locally stores all these SLPs of the mobility trace: $\{SLP_1, SLP_2, \cdots, SLP_w\}$.

E. Signature Generation

The verifier may request prover for continuous location verification over a sequence of locations, such as $L_{d-1}, L_{d-1+1}, \cdots, L_d$ from time $T_{d-1}$ to $T_d$. After receiving the request, the prover prepares sensor data $\{A_p\}_{T_0 - T_d}$ with corresponding mobility interval $d$, and generates their signatures to guarantee authenticity.

1) Pre-known Mobility Trace: For mobility interval $d$, being at location $L_d$ at time $T_d$, the prover publishes random values $\{R_{c}\}_{0 < c \leq d}$ generated for each location point during commitment generation, and the hash value $S_{d+1}$ as signatures of sensor data collected from $T_0$ to $T_d$. Let $S_{n+1} = 0$, and we format the signatures as:

$$\Upsilon_{0,d} = \{R_0|R_1|\cdots|R_d|S_{d+1}\}. \quad (10)$$

2) Un-known Mobility Trace: For mobility interval $d$, based on raw sensor data, the prover first locates the leaf node corresponding to the entry of $PT_l$ in chained MHTs, where $0 < c \leq d$. The prover then gets the random value and off-path nodes of this leaf as signatures. We define off-path nodes as the sibling nodes on the path from this leaf node to the root of the MHT.

For example, in Fig. 4, if the prover moves to $L_0 = L_9 + D_{12}$, associated with the entry $D_{12}$ of $PT_l$ at time $T_k$, the off-path nodes include $h_{11}$ and $f_1$. For $L_1$, the prover publishes the random nonce $R_{12}$ along with the off-path nodes as the signature: $\xi_1 = \{R_{12}|h_{11}|f_1\}$.

Similarly, the signatures of sensor data recording the mobility trace from $T_0$ to $T_d$ should include $\{\xi_0|\cdots|\xi_d\}_{0 < c \leq d}$, and the hash value $S_{d+1}$. Let $S_{n+1} = 0$, and we format the signatures as:

$$\Upsilon_{0,d} = \{\xi_0|\xi_1|\cdots|\xi_d|S_{d+1}\}$$.  

(11)

F. Continuous Location Verification

1) mCLP Verification: Prover: Based on recorded sensor data (e.g., $\{A_p\}_{T_0 - T_d}$), the prover generates the signatures (e.g., $\Upsilon_{0,d}$) in the last step. To further provide credibility for the verified location points, he needs to find the collected SLPs in or near the mobility interval $d$. In our setting, the prover decides to choose the SLPs generated by witnesses in $d$th proof collection interval if $T_0 \leq T_d \in [T_d - \delta, T_d]$, where $\delta$ is the time parameter set by the prover.

Therefore, to verify his location points, the prover includes the sensor data, $MR$, signatures and SLPs in a message of continuous location proofs for verification (denoted as mCLP) as follows:

$$mCLP = EP|SLP_z|ID_p|r_p|N_p|L_{d-1} \cdots \{L_d|M_R|\Upsilon_{0,d}$

where $EP = E_{PK_{CA}}(ID_p|IP_{j,w}|E_{SK_{CA}}(H(P_{j,w})))$ and $P_{p_e} = E_{P_{CA}}(A_p|T_0 - T_0, N_p)$. In $P_p$, sensor data are encrypted by the prover with a random nonce $N_p$.

Verifier: When the verifier receives $mCLP$, it first needs to authenticate the identity of AP, the prover and the witness with the assistance of CA. The verifier then constructs an identity verification request (denoted as $ID_{Req}$) to CA:

$$ID_{Req} = EP|SR_{z,1}|\cdots|SR_{z,w}|ID_p|r_p|MR,$$  

(12)
where \( SR_{z,1}, \ldots, SR_{z,w} \) are contained in \( SLP_z \).

**CA:** Upon receiving \( IDReq \), CA decrypts these messages and then performs the verification. It is responsible for checking the signatures of the prover, witness and wireless AP with their public keys, and checking that \( C(ID_p, r_p) \) could be decommitted with \( r_p \) and \( ID_p \).

If \( IDReq \) fails the verification, CA replies to the verifier with an identity verification response (denoted as \( IDRes \)) using a one-bit notification. Otherwise, CA extracts encrypted commitment from \( MI \), sensor data from \( EP \), location proofs from \( SR_{z,1}, \ldots, SR_{z,w} \), and sends \( IDRes \) back to the verifier along with its signature:

\[
IDRes = E(\{MI, r_0\}|P_p|IP_{z,1} \cdots |IP_{z,w}|T_z),
\]

where \( IP_{z,1}, \ldots, IP_{z,w} \) are extracted from \( SR_{z,1}, \ldots, SR_{z,w} \), respectively. Moreover, if there is an \( IP \) generated by a wireless AP, CA notices the verifier with one more bit.

**Verifier:** After receiving \( IDRes \), the verifier decrypts \( MI \), and get \( C_{MT}, L_0 \) and \( T_0 \). The verifier now performs the following operations:

- **Movement Recovery:** By decrypting \( P_p \) with \( N_p \), the verifier gets raw sensor data \( \{A_{p}\}_{T_{d}-T_0} \), and thus recovers a sequence of movements \( \{D_k\}_{0<d} \).

- **Signature Verification:** According to the movements \( \{D_k\}_{0<d} \) and \( C_{MT} \), the verifier starts to check the signatures \( T_{0,d} \). For a pre-known mobility trace, the verifier reconstructs the location \( L_v \) at the mobility interval \( v \), with \( L_0 \) and the movements \( \{D_k\}_{0<d} \) recovered from the raw sensor data. \( S_v \) is then computed for \( L_v \), based on \( S_{v+1} \) and \( R_v \) obtained from \( T_{0,d} \). Given the root \( C_{MT} \) of hash chains, the verifier checks whether Equation (1) holds by repeatedly using the hash function. For an unknown mobility trace, similarly, the verifier checks the signature of the movement \( D_v \) according to \( \xi_v \) and \( S_{v+1} \) for the mobility interval \( v \). Given the root \( C_{MT} \) of the chained MHTs, it checks whether Equation (4) holds by repeatedly using the hash function.

- **IP Opening:** If the sensor data have passed the verification, the verifier opens \( IP_{z,1} \cdots |IP_{z,w} \) from \( IDRes \). It now decrypts them with \( \{r_{z,1}, \ldots, r_{z,w}\} \) contained in \( SLP_z \) of \( mCLP \), and obtains the location information of \( L_z \) at \( T_z \). An inconsistent \( L_z \) invalidates the corresponding IP.

2) **Collusion Detection:** Based on the collected IPSs, the verifier starts a trust evaluation and obtains a collusion detection result of location \( L_{d-u}, \ldots, L_d \). The intuition is that, with the commitment of the mobility trace, if the prover stood at \( L_z \) seen by one trusted witness or a number of witnesses at time \( T_z \), the prover has a high probability at \( L_{d-u}, \ldots, L_d \) near \( L_z \) at time \( T_{d-u}, \ldots, T_d \) during this movement. Thus, we consider three factors that affect the trust value. First, an IP is generated by a wireless AP or mobile device. We define a weighting parameter (denoted as \( \lambda \)) for the witness. Second, we measure the distance between \( L_z \) and \( L_{d-u}, \ldots, L_d \). At last, time is regarded as one critical factor, since location proofs collected at expected time are usually the most accurate.

![Fig. 5. Differential location granularity level for privacy protection.](image-url)

We define the trust \( \Gamma_L \) based on the collected IPSs:

\[
\Gamma_L = \lambda \cdot e^{-(L_d-L_u)} | \alpha \cdot e^{-\beta},
\]

where \( \alpha \) is defined as the location parameter that controls the distance factor’s weight in the system, and \( \beta \) is defined as the time parameter that controls the time factor’s weight. Finally, the verifier makes a decision by comparing \( \Gamma_L \) with a trust threshold, which is denoted as \( \Theta \).

These IPSs may contain other factors that affect the trust of the verified locations. In this paper, only a general solution is given based on the model that we propose. For a specific application, it is also possible to consider one or more of the factors. System designer could easily modify our trust model according to applications’ needs.

**G. Privacy Discussion**

To protect location privacy, the prover may only want to submit tailored sensor data to reveal the verifier a coarser location granularity level. Now, we discuss how to address this privacy issue.

For each location, we consider there are \( g \) location granularity levels. We denote them as \( L_d(1), L_d(2), \ldots, L_d(g) \) for location \( L_d \) at time \( T_d \). \( L_d(1) \) indicates the finest location granularity level, such as an exact geographic coordinate. \( L_d(g) \) indicates the coarsest location granularity, such as a city.

To show differential location granularity level, the prover tailors the raw sensor data after he receives the location verification request. It is observed that the sensor data farther away the time \( T_d \) is harder to estimate the exact location of \( L_d \). If the prover tailors the sensor data around \( T_d \), a fine location granularity level of \( L_d \) is shown.

As illustrated in Fig. 5, to only disclose a coarse location granularity level \( L_d(3) \), the prover extracts his sensor data as \( \{\hat{A}_p\}_{T_{u}-T_0} \), where \( 0 < u < d \). To generate the signature of \( L_d(3) \), the prover performs the phase of signature generation with the mobility interval \( u \) instead of \( d \), to authenticate \( \{\hat{A}_p\}_{T_{u}-T_0} \). The process of continuous location verification performed by the verifier is the same as before.

To protect the privacy of mobility trace, the prover may also want to hide portion of his trajectory (e.g., from \( L_0 \) to \( L_u \)) from the verifier. In our CLIP, when there are SLPs generated by wireless APs during the prover’s movement, we find that the prover could securely tail his mobility trace arbitrarily if the submitted trajectory starts or ends with a trusted location, which is in proximity with an AP. We will have a detailed discussion of this issue in our future work.
V. Security Analysis

In this section, we analyze and prove that CLIP can achieve the mentioned security properties.

**Proposition 1** Assuming that hash chains are secure, an attacker or even a prover cannot forge legitimately authenticated data after generating the commitment of the mobility trace.

Let us assume that an attacker tries to create a new location point \( L_\eta \) instead of true location \( L_\eta \) such that \( L_\eta \neq L_\eta \) but \( C_{MT} = C_{MT} \). We will show that this is possible only when there is a collision in hash function. For a pre-known mobility trace, the attacker uses a structure similar to Fig. 3 to construct hash chains for \( C_{MT} \). Let \( F_\eta = H(L_\eta | R_\eta | T_\eta) \). If \( F_\eta = F_\eta' \), there is a collision of hash function.

Else, let \( S_\eta' = H(F_\eta' | S_{\eta+1}' \rangle \). If \( S_\eta = S_\eta' \), then \( F_\eta' | S_{\eta+1} \rangle \) form another collision of hash function \( H \); otherwise, we have \( S_\eta \neq S_\eta' \). Similarly, we can discuss for \( \eta = 1, \eta - 2, \cdots, 1 \). The last assertion would be \( S_1 \neq S_1 \). Since \( C_{MT} = C_{MT} \), a collision of hash function must exist at some step. The proof for an unknown mobility trace is similar to the pre-known mobility trace, and it is skipped for the sake of brevity.

**Proposition 2** The location granularity level that a verifier obtains from a prover is the granularity level that he wants to disclose to the verifier.

To show one coarse location of \( T_d \), the prover sends the verifier his tailored sensor data \( \{A_p \}_{T_a \to T_0} \) with signatures \( Y_{0,u} \), where \( 0 < u < d \). The signatures only include the hash values of location points from \( T_u \) to \( T_d \) in \( S_{u+1} \). Due to the one-way property of hash function, the verifier cannot learn any more location information from \( L_a \) to \( L_d \).

**Proposition 3** No location information can be obtained by CA from \( ID_{Req} \).

Without knowing \( N_p, r_0 \) and \( r_{a,1} \cdots r_{a,w} \), CA cannot decrypt any location information from an \( ID_{Req} \). It can authenticate the identity of prover, witness and wireless AP, but learn no location of the prover and the witness with the \( ID_{Req} \).

Thus, CA has no access to the mobility trace but only to the encrypted location values.

**Proposition 4** A prover/witness cannot learn about the identity of a witness/prover.

During the phase of SLPs collection, the identity of the prover \( ID_p \) is committed as \( C(ID_p, r_p) \). As the witness does not know \( r_p \), he cannot obtain \( ID_p \). In a \( SR \), the identity of the witness \( ID_w \) is encrypted with \( PK_{CA} \). Since CA’s private key is not owned by the prover, he cannot decrypt \( SR \) and acquire \( ID_w \).

**Proposition 5** For different events of SLPs collection, a witness cannot link SQs generated by the same prover.

For different mobility traces, a prover may use different \( r_p \). Even a number of SQs from the same prover are received by a witness at different time points, the witness cannot obtain any more information by linking these SQs.

**Proposition 6** For different events of SLPs collection, a prover cannot link SLPs sent by the same witness.

A witness may choose different \( r_{a,w} \) for different \( SR_{a,w} \). Moreover, a \( SR \) created by the witness is encrypted with \( PK_{CA} \). Even a number of SLPs from the same witness have been received by a prover at different time points, the prover cannot get any more information by linking them.

**Proposition 7** A SLP generated for a prover cannot be used by another prover.

If a prover gives a verifier his own \( ID_p \) and another prover’s \( ID_p \), CA is able to detect that \( ID_p \) in the \( ID_{Req} \) from the verifier does not match \( C(ID'_p, r'_p) \) in the \( SR \).

**Proposition 8** The location or time information cannot be modified in a SLP.

The spatial-temporal information is included in each \( P \) and signed by a witness’s private key, shown as \( E_{SK_w}(H(P)) \). As others do not own the witness’s private key, they cannot change the location or time information in the SLP.

**Proposition 9** A SLP created by a witness cannot be repudiated.

According to the assumption that a user will never disclose his private key, his signature will guarantee the property of non-repudiation of a SLP.

VI. Experiments and Results

A. Simulation

1) Simulation Setup: We implement the spatial-temporal trust model with C simulation. In our simulation, we test the number of 1000 users, who are moving with a random mobility model. The proportion of attackers among the users is changed from 1% to 10%. We randomly select a prover among all users, who collects SLPs from nearby witnesses. We assume the distribution of mobile devices follows a Gaussian distribution with mean value of \( \mu_w \) and standard deviation of \( \sigma_m \), and the distribution of APs with mean value of \( \mu_a \) and standard deviation of \( \sigma_a \). The distance between a witness and the prover is modeled by a uniform distribution \( U([0, \phi_w]) \).

2) Simulation Results: First, we conduct extensive tests to check the accuracy with different choices of trust model parameters, maximum time gap \( \delta \) and trust threshold \( \Theta \). The results are shown in Fig. 6. As the location parameter \( \alpha \) and time parameter \( \beta \) work similarly, we change \( \alpha \) and \( \beta \) together and see the impacts. With a large value of \( \delta \), accuracy decreases since it implies that attackers may take more time for successful collusion, and the malicious user for forging.

### Table II. Simulation Parameters

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Value</th>
<th>Parameter</th>
<th>Value</th>
</tr>
</thead>
<tbody>
<tr>
<td>Location parameter ( \alpha )</td>
<td>0.002</td>
<td>Time parameter ( \beta )</td>
<td>0.002</td>
</tr>
<tr>
<td>Trust threshold ( \Theta )</td>
<td>0.6</td>
<td>Maximum time gap ( \delta )</td>
<td>10</td>
</tr>
<tr>
<td>Proportion of attackers ( p_{atk} )</td>
<td>2%</td>
<td>Prob. collusion attacks ( p_{col} )</td>
<td>0.2</td>
</tr>
<tr>
<td>Max. distance ( (m_w) )</td>
<td>10</td>
<td>Mean (mobile devices) ( (\mu_m) )</td>
<td>5</td>
</tr>
<tr>
<td>S.d. (mobile devices) ( (\sigma_m) )</td>
<td>2</td>
<td>Mean (APs) ( (\mu_a) )</td>
<td>1</td>
</tr>
<tr>
<td>S.d. (APs) ( (\sigma_a) )</td>
<td>0.2</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>
possible mobility trace. With a different choice of $\delta$, we have to set a different $\Theta$ to achieve a high accuracy rate. From these simulations, we choose $\Theta = 0.6$ and $\delta = 10$ because this couple produces the best result.

Second, we study how the number of witnesses affects the detection accuracy. We change the values of $\mu_a$, $\mu_m$ and $\sigma_m$. The results show that the accuracy level improves when we have more mobile devices as witnesses involved in the events of SLPs collection. It is worthy noting that the detection accuracy is further enhanced when more SLPs are generated by wireless APs.

Finally, we investigate our trust model’s performance under different proportion of attackers $p_{att}$ and probability of collusion attacks $p_{col}$. It is evident that our trust model is resistant to collusion attacks and has a very high accuracy. For $p_{att} = 5\%$ and $p_{col} = 0.4$, the model has accuracy above 0.9. The accuracy is high ($> 0.9$) even for $p_{col} > 0.8$, when $p_{att} < 2\%$.

B. Prototype Implementation

1) Implementation Setup: A prototype application is implemented on the Android platform with Java. We perform the experiments based on two Samsung Exhibit II devices, which are equipped with 1GHz chipset, 1GB ROM, 512MB RAM, accelerometer, GPS, compass, Bluetooth, Wi-Fi, and running Android. We use Bluetooth for communication between two mobile devices. SHA1 is used as the one-way hash function and AES with a 128-bit key as symmetric key encryption algorithm. We use RSA as the public key encryption algorithm, and deploy DSA for authentication.

CLIP senses the user’s movement conditions by using low power sensors, such as accelerometer and compass. Meanwhile, the raw sensor data are stored on the user’s mobile phones for offline data analysis, and then used to recover the mobility trace. We study the computational time as the indicator of energy consumption, and storage resources that are required for CLIP implementation. As the verifier and CA that do mCLP verification are servers with high computational capacity, we concentrate on the phases of commitment generation, SLPs collection, and signature generation that are run on mobile devices. Each of the results is obtained on 10 runs of test. During the test, background processes were not allowed to run in parallel.

2) Implementation Results: To generate the commitment of the predicted mobility trace, we should consider the accuracy of localization according to these low power sensors. This is because more noise would be added in a finer location representation than useful information. In our implementation, we set the scalar of each vector to be 2 meters. We experimented with both the pre-known and un-known mobility trace. For the un-known mobility trace, three kinds of average mobility speed are chosen to represent driving, cycling and walking: 80 mph, 15 mph and 4 mph, respectively. We first evaluate the computational time and storage cost with different layers of MHTs. The duration of these traces is 15 minutes. The time is less than 10 seconds in terms of cycling and walking activities on a smartphone. With 5-Layer MHTs, Fig. 7 also shows the overhead for commitment generation under different duration of mobility trace. If the trajectory is clear, a commitment can be instantly created. For an un-known mobility trace, we confirm CLIP reduces the computation complexity from $O(2^{|L|\cdot n})$ to $O(n \cdot 2^{|L|})$, where $Q_L$ is the layer of MHTs and $n$ is the duration of the mobility trace.

In the process of SLPs collection, we use the commitment scheme proposed in [24] for identity commitment. We examine the impact of communication distance and key size on the application’s performance. As we use both DSA and RSA in the implementation, three key size pairs are tested. We find that the user spends less than 0.5 second to generate a SLP with 3072-bit RSA key and 1024-bit DSA key (See Fig. 8 for other key size). We also investigate the overhead of signature generation for the mobility trace of 15 minutes. Fig. 9 shows that the user can fast generate the signatures, and take almost the same computational cost under different verification requests.

VII. DISCUSSION

In this paper, we design CLIP with the goal of providing both security and privacy assurance for verification of continuous location points visited by a mobile user. To achieve this purpose, CLIP constructs secure provenance with both users’ mobility traces based on energy-efficient sensors in mobile phones, and SLPs generated by co-located wireless APs or mobile devices. To secure the mobility trace, CLIP constructs a
commitment with light-weight hash chains or chained MHTs. Moreover, CLIP provides the properties of anonymity, non-repudiation and non-transferability of a special location point with SLPs. To simultaneously protect users’ location privacy, CLIP enables mobile users to disclose differential location granularity level. A spatial-temporal trust model is used to detect the collusion attacks.

Our analysis and simulation results indicate that CLIP is be able to achieve the pre-defined security and privacy requirements, and resilient to collusion attacks. The implementation on Android shows that low computational and storage overhead is required for CLIP.

In this work, we construct MHTs using average mobility speed of user. It is possible to improve the construction of MHTs based on many factors, such as user’s current location, user’s past trajectory, time-of-day and traffic. In the event of SLPs collection, collection of more SLPs leads to a more accurate results. Continuously sensing for P2P connections for full mobility trace may be a time costly operation. This can be reduced by triggering sensing operation based on the discharging state of the mobile phone.

In the future, we will design SLPs with differential location granularity levels in witness endorsements, to enable more variety of applications. In CLIP, APs are not mandatorily required after the prover starts to move, although our trust model could be further improved with the presences of APs. In future, we will consider how the distribution of wireless infrastructure affects the trust values.

ACKNOWLEDGMENT

This paper was supported by China Scholarship Council (No. 201306230026). We would like to thank the anonymous reviewers for their helpful comments.

REFERENCES


