# STAMP: Enabling Privacy-Preserving Location Proofs for Mobile Users

Xinlei Wang, Amit Pande, Jindan Zhu, Prasant Mohapatra

***Abstract*—Location-based services are quickly becoming immensely popular. In addition to services based on users' current location, many potential services rely on users' location history, or their *spatial-temporal provenance*. Malicious users may lie about their spatial-temporal provenance without a carefully designed security system for users to prove their past locations. In this paper, we present the Spatial-Temporal provenance Assurance with Mutual Proofs (STAMP) scheme. STAMP is designed for ad-hoc mobile users generating location proofs for each other in a distributed setting. However, it can easily accommodate trusted mobile users and wireless access points. STAMP ensures the integrity and non-transferability of the location proofs and protects users' privacy. A semi-trusted Certification Authority is used to distribute cryptographic keys as well as guard users against collusion by a light-weight entropy-based trust evaluation approach. Our prototype implementation on the Android platform shows that STAMP is low-cost in terms of computational and storage resources. Extensive simulation experiments show that our entropy-based trust model is able to achieve high ($> 0.9$) collusion detection accuracy.**

***Keywords: Spatial-temporal provenance, location proof, privacy, trust***

## I. Introduction

As location-enabled mobile devices proliferate, location-based services are rapidly becoming immensely popular. Most of the current location-based services for mobile devices are based on users' current location. Users discover their locations and share them with a server. In turn, the server performs computation based on the location information and returns data/services to the users. In addition to users' current locations, there is an increased trend and incentive to prove/validate mobile users' past geographical locations. This opens a wide variety of new location-proof based mobile applications. Saroiu et. al described several such potential applications in [1]. Let us consider three examples: (1) A store wants to offer discounts to frequent customers. Customers must be able to show evidence of their repeated visits in the past to the store. (2) A company which promotes green commuting and wellness may reward their employees who walk or bike to work. The company may encourage daily walking goals of some fixed number of miles. Employees need to prove their past commuting paths to the company along with time history. This helps the company in reducing the healthcare insurance rates and move towards sustainable lifestyle. (3) On the battlefield, when a scout group is sent out to execute a mission, the commanding center may want every soldier to keep a copy of their location traces for investigation purpose after the mission.

The above applications require users to be able to obtain proofs from the locations they visit. Users may then choose to present one or more of their proofs to a third-party verifier to claim their presence at a location at a particular time. In this paper, we define the past locations of a mobile user at a sequence of time points as the *spatial-temporal provenance* (STP) of the user, and a digital proof of user's presence at a location at a particular time as an *STP proof*. Many works [1]–[3] in literation have referred to such a proof as *location proof*. In this paper, we consider the two terms interchangeable. We prefer "STP proof" because it indicates that such a proof is intended for past location visits with both spatial and temporal information. Other terminologies have been also used for similar concepts, such as location claim [4], provenance proof [5], and location alibi [6].

Today's location-based services solely rely on users' devices to determine their location, e.g., using GPS. However, it allows malicious users to fake their STP information. Therefore, we need to involve third parties in the creation of STP proofs in order to achieve the integrity of the STP proofs. This, however, opens a number of security and privacy issues. First, involving multiple parties in the generation of STP proofs may jeopardize users' location privacy. Location information is highly sensitive personal data. Knowing where a person was at a particular time, one can infer his/her personal activities, political views, health status, and launch unsolicited advertising, physical attacks or harassment [7]. Therefore, mechanisms to preserve users' privacy and anonymity are mandatory in an STP proof system. Second, authenticity of STP proofs should be one of the main design goals in order to achieve integrity and non-transferability of STP proofs. Moreover, it is possible that multiple parties collude and create fake STP proofs. Therefore, careful thought must be given to the countermeasures against collusion attacks.

In this paper, we propose an STP proof scheme named **S**patial-**T**emporal provenance **A**ssurance with **M**utual **P**roofs (STAMP). STAMP aims at ensuring the integrity and non-transferability of the STP proofs, with the capability of protecting users' privacy. Most of the existing STP proof schemes rely on wireless infrastructure (e.g., WiFi APs) to create proofs for mobile users. However, it may not be feasible for all types of applications, e.g. STP proofs for the green commuting and battlefield examples certainly cannot be obtained from wireless APs. To target a wider range of applications, STAMP is based on a distributed architecture. Co-located mobile devices mutually generate and endorse STP proofs for each other, while at the same time it does not eliminate the possibility of utilizing wireless infrastructures as more trusted proof generation sources. In addition, in contrast to most of the existing schemes which require multiple trusted or semi-trusted

X. Wang was with the Department of Computer Science University of California, Davis, CA 95616 USA. He is now with Facebook Inc. Menlo Park, CA, 94025 USA (xlwang@ucdavis.edu). A. Pande, J. Zhu, and P. Mohapatra are with the Department of Computer Science University of California, Davis, CA 95616 USA (e-mail: pande@ucdavis.edu; jdzhu@ucdavis.edu; pmohapatra@ucdavis.edu).

third parties, STAMP requires only a single semi-trusted third party which can be embedded in a Certificate Authority (CA). We design our system with an objective of protecting users' anonymity and location privacy. No parties other than verifiers could see both a user's identity and STP information (verifiers need both identity and STP information in order to perform verification and provide services). Users are given the flexibility to choose the location granularity level that is revealed to the verifier. We examine two types of collusion attacks: (1) A user $A$ who is at an intended location masquerades as another colluding user $B$ and obtains STP proofs for $B$. This attack has never been addressed in any existing STP proof schemes. (2) Colluding users mutually generate fake STP proofs for each other. There have been efforts to address this type of collusion. However, existing solutions suffer from high computational cost and low scalability. Particularly, the latter collusion scenario is in fact the challenging *Terrorist Fraud* attack [8], which is a critical issue for our targeted system, but none of the existing systems has addressed it. We integrate the Bussard-Bagga distance bounding protocol [9] into STAMP to protect our scheme against this collusion attack. Collusion scenario (1) is hard to prevent without a trusted third party. To make our system resilient to this attack, we propose an entropy-based trust model to detect the collusion scenario. We implemented STAMP on the Android platform and carried out extensive validation experiments. The experimental results show that STAMP requires low computational overhead.

The **contributions** of this paper can be summarized as:

1) A distributed STP proof generation and verification protocol (STAMP) is introduced to achieve integrity and non-transferability of STP proofs. No additional trusted third parties are required except for a semi-trusted CA.
2) STAMP is designed to maximize users' anonymity and location privacy. Users are given the control over the location granularity of their STP proofs.
3) STAMP is collusion-resistant. The Bussard-Bagga distance bounding protocol [9] is integrated into STAMP to prevent a user from collecting proofs on behalf of another user. An entropy-based trust model is proposed to detect users mutually generating fake proofs for each other.
4) STAMP uses a entropy-based trust model to guard users from prover-witness collusion. This model also encourages witnesses against selfish behavior.
5) Modifications to STAMP to facilitate the utilization of stationary wireless infrastructure APs or trusted mobile users are presented.
6) A security analysis is presented to prove STAMP achieves the security and privacy objectives.
7) A prototype application is implemented on the Android platform. Experiments show that STAMP requires preferably low computational time and storage.
8) Simulation experiments validate that our entropy-based trust model is able to achieve over 0.9 collusion detection accuracy with fairly high percentage ($\sim$5%) of colluding attackers.

The rest of the paper is organized as follows: Section II discusses related work. Section III describes our system model. In Section IV, we discuss the security requirements in detail and describe the threat model of this work. In Section V, we present the details of the STAMP protocol. Section VI provides an overview of how STAMP can be practically used in number of scenarios including trusted mobile users and wireless APs. A security analysis of STAMP against different types of attacks is provided in Section VII. In Section VIII, we describe our implementation and simulation and present our experimental results on the performance evaluation. We give a discussion and outline our future work in Section IX. Finally, Section X concludes the paper.

## II. Related Work

The notion of unforgeable location proofs was discussed by Waters et al. [10]. They proposed a secure scheme which a device can use to get a location proof from a location manager. However, it requires users to know the verifiers as a prior. Saroiu et al. [1] proposed a secure location proof mechanism, where users and wireless APs exchange their signed public keys to create timestamped location proofs. These schemes are susceptible to collusion attacks where users and wireless APs may collude to create fake proofs.

VeriPlace [2] is a location proof architecture which is designed with privacy protection and collusion resilience. However, it requires three different trusted entities to provide security and privacy protection: a TTPL (Trusted Third Party for managing Location in formation), a TTPU (Trusted Third Party for managing User information) and a CDA (Cheating Detection Authority). Each trusted entity knows either a user's identity or his/her location, but not both. VeriPlace's collusion detection works only if users request their location proofs very frequently so that the long distance between two location proofs that are chronologically close can be considered as anomalies. This is not a realistic assumption because users should have the control over the frequency of their requests.

Hasan et al. [5] proposed a scheme which relies on both location proofs from wireless APs and witness endorsements from Bluetooth-enabled mobile peers, so that no users can forge proofs without colluding with both wireless APs and other mobile peers at the same time. It eliminates the necessity of multiple trusted parties. Two privacy preserving schemes based on hash chains and Bloom filters respectively are described for protecting the integrity of the chronological order of location proofs. Li et al [11] proposed a token based scheme for location-based rewards. However, this scheme generates tokens and not location proofs, and works in presence of wireless AP. Pham et al. [12] propose location-based activity summary in a privacy preserving manner.

All the above systems are centralized, that is, they all require central infrastructures (wireless APs) to act as the location authorities and generate location proofs. However, we want to design a framework that can also work for distributed scenario where users are far from any trusted AP.

In Davis et al.'s alibi system [6], their private corroborator scheme relies on mobile users within proximity to create alibi's (i.e., location proofs) for each other. The security and privacy of the system is achieved based on a cryptographic commitment scheme. However, they do not deal with any collusion attacks. Also, multi-level location granularity is not considered in their work.

The system that is most closely related to our work is Zhu et al.'s APPLAUS [3]. It is a location proof system that is also based on co-located mobile devices mutually generating location proofs. In order to protect privacy, the knowledge of private information is separately distributed to three parties: a location proof server, a CA, and the verifier. Periodically

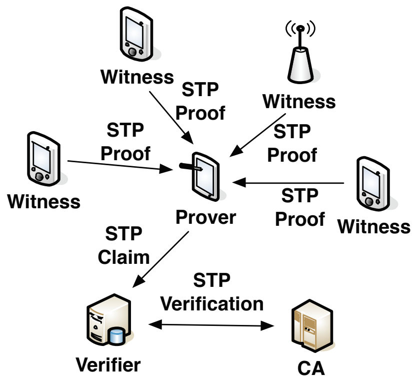


Fig. 1: An illustration of system architecture

changed pseudonyms are used by the mobile devices to protect their real identities from each other, and from the location proof server. We believe the location proof server is redundant for accomplishing the goals. Periodically changed pseudonyms incurs high operational overhead because of the requirement for highly cautious management and scheduling. Dummy proofs have to be regularly generated in order to achieve the privacy properties, which also incurs high communication and storage overhead. The collusion detection in APPLAUS is based on betweenness ranking and correlation clustering. These approaches require the location proof server to have access to at least the majority of the concurrent (within a short delay) location proofs at the same location (within a small region). This needs users to submit their location proofs right after generating them, which is infeasible when there is no Internet connection on-the-spot. Moreover, these approaches cost large computing power to run the detection (>200 seconds for 5000 pseudonyms) and their successful detection ratio is high ($> 0.9$) only when the percentage of the colluding attackers is rather low ($< 0.1\%$).

A preliminary result of this effort was presented in [13].

## III. System Model

As we explained, wireless infrastructure may not be available everywhere and hence a system based on wireless APs creating STP proofs would not be feasible for all scenarios. In addition, the deployment cost would be high if we require a large number of wireless APs to have the capability of generating STP proofs. Therefore, we think a distributed STP proof architecture, i.e., mobile users obtaining STP proofs from nearby mobile peers, would be more feasible and appropriate for a wider range of applications. We design a generic decentralized protocol, and then show how it can work well for centralized case also.

Figure 1 illustrates the architecture of our system. There are four types of entities based on their roles:

- **Prover**: A *prover* is a mobile device which tries to obtain STP proofs at a certain location.
- **Witness**: A *witness* is a device which is in proximity with the prover and is willing to create an STP proof for the prover upon receiving his/her request. The witness can be untrusted or trusted, and the trusted witness can be mobile or stationary (wireless APs). Collocated mobile users are untrusted.
- **Verifier**: A *verifier* is the party that the prover wants to show one or more STP proofs to and claim his/her presence at a location at a particular time.
- **Certificate Authority (CA)**: The *CA* is a semi-trusted server (untrusted for privacy protection, see Section IV-C for details) which issues, manages cryptographic credentials for the other parties. CA is also responsible for proof verification and trust evaluation.

A prover and a witness communicates with each other via Bluetooth or WiFi in ad hoc mode. A peer discovery mechanism for discovering nearby witness is required and preferably provided by underlying communication technology instead of our protocol. The proof generation system of prover is presented a list of available witnesses. When there are multiple witnesses willing to cooperate, the prover initiate protocol with them sequentially. STP claims are sent to verifiers from provers via a LAN or Internet, and verifiers are assumed to have Internet connection with CA. Each user can act as a prover or a witness, depending on their roles at the moment. We assume the identity of a user is bound with his/her public key, which is certified by CA. Users have unique public/private key pairs, which are established during the user registration with CA and stored on users' personal devices. There are strong incentives for people not to give their privacy away completely, even to their families or friends, so we assume a user never gives his/her mobile device or private key to another party.

## IV. Requirements and Challenges

Before introducing the details of our protocol, we first present and discuss the important issues and design challenges involved, in order to give an intuition of our objectives of constructing the protocol.

### A. Security

The security of STP proofs are two fold: *integrity* and *non-transferability*. The integrity property requires that no prover can create fake STP proofs by himself/herself or by collaborating with one or more other untrusted parties in the system. The non-transferability property requires that no prover can claim the ownership of another prover's legitimate STP proofs.

### B. Privacy

**Anonymity**: Location privacy is an extremely important factor that needs to be taken into consideration when designing any location based systems. Revealing both identity and location information to an untrusted party poses threats to a mobile users. First, a prover should be able to hide his/her identity from a witness. In addition, it is not only the prover's anonymity that we should pay attention to, a witness's anonymity should also be preserved. Since a witness who agrees to create an STP proof is co-located with the prover, his/her identity should not be revealed to the prover, either.

**Pseudonyms**: Pseudonyms are often used to provide anonymity. Nevertheless, if the same pseudonym is used by a mobile user, it is possible for an adversary to link multiple locations of the same pseudonym. By profiling and analyzing the user's location trace, the adversary could reveal the identity of the user or at least significantly reduce the anonymity set. True anonymity requires unlinkability [14]. Anonymity can be effectively enhanced if a user is assigned with multiple pseudonyms, and pseudonyms are carefully chosen when communicating with another party. The APPLAUS scheme [3] adopts such an approach. However, this incurs high operational overhead because of the management of identities and their corresponding pseudonyms. It also requires a deliberate pseudonym scheduling algorithm which statistically eliminates the possibility of linking multiple pseudonyms or user profiling based on a single pseudonym. In addition, the pseudonym manager (e.g., CA) has to be completely trusted. Otherwise, it could be the single point of failure. If an adversary breaks into the pseudonym manager and obtains a copy of the pseudonym mapping, the whole system would break down. Therefore, we do not design STAMP based on pseudonyms. Instead, we use cryptographic encryption and commitment techniques to hide users' identities in the STP proof generation process.

**Location granularity**: An STP proof system needs to be flexible in terms of location granularity, in order to enforce location privacy and accommodate localization error. The location of a prover could be represented by different levels of granularity, for example, a city, a neighborhood, or an exact geo-coordinate point. Though a prover needs to reveal both his/her identities and STP information in order to get services from a verifier, the prover does not necessarily trust the verifier completely. When a prover tries to claim his/her location at a particular time to a verifier, he/she should not be obligated to reveal his/her most accurate location to the verifier. Depending on the requested service, a prover should have control over the granularity of his/her location that is revealed to the verifier.

### C. Threat Model

**Prover**: A malicious prover seeks to create fake STP proofs without physically being present at a location. This includes creating fake STP proofs by himself/herself, lying to a witness about his/her location, tampering with the spatial-temporal information in his/her existing proofs, as well as stealing and using another user's STP proofs. Moreover, a malicious prover also attempts to obtain a witness's identity information in the entire process of STP proof generation.

**Witness**: A malicious witness's goals include acquiring a prover's identity information and repudiating an STP proof that is generated by him/her.

**Verifier**: A verifier is often a service provider or an authority that is trying to validate a prover's STP claim. A prover has to present both his/her identity and STP information to the verifier in order to get a service or simply prove his/her alibi. We assume that a verifier is trusted in terms of privacy leakage, that is, a verifier never leaks a prover's identity or STP information to any other parties. However, a prover should be able to only give a verifier his/her STP information that is necessary. In other words, a prover should have the control over which STP proofs and what location granularity of the STP proofs are revealed to a verifier.

**CA**: We assume CA is trusted but curious, in the sense that it is only trusted in term of correctly performing its functions, i.e., user registration, key and credential management, and trust assessment for STP proofs. Also, CA does not intentionally leak any information it stores to other individual users. However, CA may intend to use any information it learned to profile user's spatial-temporal history and thus a potential privacy abuse may happen at CA.

**Collusion**: We specifically tackle two different collusion scenarios in this work: (1) A witness can collude with a prover by creating an STP proof for him/her even though one or both of them are not at the location as claimed in the STP proof. We name this collusion scenario as *W-P collusion*. To the best of our knowledge, there is no good solution to detect this type of collusion yet. (2) A prover $A$ who requires a colluding prover $B$ who is at a specific location to masquerade as him/her and generate a fake STP proof. Though we assume $A$ does not give his/her private key to $B$, it is possible for $A$ and $B$ to have a hidden communication tunnel during the STP proof generation process, so that $B$ could relay messages to $A$, $A$ signs on them and returns them to $B$ in real time. This kind of collusion attack is a type of *Wormhole* attack [15], which has been more commonly referred to as the *Terrorist Fraud* attack [8] in location verification. It is one of the most challenging attacks to protect against in location verification. Applied to our context, we name this collusion scenario as *P-P collusion*.

Establishing an anonymous communication channel, attacks via communication links and DoS attacks (e.g., jamming and flooding) are out of the scope of this paper.

TABLE I: List of notations

| | |
|---|---|
| $M_1 \| M_2$ | Concatenation of messages $M_1$ and $M_2$ |
| $K_u^+$ | Public key of user $u$ |
| $K_u^-$ | Private key of user $u$ |
| $E^K(M)$ | Encryption of message $M$ with key $K$ |
| $H(M)$ | One-way hashing of message $M$ |
| $C(M,r)$ | Commitment to message $M$ with nonce $r$ |

## V. The STAMP Scheme

### A. Preliminaries

*1) Location Granularity Levels:* We assume there are $n$ granularity levels for each location, which can be denoted by $L_1, L_2, \ldots, L_n$, where $L_1$ represents the finest location granularity (e.g., an exact Geo coordinate), and $L_n$ represents the most coarse location granularity (e.g., a city). Hereafter, we refer to location granularity level as *location level* for short. When a location level $L_x$ is known, we assume it is easy to obtain a corresponding higher location level $L_y$ where $y > x$. The semantic representation of location levels are assumed to be standardized throughout the system.

*2) Cryptographic Building Blocks:* STAMP uses the concept of *commitments* to ensure the privacy of provers. A commitment scheme allows one to commit to a message while keeping it hidden to others, with the ability to reveal the committed value later. The original message cannot be changed after it is committed to. A commitment to a message $M$ can be denoted as $C(M, r)$ where $r$ is a nonce used to randomize the commitment so that the receiver cannot reconstruct $M$, and the commitment can later be verified when the sender reveals both $M$ and $r$. A number of commitment schemes [16], [17] have been proposed and commonly used. Our system does not require a specific commitment scheme. Any scheme which is perfect binding and computational hiding can be used. In our implementation, we used [16], which is based on one-way hashing.

One-way hash functions have the similar binding and hiding properties as commitment schemes. However, for privacy protection purpose, we do not use hash functions because they are vulnerable to *dictionary* attacks. An adversary who has a full list of possible inputs could run an exhaustive scanning over the list to crack the input of a hash function.

We assume every user has the ability to generate one-time symmetric keys. All parties have agreed upon a one-way hash function and a commitment scheme. The commitment scheme is implemented based on any pseudo-random generator. All cryptographic notations have been summarized in Table I.

*3) Distance Bounding:* A location proof system needs a prover to be securely localized by the party who provides proofs. A distance bounding protocol serves the purpose. A distance bounding protocol is used for a party to securely verify that another party is within a certain distance [18]. Different types of distance bounding protocols have been studied and proposed. A most popular category is based on *fast-bit-exchange*: one party sends a challenge bit and another party replies with a response bit and vice versa. By measuring the round-trip time between the challenge and the response, an upper bound on the distance between the two parties can be calculated. This fast-bit-exchange phase is usually repeated a number of times.

One of the most challenging problems in distance bounding is the Terrorist Fraud attack, i.e., the P-P collusion scenario. The Terrorist Fraud attack is hard to defend against because a fast-bit-exchange process demands no processing delay (or at least extremely small processing delay) at the prover end between receiving a challenge bit and replying a response bit [18]. Thus, signing cannot be executed in the middle of a fast-bit-exchange, which means a hidden communication tunnel between two colluding parties allows them to execute fast-bit-exchange and signing separately. Thereby, one is only certain that the party who executed the fast-bit-exchange is nearby, but the party may not actually possess the private key of the identity who he/she claimed to be.

To the best of our knowledge, three existing distance bounding protocols [9], [19], [20] addressed the Terrorist Fraud attack. The schemes proposed in [19], [20] are based on pre-established shared secrets, and thus does not fit our scheme considering the anonymity requirement between a prover and a witness. The Bussard-Bagga protocol proposed in [9] is based on a zero-knowledge proof technique, and it allows the prover to be authenticated via a private/public key pair. Hence, we adopt the Bussard-Bagga protocol as our distance bounding protocol. The protocol consists of three stages. The first stage is the *preparation* stage, where the prover encrypts his/her private key $K_p^-$ with a random symmetric key $k$ and gets an encrypted message $e$. The prover then commits to each bit of $e$ and $k$, resulting two sequences of bit commitments $C_e$ and $C_k$. In the second *distance bounding* stage, the prover sends $C_e$ and $C_k$ to the location verifier (or the witness in our context), the location verifier then starts a multi-round fast-bit-exchange. In round $i$, the prover replies the $i$th bit of $k$ or $e$ depending on the challenge bit. Since the location verifier never learns both bit values, he/she can never learn about $K_p^-$. After the fast-bit-exchange, the location verifier de-commits and verifies the corresponding bit commitments in $C_e$ and $C_k$ (only for the received bits) by asking the prover to provide the nonces used for those commitments. In the third *zero-knowledge proof* stage, the prover convinces the verifier that he/she knows $K_p^-$ through a zero-knowledge proof. It is not possible for a user to give away the values of $k$ and $e$, which would mean that $K_p^-$ is given away. Because of this, the protocol is not vulnerable to the Terrorist Fraud attack. In the scenario we are considering, a witness does not know the identity of a prover, we therefore cannot rely on the witness only to authenticate the prover via the zero-knowledge proof. We integrate the Bussard-Bagga protocol into STAMP by breaking up its execution and have the witness and verifier jointly authenticate the prover. The details are given in Section V-B.

## *B. Protocol*

*1) Overview:* Our protocol consists of two primary phases: *STP proof generation* and *STP claim and verification*. Figure 2 gives an overview of the two phases and the major communication steps involved.

When a prover collects STP proofs from his/her co-located mobile devices, we say an *STP proof collection event* is started by the prover. An STP proof generation phase is the process of the prover getting an STP proof from one witness. Therefore, an STP proof collection event may consist of multiple STP proof generations. The prover finally stores the STP proofs he/she collected in the mobile device.

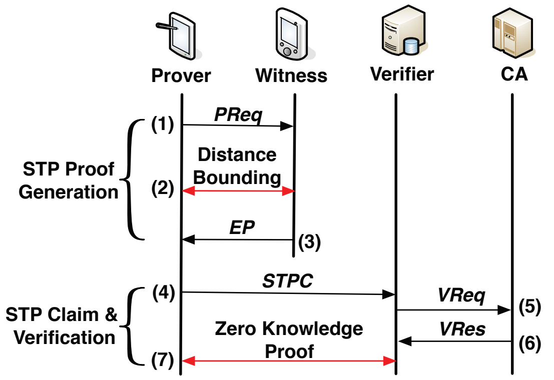


Fig. 2: An illustration of STAMP protocol

When a prover encounters a verifier (the frequency of such encounters is specific to the application scenarios) and he/she intends to make a claim about his/her past STP to the verifier, the STP claim and verification phase takes place between the prover and the verifier. A part of the verification job has to be done by CA. Therefore, communication between the verifier and CA happens in the middle of the STP claim and verification phase.

In Figure 2, the two arrowed lines in red color represent the latter two stages of the Bussard-Bagga protocol. These stages require multiple interactions between the two involved parties, and thereby are represented by doubly arrowed lines. The preparation stage of the Bussard-Bagga protocol does not need to be executed for every STP proof generation and thus is not shown. Users could run the preparation stage before each STP proof collection event or pre-compute and store several sets of the bit commitments and primitives, and randomly choose one set of them when needed. Subsequently, we present the details of the STAMP protocol.

*2) STP Proof Generation:* **Prover** Suppose a prover wants to start an STP proof collection event at time $t$, the prover first broadcasts an *STP proof request* (denoted as $PReq$) to other nearby mobile devices and waits for responses. A $PReq$ is constructed as follows:

$$PReq = C(ID_p, r_p)|L_1|t \tag{1}$$

where $ID_p$ is the prover's ID, $r_p$ is a random nonce generated by the prover for the commitment to $ID_p$, and $L_1$ is the lowest level of the current location.

**Witness**: A witness who receives a $PReq$ decides if he/she accepts the request. If the request is accepted, the witness sends an $ACK$ back to the prover, after which, the two parties start the execution of the distance bounding stage of the Bussard-Bagga protocol. This enables the witness to know that the party who is requesting an STP proof is within a certain range. However, the witness has no way to verify if the party has the private key which in fact corresponds to the committed identity. The zero-knowledge proof stage cannot be carried out by the witness because it requires the knowledge of the prover's public key. As shown in Figure 2, we leave the zero-knowledge proof stage to be executed by a verifier later in the STP claim and verification phase. However, this does not mean the witness can simply ignore the zero-knowledge proof

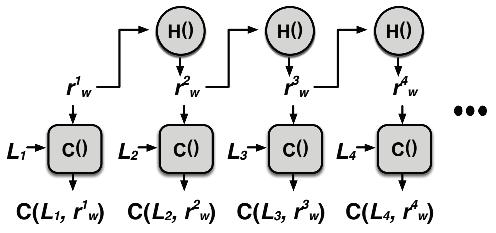


Fig. 3: Construction of location level commitments

stage. Based on the Bussard-Bagga protocol, the witness is able to compute a big integer $z$ from the bit commitments received from the prover after the distance bounding stage. The witness has to make sure that $z$ is securely enclosed in the STP proof so that the verifier can run the zero-knowledge proof stage base on $z$ and the prover's public key $K_p^+$.

If the distance bounding stage succeeds, the witness starts creating an STP proof for the prover. The witness first creates an *STP record* (denoted as $STPR$):

$$STPR = C(L_1, r_w^1)|\ldots|C(L_n, r_w^n)|t \qquad (2)$$

where $r_w^1$ is a random nonce generated by the witness and used to commit to $L_1$ provided in $PReq$. The higher location levels $L_2, \ldots, L_n$ are also committed with different nonces $r_w^2, \ldots, r_w^n$, which in turn are derived based on a *hash chain* of $r_w^1$:

$$r_w^x = H(r_w^{x-1}) \qquad \forall\ x = 2, 3, \ldots, n \qquad (3)$$

Figure 3 illustrates the construction of the location level commitments.

A plaintext STP proof (denoted as $P$) is then created as follows:

$$P = C(ID_p, r_p)|STPR|z \qquad (4)$$

$P$ is finally endorsed by the witness and encrypted using CA's public key. The *endorsed STP proof* (denoted as $EP$) is given by:

$$EP = E^{K_{CA}^+}(ID_w|P|E^{K_w^-}(H(P))) \qquad (5)$$

where $ID_w$ is the witness's ID. Finally, the witness sends $EP|r_w^1$ to the prover. $EP$ is encrypted using CA's public key to protect the witness's ID from being seen by the prover. $r_w^1$ should not be seen by CA and thus is not included in $P$ but sent along with $EP$.

**Prover**: Suppose the prover finally receives $EP|r_w^1$ from $m$ witnesses (denoted as $EP_1|r_{w,1}^1, \ldots, EP_m|r_{w,m}^1$) for this STP proof collection event, the prover stores them locally together with the associated primitives (i.e., $r_p$ and the Bussard-Bagga primitives) and the spatial-temporal information (i.e., $L_1$ and $t$). We say the prover now has created an *STP proof entry* for himself/herself at location $L_1$ and time $t$.

*3) STP Claim and Verification:* **Prover**: At the beginning of an STP claim and verification phase, the prover extracts the necessary data from his/her corresponding STP proof entry and creates an *STP claim* (denoted as $STPC$) as follows:

$$STPC = EP_1|\ldots|EP_m|r_{w,1}^x|\ldots|r_{w,m}^x|ID_p|r_p|L_x|t \qquad (6)$$

where $L_x$ is the lowest location level that the prover intends to reveal to the verifier. $r_{w,1}^x, \ldots, r_{w,m}^x$ are derived from the $r_{w,1}^1, \ldots, r_{w,m}^1$ based on a hash chain operation.

**Verifier**: After receiving the prover's $STPC$, the verifier needs CA's assistance in verifying the $STPC$. The verifier now constructs a *verification request* (denoted as $VReq$) by extracting the following information from the $STPC$:

$$VReq = EP_1|\ldots|EP_m|ID_p|r_p \qquad (7)$$

The $VReq$ is then sent to the CA.

**CA**: When CA receives a $VReq$, it is able to decrypt everything in $EP_1|\ldots|EP_m$ except for the committed location levels in the $STPR$s, because CA does not know any of the random numbers used by the witnesses to construct the location level commits. CA is now responsible for two tasks: *EP verification* and *P-W collusion detection*.

First, CA performs EP verification by checking the following in each $EP$ enclosed in the $VReq$:

- Signature of $K_w^-$ agrees with the public key of $ID_w$;
- $H(P)$ agrees with $P$;
- $C(ID_p, r_p)$ can be de-committed with $ID_p$ and $r_p$.

For all the $EP$s that passed the verification, CA starts a trust evaluation and obtains a P-W collusion detection result. We present the details of the P-W collusion detection process separately in Section V-B4.

If all the $EP$s fail the verification or the P-W collusion detection returns a positive result, CA sends back a *verification response* (denoted as $VRes$) with a one-bit failure notification to the verifier. Otherwise, CA creates a $VRes$ as follows and sends it back to the verifier:

$$VRes = E^{K_{CA}^-}(STPR_1|\ldots|STPR_m|z) \qquad (8)$$

where $STPR_1|\ldots|STPR_m$ are the $STPR$s extracted from $EP_1|\ldots|EP_m$ respectively, and $z$ is the big integer resulted from the distance bounding stage. Notice that $z$'s value is derived from the prover's bit commitments that are prepared for the Bussard-Bagga protocol. For different witnesses, we request the prover to use the same bit commitments during a same STP proof collection event. In this case, the same $z$ should be in each of $EP_1|\ldots|EP_m$. Therefore, only one copy of $z$ is attached in $VRes$.

**Verifier**: Upon receiving the $VRes$, the verifier performs two additional verification operations:

- **Zero-knowledge proof**: The zero-knowledge proof is done based on $z$ and the prover's public key $K_p^+$. A multi-round interaction is executed to minimize the prover's chance of cheating.
- **STPR opening**: From $VRes$, the verifier obtains $STPR_1|\ldots|STPR_m$ . The temporal information $t$ in each $STPR$ is first checked against $t$ claimed by the prover. A de-commitment is then done for $C(L_x, r_w^x)$ in each $STPR$, with $L_x$ and $r_{w,1}^x|\ldots|r_{w,m}^x$ obtained from the $STPC$. An inconsistent $t$ or a location commitment which cannot be de-committed nullifies the corresponding $STPR$ and thus the $EP$.

Now, suppose the verifier has a list of legitimate $EP$s passed the verifications, the verifier finally needs to determine if the prover's STP claim is successful by looking at number of legitimate $EP$s. Without loss of generality, we do not specify how the verifier makes such a decision. For instance, the verifier may consider the prover's STP claim successful as long as the number of legitimate $EP$s or the percentage of legitimate $EP$s among the originally received $EP$s exceeds a certain threshold.

*4) P-W Collusion Detection:* If a prover colludes with a witness, it is easy for the witness to give the prover a legitimate STP proof with fake spatial-temporal information. Since the STP proof generation process is done in an opportunistic manner and we do not assume a trusted party (e.g., a location authority or a trusted witness) in this process, a P-W collusion cannot be prevented or detected with a 100% certainty. As a countermeasure against P-W collusions, we proposed an entropy-based trust model which measures the likelihood of such an attack. The trust evaluation is done by CA, which requires CA to keep track of the STP proof transaction history between any two users. A user's STP proof transactions include both the STP proofs he/she gets as a prover and the STP proofs he/she creates as a witness.

First of all, we want to measure each user's collusion likelihood, based on his/her past STP proof transaction history. The intuition is that a legitimate user should not intentionally choose his/her witnesses, and therefore a user who gets majority of his/her STP proofs from a small set of users has a high likelihood to be colluding with these users. Based on this intuition, we define two factors which determine a user $u$'s collusion likelihood:

- **Diversity**: This is defined as the number of different users who had STP proof transactions with $u$. A higher diversity indicates that $u$ does not rely on a small group of witnesses, and thus suggests low collusion likelihood.
- **Fairness**: This is defined as the randomness in the distribution of STP proof transactions among all the different users who have had STP proof transactions with $u$. A highly random distribution indicates $u$ does not manipulate the witness choosing process, and thus suggests low collusion likelihood.

We use *entropy* to measure the collusion likelihood of a user because of its capability of capturing both the above two factors. In the STAMP system, provers meet witnesses on-the-spot. Thus, the system follows the Markov property which assumes that prover-witness pairing is memoryless. This is analogous to the rationale behind the definition of entropy in information theory. Given a user, if the diversity and fairness of his/her past STP proof transactions are high, the unpredictability of the prover-witness pairing will be high. Hence, we would like consider the user has a low collusion likelihood. Entropy is a measure of such unpredictability. Assuming $u$ has a total number of $N$ different users who had STP proof transactions with him/her, we denote this set of users as $u_1, u_2, \ldots, u_N$. Applying the definition of entropy into our context, $u$'s entropy is given by:

$$E_u = -\sum_{i=1}^{N} p(u, u_i) \log p(u, u_i) \tag{9}$$

where $p(u, u_i)$ denotes the percentage of past STP proof transactions between $u$ and $u_i$ out of $u$'s total past STP proof transactions.

The entropy measure gives an incentive to users to increase the diversity and fairness in the process of generating their STP proofs, regardless of the size of the user set. For scenarios where there is a massive number of users, each user is encouraged to reach out and interact with different users in order to maintain a high entropy. Certain applications may require users to interact with only a limited set of other users, such as the battlefield scout group example. In such applications, lacking diversity and fairness in a user's STP proof generation pattern is still undesirable and may be deemed as having a higher collusion tendency. This also mitigates the issue where a malicious user with multiple devices to generate fake proofs for himself/herself.

The trust of an $EP$ (denoted as $T$) is a scalar in [0, 1] which is evaluated by CA based on the prover's and witness's entropy as well as the specific STP proof transaction history between these two users. We define $T$ as follows:

$$T = 1 - e^{-\frac{E_P \cdot E_W \cdot \omega}{\mathbf{Q}(p,w)}} \tag{10}$$

where $\mathbf{Q}(P, W)$ denotes the number of STP proof transactions between the prover and the witness out of the total number of their distinct STP proof transactions in the past; $\omega$ is a scaling parameter which can be fine-tuned. The exponential term can be thought of as the penalty applied to the trust based on the collusion likelihood between the prover and the witness derived from their past STP transaction history with each other as well as with other users.

With the trust level of each $EP$ extracted from a $VReq$, CA needs to determine if these $EP$s resulted from a collusion. Such a decision should be made by consolidating the trust values calculated for all the $EP$s into one measure (denoted as $\hat{T}$):

$$\hat{T} = F(T_1, T_2, \ldots, T_m) \tag{11}$$

where $F()$ is the *trust consolidation function*. Different trust consolidation functions could be used. Some straightforward examples are $max$, $min$, and $average$. We suggest using *weighted average*, where the trust values are weighted by the number of past STP proof transactions of the corresponding $EP$'s witness. This makes the witnesses who have more past STP proof transactions (i.e., whose entropy is expected to be more accurate) have bigger influence on $\hat{T}$. Ultimately, a decision can be made by comparing $\hat{T}$ against a trust threshold (denoted as $\theta$).

## VI. Usage & Modifications

### A. Selfish node

Our proposed entropy-based trust model guards from P-W collusion by giving lower trust values to STP proofs generated by common or repeating witnesses. It also serves as an incentive mechanism for users to generate STP proofs for strangers. In a generic case, peer mobile users may be selfish. They may choose to save their battery power over generating STP proofs for other users, particularly when they are strangers. Let us consider a simple case when User $u_A$ wants to generate his STP proofs from stranger $u_B$. Let us say that the prior history of $u_B$ (participating in generation of STP proofs for himself/herself or others) is given by the set $\{(u_1, n_{u1}), (u_2, n_{u2})...(u_i, n_{ui})\}$ where $n_{uj}$ indicate the number of STP proof generation events with user $u_j$. Let $N = \sum_{i=1}^{I} n_{uj}$

$$E_{u_B} = -\sum_{i=1}^{I} \frac{n_{uj}}{N} \log \frac{n_{uj}}{N} = \log \prod_{i=1}^{I} \left(\frac{N}{n_{uj}}\right)^{\frac{n_{uj}}{N}}$$

$$\Rightarrow E_{u_B} = \log \left( \frac{N}{\prod_{i=1}^{I} n_{uj}^{\frac{n_{uj}}{N}}} \right) \tag{12}$$

On adding a request from $U_A$, the new Entropy for B becomes

$$E'_{u_B} = -\sum_{i=1}^{I} \frac{n_{uj}}{N+1} \log \frac{n_{uj}}{N+1} - \frac{1}{N+1} \log(\frac{1}{N+1})$$

$$E'_{u_B} = \log \left( \prod_{i=1}^{I} \left(\frac{N+1}{n_{uj}}\right)^{\frac{n_{uj}}{N+1}} \right) \times (N+1)^{\frac{1}{N+1}}$$

$$\Rightarrow E'_{u_B} = \log \left( \frac{N+1}{\prod_{i=1}^{I} n_{uj}^{\frac{n_{uj}}{N+1}}} \right) \tag{13}$$

It can be proved that $E'_{u_B} > E_{u_B}$ by showing that $N+1 > N$ and $n_{uj}^{\frac{n_{uj}}{N}} > n_{uj}^{\frac{n_{uj}}{N+1}}$. The entropy in Equation 10 includes both witness and prover. In the future, when $u_B$ is a prover, his trust would be higher if he generates STP proofs for $u_A$. This gives an incentive to a stranger witness $u_B$ to generate STP proofs for $u_A$.

## B. Coarse grain location

Trust computation becomes more reliable with increased number of users, hence choosing a coarser location level may be preferable for those services which seek higher reliability and trust but lower location granularity. We now show how STAMP can be used to collate STP proofs from witnesses from different locations to verify coarse grain location with higher trust.

Let us consider a scenario where a user moves from location (lowest level) $L^A$ to $L^B$. It is quite possible that levels of location are concurrent while higher levels of locations are same. $L^A_m = L^B_m, \forall p \leq m \leq n$. In this case, while claiming a location level $m \geq p$, the prover can use the STP proofs generated from the witnesses from both locations $A$ and $B$. The STP claim sent by the verifier will be slightly modified to account for the different times, when $EP$s at locations $L_A$ and $L_B$ are calculated.

$$STPC = EP^A_1|t^A|\dots|EP^A_p|t^A|EP^B_1|t^B|\dots|EP^B_q|t^B| \\ r^{(A)m}_{w,1}|\dots|r^{(A)m}_{w,p}|r^{(B)1}_{w,p}|\dots|r^{(B)m}_{w,q}|ID_p|r_p|L_m \tag{14}$$

The subsequent requests $Vreq$ and $VRes$ will be similarly modified to accommodate multiple times and bit commitment values $z$.

## C. Trusted Witnesses

STAMP is useful for a wide range of application where a centralized infrastructure (trusted wireless APs) is not available. The green commuting application we described in Section I is a good example scenario.

In some scenarios, a trusted mobile or stationary user may be available or required. For example, a store which wants to give discounts to its frequent customers may have some trusted mobile users such as customer service agents who are amongst the crowd in the store. In the prior case, we have incognito trusted mobile users. For users going to a park, it was observed that there are frequent events when users find no co-located user to generate STP proofs. Thus, the authorities set up a trusted wireless AP to generate STP proofs for travelers. The exact location of such trusted wireless AP is known. In these scenarios, since the witness is trusted, the prover can send all $EPs$ to CA or skip using CA since the proofs are already trusted (Figure 4). The first model fits well for incognito trusted mobile users while the other model serves well for wireless APs.

**Trusted mobile users:** In first case, the trusted witness is not readily recognized by the prover. The prover will send original STP claim to the CA. The CA will recognize trusted witness among the many and also improve trust score for other witnesses, as an added incentive (recognizing their honest work). To bring this into effect, entropy calculation is modified as follows:

$$E_u = -\left(1 + \left(\frac{N_t}{N}\right)^{\kappa}\right) \sum_{i=1}^{N} p(u, u_i) \log p(u, u_i) \tag{15}$$

where $N_t$ is the number of proofs generated by user $u$ with trusted witnesses, $N$ is the total number of proofs and $\kappa$ is a scaling parameter.

**Wireless AP:** In the second case, the endorsed STP proof $EP$ can be encrypted using verifier's public key, hence the verifier

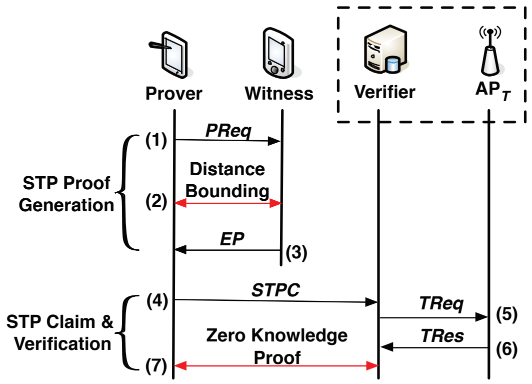


Fig. 4: STAMP protocol with trusted wireless AP

must be known in advance. The prover can skip other witness' proofs. Thus, it is restricted to specific applications only. Or, it is possible for trusted witness to sign the STP proofs using his private key, enabling any verifier to view the endorsed STP proof.

$$P = C(ID_p, r_p)|STPR|E^{K^+_{AP_T}}(z) \tag{16}$$

$$EP = E^{K^-_{AP_T}}(ID_w|P|E^{K^-_w}(H(P))) \tag{17}$$

EP is encrypted with private key of trusted AP, denoted as $AP_T$, so that verifier can view it using $AP_T$'s public key. However, this leads to the possibility of prover knowing the bit commitment $z$, so it is also encrypted using public key of $AP_T$. The prover directly sends STP claim to the verifier

$$STPC = EP_1|r^x_{AP_T}|ID_p|r_p|L_x|t \tag{18}$$

where $L_x$ and $r^x_{AP_T}$ correspond to the lowest location level desired to be revealed to verifier. The verifier communicates with $AP_T$ (instead of CA in generic model), to obtain big integer $z$ via exchange of messages $TReq$ and $TRes$.

$$TReq = EP_1|ID_p|r_p \tag{19}$$

$$TRes = E^{K^-_{CA}}(STPR|z) \tag{20}$$

Then, it follows the standard zero-knowledge proof protocol. Note that in many situations the AP is a part of the verifier, in which case the above message exchange will be done locally at the verifier.

Wireless APs can also be leveraged for coarse grain locations as well as improve trust for recent future (or past) location proofs. Let us say that the user obtained STP proofs from $AP_T$ at time $t_1$ at location A and has recently moved to location B at time $t_2$. It is possible that some levels of location levels are same for A and B.

$$\exists p, \text{ s.t. } L^A_m = L^B_m, \forall p \leq m \leq n$$

For location provenance at time $t_2$ also, the user can send the entire $EP$ by $T$ along with STP proofs generated at location B. A mobility model is used to decide whether the current time and location are feasible, based on location A. A simple mobility model will consider the physical distance $\phi$ and time $\tau$ of the two STP proof events. Let $\upsilon$ denote the maximum velocity in the considered locations. The condition $f = \frac{\phi}{\upsilon \times \tau} \leq 1$ can be used as a simple decision rule. More elaborate rules can be generated considering the exact location maps, sensor traces from devices and other information [21]–[24]. If 'no', then one-bit failure notification is sent to the verifier. If 'yes', trust is calculated based on accounts given by other witnesses.

Coarse grain location $L_m$ or higher levels, is considered trusted and the witnesses are also incentivized ($N_t$ is incremented by 1). For fine grain location verification, trust is calculated based on entropy-based trust function and the values are returned to the verifier along with an appendum of full coarse grain location verification.

## VII. Security Analysis

In this section, we analyze the security properties of the STAMP protocol and prove that the protocol can achieve our security goals.

**Proposition 1.** *A prover cannot create a legitimate $EP$ without a witness.*

Since users do not give away their private keys, a prover has no access to another user's private key. A plaintext STP proof ($P$) has to be signed by a legitimate witness to create a legitimate $EP$. If a prover uses his/her own private key or an illegitimate private key to create a signature for $EP$. CA will be able to detect it.

**Proposition 2.** *Without colluding with a witness, a prover cannot create a legitimate $EP$ without being present at the claimed location at the claimed time.*

Based on Proposition 1, a prover has to ask a witness to create a legitimate $EP$. Let us now consider two attacks: (1) a prover asserts a false location/time in a $PReq$; (2) a prover establishes a hidden communication tunnel with a proxy at the intended location and ask the proxy to send a $PReq$ for him/her (i.e., P-P collusion).

When a legitimate witness receives a $PReq$, he/she can easily check if $t$ in $Preq$ is within an acceptable range from the current time. Subsequently, the execution of the distance bounding stage enables the witness to determine if the party who sent the $PReq$ is within an acceptable distance. Since no signal travels faster than the speed of light, a prover who communicates with the witness from a distant location will be detected by the fast-bit-exchange in the distance bounding stage. Hence, Attack (1) can easily be detected by the witness.

Based on the Bussard-Bagga protocol, the zero-knowledge proof stage is able to guarantee that a party who ran the distance bounding stages with the witness in fact has the private key corresponds to the committed $ID_p$ in a $PReq$. That means, a prover has to give his/her private key to the proxy in order to pass both the distance bounding stage with the witness and the zero-knowledge proof stage with the verifier. Assuming a user never gives away his/her private key, our protocol ensures that Attack (2) cannot succeed.

**Proposition 3.** *A prover cannot change the spatial and/or temporal information in an $EP$.*

The location levels $L_1, \ldots, L_n$ are committed by the witness in an $STPR$. The $STPR$ is in turn encrypted by CA's public key in an $EP$. The prover does not have CA's private key, and thus cannot decrypt an $EP$ and see the location level commitments.

**Proposition 4.** *A prover cannot use an $EP$ created for another prover.*

By the binding property of commitments, a prover's ID is bound with the $C(ID_p, r_p)$, which is in turn encrypted in an $EP$. A prover therefore cannot change the $ID_p$ bound with an $EP$. If a prover claims to a verifier with his/her own $ID_p$ and another prover's $EP$, CA will detect that the $C(ID_p, r_p)$ in the $EP$ does not agree with the $ID_p$ in the $VReq$ sent by the verifier. If a prover claims to a verifier with another prover's $ID_p$ and $EP$, hoping to get services without showing his/her own identity, the verifier will detect that the prover does not possess the private key corresponding to $ID_p$ via the zero-knowledge proof stage.

**Proposition 5.** *A witness cannot repudiate a legitimate $EP$ created by him/her.*

A legitimate $EP$ contains $E^{K_w^-}(H(P))$. Based on the assumption that no user gives away his/her private key, $E^{K_w^-}(H(P))$ assures the non-repudiation property of an $EP$.

**Proposition 6.** *A prover and a witness cannot find out each other's identity.*

During an STP proof generation process, the prover's identity $ID_p$ is committed. Since $r_p$ is not known to the witness, he/she cannot de-commit $C(ID_p, r_p)$ and obtain $ID_p$. The witness's identity $ID_w$ is enclosed in $EP$, which is encrypted by CA's public key. Since the prover does not possess CA's private key, he/she cannot decrypt $EP$ and obtain $ID_w$. Furthermore, based on the Bussard-Bagga protocol, the distance bounding stage does not reveal the two parties' identities to each other.

**Proposition 7.** *$PReqs$ sent from the same prover for different STP proof collection events are unlinkable to a witness.*

A prover chooses different $r_p$s at different locations. Even a witness has received multiple $PReq$s from the same prover at different locations, there is no information that could help the witness to link these $PReq$s and thus obtain a location trace of the prover.

**Proposition 8.** *STP proofs generated from the same witness for different STP proof collection events are unlinkable to a prover.*

A witness chooses different $r_w^1$'s for different STP proof collection events. The $EP$ generated by a witness is always encrypted by CA's public key. Even the same prover has received multiple pieces of $EP|r_w^1$ from the same witness at different locations, there is no information that could help the prover to link these pieces of $EP|r_w^1$ and thus obtain a location trace of the witness.

**Proposition 9.** *The lowest location level a verifier learns about a prover is the level that the prover intends to reveal to him/her.*

In an $STPC$, the prover sends an $r_w^x$ to the verifier for each $EP$, where $x$ is the location level that the prover intends to reveal to the verifier. Due to the one-wayness of the hash-chain applied on $r_w^1$, the verifier cannot obtain any $r_w^y$ where $y < x$. Therefore, though the verifier has the access to $C(L_1, r_w^1), \ldots, C(L_n, r_w^n)$, the lowest location level commitment that can be opened by the verifier is $C(L_x, r_w^x)$. Due to the randomness introduced by the $r_w$'s, a dictionary attack over all possible locations under the location level $x$ is infeasible.

**Proposition 10.** *CA cannot learn any location information about a prover or witness from $VReq$.*

CA cannot de-commit any location level commitments from a $VReq$ and thus cannot obtain any location information about the prover and the witnesses associated with the $VReq$. Due to the randomness introduced by the $r_w$'s, a dictionary attack over all possible locations is infeasible.

**Proposition 11.** *Trusted users increase the overall trust of the system.*

Trusted mobile users are anonymous and hence, unlike the wireless AP case, a root of trust is needed in form of CA. The modified entropy evaluation system increases the entropy when a user or witness endorses for a user, also signed by a trusted user.

**Proposition 12.** *Nobody can fake himself/herself as a* *trusted user.*

In our scheme, each user has his/ her own private-public key pair, which uniquely identifies the individual. Since, the scheme involves encryption using these asymmetric keys. Since any existing user will also not sacrifice and give away his own private/ public keys, nobody else can fake as a trusted user.

## VIII. Experiments and Results

### A. *Prototype Implementation*

We implemented a prototype client application on Android with Java. Our experiments are carried out on two Samsung Exhibit II 4G devices equipped with Qualcomm MSM 8255 1GHz chipset, 512MB RAM, 1GB ROM, GPS, and Bluetooth, and running Android OS 2.3. Bluetooth is used as the communication interface between mobile devices. We use DSA key pairs for signing/authentication operations because DSA is based on the discrete-log problem, which makes it possess the mathematical properties desired by the Bussard-Bagga protocol. Since DSA is not designed for encryption/decryption purpose, we use RSA key pairs as subkeys for encryption/decryption operations. We use SHA1 as the one-way hashing function and 128-bit AES as the symmetric key encryption scheme. We implemented the string commitment scheme presented in [16] and use it for ID and location commitments. We model each location with six levels: exact location, neighborhood, town/city, region/county, state and country, where each level is represented by a name string except that the lowest level also has the geo-coordinates.

*1) Performance in Static Scenario:* With our implementation, we examine the computational time (also an indicator of power consumption) and storage that are needed to run STAMP. Since the STP verification is done by verifiers and CA where desktops or servers with high computational power can be used, we focus our testing on the STP proof generation phase that is executed on mobile phones. The results we show are obtained based on 10 runs of each test. No other background processes were running in parallel during the tests.

First, we study the impact of key size on the performance of our client application. Since both DSA and RSA are used in our implementation, we test three key size combinations representing three different security levels: (1) 512-bit DSA with 1024-bit RSA (denoted as 512/1024); (2) 768-bit DSA with 2048-bit RSA (denoted as 768/2048); (3) 1024-bit DSA with 3072-bit RSA (denoted as 1024/3072). Figure 5(a)-(c) show the computational resources required with these three key size settings.

Figure 5(a) shows the time needed for a prover to get an STP proof from a witness and for the portion of this process taken by the Bussard-Bagga distance bounding. We could observe that a majority portion of the STP proof generation is taken by the Bussard-Bagga distance bounding. It is easy to disable the distance bounding stage for application scenarios where P-P collusion is not a concern. In that case, the time for each proof generation will be significantly reduced to less than 0.2s even if large keys are used. Figure 5(b) shows the time needed for the Bussard-Bagga preparation stage. We can see this could cause long delay ($\sim$18s) if large keys are used. However, as we explained in Section V-B, to achieve best unlinkability, it could be executed only once for an STP proof collection event. Under a relaxed unlinkability requirement, users could also pre-compute and store several sets of the bit commitments and primitives, and randomly choose one set of them when needed. Figure 5(c) shows the size of an $EP$ that needs to be stored on a prover's mobile device. Since multiple $EP$s could be received for each STP proof collection event, the size of an $EP$ is the main factor that determines the storage need for an STP proof entry. We can see that each $EP$ is less than 2000 bytes. Though several such $EP$s may need to be retained for each STP proof entry, the storage consumption is definitely acceptable considering the storage capacity of today's mobile devices.

TABLE II: Default simulation settings

| Parameter | Value |
|---|---|
| Number of users | 1000 |
| Percentage of colluding attackers (PC) | 2% |
| Collusion tendency (CT) | 0.2 |
| Mean of witnesses ($\mu_w$) | 5 |
| Standard deviation of witnesses ($\sigma_w$) | 2 |
| Trust scaling parameter ($\omega$) | $1 \times 10^{-4}$ |
| Collusion trust threshold ($\theta$) | 0.6 |
| Training STP proof collection events | 10000 |

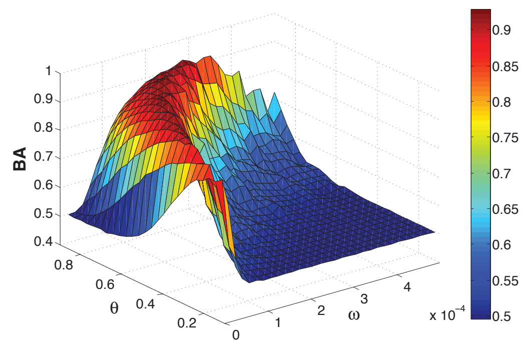


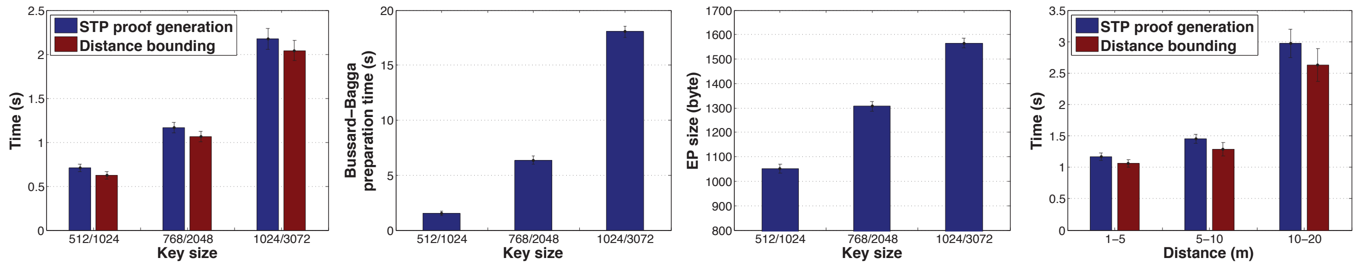
Fig. 6: BA under different $\omega$ and $\theta$

Figure 5(a)-(c) tell us the choice of key size is critical. Larger keys provide stronger security, but also require more resources in terms of computational time and storage. For achieving general security requirements for a light-weight mobile application, we suggest using a small key size setting.

In addition to key size, we study the impact of communication distance between mobile devices on the STP proof generation time. Figure 5(d) shows the results of our testing with the 768/2048 key size setting. As expected, the communication distance negatively affects the STP proof generation time as well as the distance bounding time.

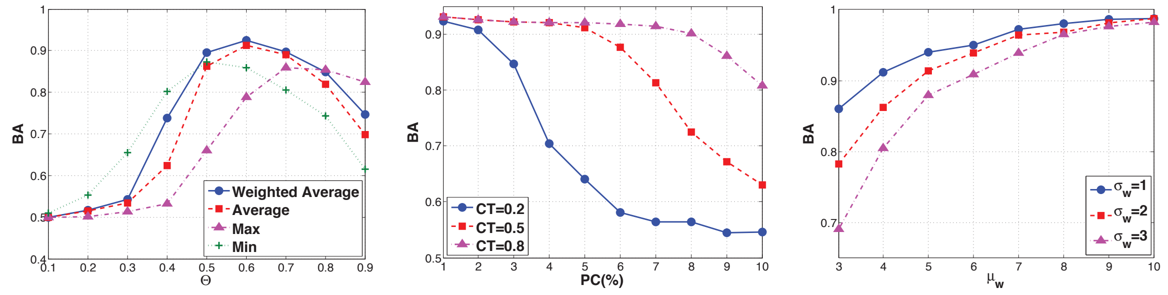
*2) Performance in Mobile Scenario:* We are also interested in an alternative scenario when prover or/and witnesses are mobile. It evaluates the feasibility when our scheme is applied to location-based services with continuous tracking. We perform experiments in three typical mobility mode, namely *Walking(W)*, *Biking(B)*, and *Driving(D)*, with two speed levels respectively. An outdoor path of 45 meters long is used for all three modes, while an additional path of 161 meters is dedicated for driving test in high mobility. In each mode, the witness moves towards the stationary prover and then move pass him and away, while the scheme automatically scanning for available witnesses, establishing connections, and generating location proofs. Since the protocol remains mostly the same as static scenarios, we focus more on qualitative metric of success rate instead of quantitative indicators which are similar to that of static scenario. Specifically, experiments are repeated four times for each setting and ratio of successfully completion of protocol for at least once is reported. We choose a balanced key size setting of 768/2048 bits in all experiments.

The result is shown in Figure 8. As expected, the perfor-



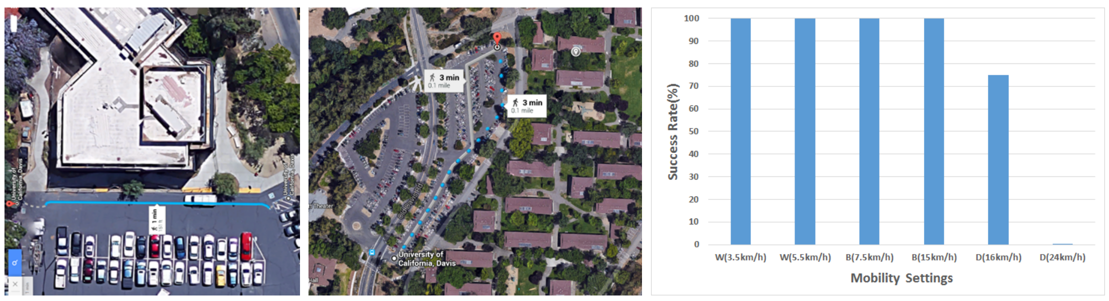
(a) Time to generate an STP proof under different key sizes (b) Time of Bussard-Bagga preparation stage under different key sizes (c) Size of EP under different key sizes (d) Time to generate an STP proof under different communication distances

Fig. 5: Implementation results



(a) BA under different trust consolidation functions (b) BA under different percentage of colluding attackers and collusion tendency (c) BA under different number of witnesses

Fig. 7: Impact of system settings and network conditions on BA



(a) Path 1 for Mobility Tests (45 meters) (b) Path 2 for Driving Tests in High Speed (161 meters) (c) Success Rate for Mobile Settings

Fig. 8: Evaluation in Mobile Scenario

mance is fairly stable in lower mobility involved in Walking and Biking settings. When walking at a low speed of 3.5 km/h, multiple location proofs can be successfully generated during one trip. We note that substantial time is consumed by Bluetooth inquiry and paging process for peer discovery, which takes approximately 12 seconds, in addition to the actual proof generation process. Nonetheless the results confirm Bluetooth connection provides adequate transmission range and is resilient enough in low and moderate mobility modes for our protocol to complete. On the other hand, when mobility level is increased the performance degraded drastically, as observed in driving tests. In our experiments all failures resulted from parties move out of transmission range before the inquiry process can complete. The intrinsic bottleneck of Bluetooth transmission range and overhead in discovery limits the performance of our scheme in high mobility scenarios. We will discuss this implication and possible solutions in Section IX.

## B. Simulation

To measure the effectiveness and accuracy of our P-W collusion detection, we implemented our trust model with Java simulation. In this section, we present our simulation details and the performance results that we obtained from our simulation experiments.

*1) Simulation Setup:* Since the main purpose of our simulation is to evaluate the effectiveness of our trust model in a hostile environment, we first test the case where there is no trusted mobile users. In our simulation, a total number of 1000 users are deployed. All users are traveling with a random mobility model. In each STP proof collection event, a random prover is selected among all the users. The selection of witnesses is modeled by a Gaussian distribution with default mean ($\mu_w$) of 5 and default standard deviation ($\sigma_w$) of 2. The *percentage of colluding attackers* (PC) among these users is varied from 1% to 10%. We allow all the attackers to find each other through a hidden channel and form a collusion group. Whenever an attacker needs to get a fake STP proof, he/she seeks assistance from random witnesses in the collusion group, in order to maximize his/her own entropy by making his SPT proof generation patten as unpredictable as possible within the collusion group. Each attacker is configured with a *collusion tendency* (CT) in the range of (0, 1], which represents the attacker's probability of launching a collusion for each of his/her STP proof collection event. In our experimental tests, we vary some critical parameters to see their impact on the performance. Default settings are used for parameters that are not under test. Table II summarizes our default settings.

We run a training phase with the first 10000 STP proof collection events, i.e., an average of 10 STP proof collection events for each user. Each of the data points shown in our simulation results is based on another 100000 STP proof collection events after the training phase, i.e., an average of 100 STP proof collection events for each user.

*2) Performance Metric:* We use the *Balanced Accuracy* (BA) [25] as our performance metric, which is a commonly used accuracy measure for classification algorithms. BA is defined as the arithmetic mean of sensitivity (true positive rate) and specificity (true negative rate). We choose BA because: (1) its interpretation is straightforward, (2) it takes both sensitivity and specificity into account, and (3) it avoids inflated performance estimates on imbalanced datasets.

*3) Simulation Results:* First, two crucial parameters that affect how our trust model performs are the trust scaling parameter $\omega$ and collusion trust threshold $\theta$. We run extensive tests to see the BA distribution under different choices of $\omega$ and $\theta$ with our default setup. The results are shown in Figure 6. Our trust model performs well (BA $> 0.9$) only when good $\omega$ and $\theta$ are chosen. From Figure 6, we observe that our trust model is very sensitive to the choice of $\omega$, which agrees with Equation 10. For different choices of $\omega$, different $\theta$ has to be set to achieve a high BA. We choose $\omega = 1\times10^{-4}$ and $\theta = 0.6$ because this pair yields the best result in our testing.

Subsequently, we examine how the choice of trust consolidation functions affects the accuracy of our collusion detection. Figure 7(a) shows the BA results when we used *weighted average*, *average*, *max* and *min*. The tests are carried out under the default settings except that we vary the collusion trust threshold $\theta$, because different trust consolidation functions get their best BA with different $\theta$. *max* gets its best BA with a higher $\theta$ than other approaches because *max* only compares the highest trust value with $\theta$. Similarly, *min* is gets its best BA with a lower $\theta$. Comparing the best BAs yielded by the four approaches, we can see *weighted average* outperforms the other three. In our simulation, the STP proof collection events are randomly distributed to the users, so users' amounts of past STP proof transactions are roughly balanced. We anticipate the BA of *weighted average* will be more pronounced if users' amounts of past STP proof transactions are more skewed.

Figure 7(b) shows the performance of our trust model with different percentages of colluding attackers (PC) and when the attackers have different collusion tendencies (CT). We can see that our trust model is more resistant against attackers with a higher collusion tendency. Under high collusion tendency (CT=0.8), our trust model achieves a BA over 0.9 for up to 8% of colluding attackers. This means if attackers launches collusions frequently so that 80% of their STP proofs are fake, our trust model is able to detect the collusions with BA over 0.9 even when 8% of all users are colluding. If the percentage of colluding attackers is low (PC $\leq$ 2%), Figure 7(b) shows that our trust model can achieve a BA over 0.9 even when CT is

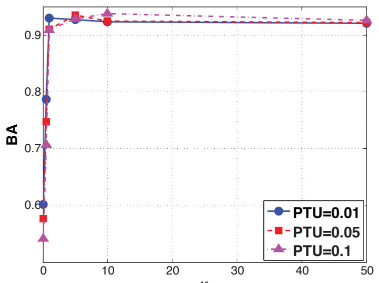


Fig. 9: BA under different $\kappa$ and percentage of trusted users

as low as 0.2. This means if 2% of all users are colluding, and every attacker is intelligent and tries to cover their collusions by having 80% of their STP proof transactions legitimate, our trust model can still detect collusions with a BA over 0.9.

Subsequently, we want to study the impact of the number of witnesses involved in an STP proof collection event. Since we model the number of witnesses for each STP proof collection event with Gaussian Distribution, we vary the mean ($\mu_w$) and standard deviation ($\sigma_w$) and test the resulting BA when other settings are set to default. Figure 7(c) shows the BA levels we get for different $\mu_w$ and $\sigma_w$. We can conclude that our trust model performs better when there are more witnesses involved in STP proof collection events on average. Thereby, a good way to make P-W collusions hard and also enhance the collusion detect accuracy is to require more witnesses for an STP proof.

Apart from all the above performance studies focusing on different factors, we would also like to look into the case where trusted wireless mobile users present and study the impact of them with our entropy Equation 15. There are two factors we would like to investigate for this case, the percentage of trusted users (PTU) and the scaling parameter $\kappa$. Figure 9 shows the BA levels we get for different PTU and $\kappa$. If we consider $\kappa = \infty$ makes Equation 15 the same as Equation 9, we can see a proper choice of $\kappa$ when trusted users present can actually increase our BA as the peak BA value exists on all three curves. Generally a value in the range [1, 10] is a good choice. A value smaller than 1 makes the weight of $\left(\frac{N_t}{N}\right)^\kappa$ too large, which would cause a big drop of BA as the trust model is very sensitive to the entropy values. The best value of $kappa$ is affected by PTU. The higher the PTU, the larger the best $\kappa$ value tends to be. This is because higher PTU makes $\frac{N_t}{N}$ larger and thus larger $\kappa$ has the effect of limiting the weight of $\left(\frac{N_t}{N}\right)^\kappa$. It should also be noted that the peak BA values becomes larger when PTU increases, this confirms that more trusted users in the system would make our trust evaluation more accurate given that a good $\kappa$ value is chosen.

## IX. Discussion and Future Work

From our experimental results, we observe that under small key size settings, our scheme works efficiently in terms of both computational and storage resources. However, the computational latency could become rather long when large keys are desired. A major part of computational cost is caused by the Bussard-Bagga protocol, which is known for its expensive computation due to large amount of modular exponentiations [18]. Other than defending against the Terrorist Fraud attack (P-P collusion), functionalities of STAMP do not specifically rely on the Bussard-Bagga protocol. Therefore, under circumstances where P-P collusion is not a concern, we suggest to disable the Bussard-Bagga stages in STAMP, which will result less than 0.2s for each STP proof transaction (distance bounding time deducted from STP proof generation time) without the necessity of the preparation stage. Furthermore, active on-going research in the location verification field is being conducted to achieve the same security property as the Bussard-Bagga protocol with much better performance. A new distance bounding scheme can be easily plugged into STAMP and replace the Bussard-Bagga protocol. It is also a part of our future work to investigate such possibilities.

Bluetooth is a ubiquitous short-range, low-power communication technology that also provides a robust device discovery mechanism, making it a logical choice for implementing our prototype. As observed in evaluation, limited range and discovery latency due to underlying Bluetooth technology exerts another negative impact on performance of our protocol, especially in high mobility scenarios. Such drawbacks are not unique for our scheme and several methods [26], [27] have been proposed to achieve a trade-off between discovery and latency which we can adapt in our future work. Furthermore it is necessary to emphasize that our protocol for proof generation is designed to be agnostic of communication technologies and should be interoperable with other types of ad hoc connections such as Wi-Fi mesh and vehicle networks. Appropriate method can be selected adaptively according to different situations with respect to mobility, witness density, etc. We intend to implement a framework in our future prototype to facilitate the switch among multiple compatible communication methods.

Our P-W collusion detection is supported by entropy-based trust evaluation, instead of complex graph algorithms like the ones used by the APPLAUS system. Therefore, each run of our P-W collusion detection only requires a number of cheap computations. It is much more efficient than APPLAUS where a few hundred seconds are needed to run a detection among a few thousands of users. The weakness of our detection, however, is that if attackers only launch collusions very infrequently, or if there is a large pool of users that an attacker can choose to collude with, the accuracy may drop significantly. Nevertheless, unless trusted infrastructures are deployed at every location, it is always hard to tell if an STP proof is a result of collusion or not. Our trust model serves as a good countermeasure so that malicious users are deterred from launching collusions of their own free will or with only a small group of users. In many cases, people are around with their family members and friends more often, this will inevitably affect people's entropy. However, we consider this as a general case for most of the users and thus it is possible to adjust the parameter in Equation 10 to maintain an unshifted trust range. We leave the investigation of how exactly such a social pattern affects our trust evaluation as future work.

STAMP is primarily designed for ad-hoc mobile users generating location proofs for each other in a distributed setting. In adhoc scenarios, such as a public meeting place or a park, it is likely that the user will encounter different people on different days. When a user encounters the same set of people at some specific location every day, the trust values will be reduced. In scenarios such as a classroom or an office, we envision the use of trusted witnesses or trusted wireless APs to resolve the situation.

## X. Conclusion

In this paper we have presented STAMP, which aims at providing security and privacy assurance to mobile users' proofs for their past location visits. STAMP relies on mobile devices in vicinity to mutually generate location proofs or uses wireless APs to generate location proofs. Integrity and non-transferability of location proofs and location privacy of users are the main design goals of STAMP.

We have specifically dealt with two collusion scenarios: P-P collusion and P-W collusion. To protect against P-P collusions, we integrated the Bussard-Bagga distance bounding protocol into the design of STAMP. To detect P-W collusion, we proposed an entropy-based trust model to evaluate the trust level of claims of the past location visits. Our security analysis shows that STAMP achieves the security and privacy objectives. Our implementation on Android smartphones indicates that low computational and storage resources are required to execute STAMP. Extensive simulation results show that our trust model is able to attain a high balanced accuracy ($> 0.9$) with appropriate choices of system parameters.

## Acknowledgment

This work was sponsored by the Army Research Laboratory and was accomplished under Cooperative Agreement Number W911NF-09-2-0053.

## References

[1] S. Saroiu and A. Wolman, "Enabling new mobile applications with location proofs.," in *ACM HotMobile*, 2009.

[2] W. Luo and U. Hengartner, "VeriPlace: a privacy-aware location proof architecture.," in *ACM GIS*, 2010.

[3] Z. Zhu and G. Cao, "Towards privacy-preserving and colluding-resistance in location proof updating system," *IEEE Transactions on Mobile Computing*, 2011.

[4] N. Sastry, U. Shankar, and D. Wagner, "Secure verification of location claims," in *ACM WiSe*, 2003.

[5] R. Hasan and R. Burns, "Where have you been? Secure location provenance for mobile devices," *CoRR*, 2011.

[6] B. Davis, H. Chen, and M. Franklin, "Privacy preserving alibi systems," in *ACM ASIACCS*, 2012.

[7] I. Krontiris, F. Freiling, and T. Dimitriou, "Location privacy in urban sensing networks: research challenges and directions," *IEEE Wireless Communications*, 2010.

[8] Y. Desmedt, "Major security problems with the 'unforgeable' (feige)-fiat-shamir proofs of identity and how to overcome them," in *SecuriCom*, vol. 88, pp. 15–17, 1988.

[9] L. Bussard and W. Bagga, "Distance-bounding proof of knowledge to avoid real-time attacks," *Security and Privacy in the Age of Ubiquitous Computing*, 2005.

[10] B. Waters and E. Felten, "Secure, private proofs of location," tech. rep., Department of Computer Science, Princeton University, 2003.

[11] M. Li, S. Salinas, and P. Li, "Locaward: A security and privacy aware location-based rewarding system," *IEEE Transactions on Parallel and Distributed Systems*, vol. 25, no. 2, pp. 343–352, 2014.

[12] T. V. A. Pham, K. Huguenin, I. Bilogrevic, and J.-P. Hubaux, "Secure and private proofs for location-based activity summaries in urban areas," in *ACM Ubicomp*, no. EPFL-CONF-198673, 2014.

[13] 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 *IEEE ICNP*, 2013.

[14] A. Pfitzmann and M. Köhntopp, "Anonymity, unobservability, and pseudonymity-a proposal for terminology," in *Designing privacy enhancing technologies*, Springer, 2001.

[15] Y.-C. Hu, A. Perrig, and D. B. Johnson, "Wormhole attacks in wireless networks," *IEEE Journal on Selected Areas in Communications*, vol. 24, no. 2, pp. 370–380, 2006.

[16] S. Halevi and S. Micali, "Practical and provably-secure commitment schemes from collision-free hashing," in *CRYPTO*, 1996.

[17] I. Haitner and O. Reingold, "Statistically-hiding commitment from any one-way function," in *ACM Symposium on Theory of Computing*, 2007.

[18] D. Singelee and B. Preneel, "Location verification using secure distance bounding protocols," in *IEEE MASS*, 2005.

[19] J. Reid, J. Nieto, T. Tang, and B. Senadji, “Detecting relay attacks with timing-based protocols,” in *ACM ASIACCS*, 2007.

[20] C. Kim, G. Avoine, F. Koeune, F. Standaert, and O. Pereira, “The swiss-knife rfid distance bounding protocol,” in *ICISC*, pp. 98–115, 2009.

[21] H. Han, J. Yu, H. Zhu, Y. Chen, J. Yang, Y. Zhu, G. Xue, and M. Li, “Senspeed: Sensing driving conditions to estimate vehicle speed in urban environments,” in *IEEE INFOCOM*, pp. 727–735, April 2014.

[22] I. Afyouni, C. Ray, and C. Claramunt, “Spatial models for context-aware indoor navigation systems: A survey,” *Journal of Spatial Information Science*, no. 4, pp. 85–123, 2014.

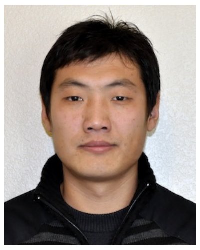
[23] N. Roy, H. Wang, and R. R. Choudhury, “I am a smartphone and i can tell my user’s walking direction,” in *ACM MobiSys*, 2014.

[24] R. Steinbach, J. Green, and P. Edwards, “Look who’s walking: Social and environmental correlates of children’s walking in london,” *Health & place*, vol. 18, no. 4, pp. 917–927, 2012.

[25] K. Brodersen, C. Ong, K. Stephan, and J. Buhmann, “The balanced accuracy and its posterior distribution,” in *IEEE ICPR*, 2010.

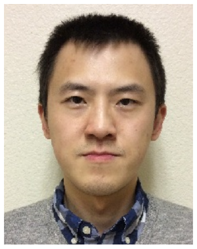
[26] B. Peterson, R. Baldwin, and J. Kharoufeh, “Bluetooth inquiry time characterization and selection,” *Mobile Computing, IEEE Transactions on*, vol. 5, pp. 1173–1187, Sept 2006.

[27] J. Zhu, K. Zeng, K.-H. Kim, and P. Mohapatra, “Improving crowd-sourced wi-fi localization systems using bluetooth beacons,” in *Sensor, Mesh and Ad Hoc Communications and Networks (SECON), 2012 9th Annual IEEE Communications Society Conference on*, pp. 290–298, June 2012.



**Xinlei Wang** received his Ph.D. degree in Computer Science from University of California, Davis in 2014, and his B.E. degree in Electrical and Electronic Engineering from Nanyang Technological University, Singapore in 2008. He is currently a Research Scientist at Facebook. He has worked in the Science Outreach for Army Research (SOAR) program at U.S. Army Research Lab in 2012 and in the Energy and Sustainability Lab of Intel Labs in 2013. His research interests include information security and privacy in mobile and social networks, trust and reputation management in distributed systems, and data provenance and its impact on Quality of Information (QoI).

**Amit Pande** received the bachelors degree in Electronics and Communications Engineering from IIT Roorkee, India, in 2007, and the Ph.D. degree in Computer Engineering from Iowa State University, USA, in 2010. He has been a Research Scientist with the University of California at Davis since 2010. He has authored over 65 peer-reviewed conference and journal papers. His current research interests are in application of data analytics to mobile, health, networking, and other applications and in network security. He received many university research excellence awards and best paper awards at international conferences.



**Jindan Zhu** received his BS and MS Degrees in Information Security from Shanghai JiaoTong University in 2006 and 2009 respectively. He is currently pursuing the PhD degree in University of California, Davis. His research interests include indoor localization, mobile sensing, and context aware system and services.

**Prasant Mohapatra** is a Professor in the Department of Computer Science and is currently serving as the Associate Chancellor of the University of California, Davis. He was the Department Chair of Computer Science during 2007-13 and served as the Interim Vice-Provost and the Campus CIO of UC Davis during 2013-14. He received his doctoral degree from Penn State University in 1993. He was/is on the editorial board of the IEEE Transactions on Computers, IEEE Transactions on Mobile Computing, IEEE Transaction on Parallel and Distributed Systems, ACM WINET, and Ad Hoc Networks. He has been a Guest Editor for IEEE Network, IEEE Transactions on Mobile Computing, IEEE Communications, IEEE Wireless Communications, and the IEEE Computer. He is a Fellow of the IEEE and a Fellow of AAAS.