Privacy-preserving Proof-of-Location With Security Against Geo-tampering

Journal:	*Transactions on Dependable and Secure Computing*
Manuscript ID	TDSC-2020-10-0607.R1
Manuscript Type:	Regular Paper
Keywords:	Proof-of-Location, Geo-tampering, Distance bounding

Privacy-preserving Proof-of-Location With Security Against Geo-tampering

Mamunur Akand, Reihaneh Safavi-Naini, Marc Kneppers, Matthieu Giraud, and Pascal Lafourcade

Abstract—A Proof-of-Location (POL) system is used to issue a proof-of-location token (pol) to a user who has been present at a location ℓoc, such that it can be later presented to a verifier to assure the presence of the user at ℓoc. Basic POL security requirements are *unforgeability* of pol, and its *non-transferability* (a pol issued to user u_1 cannot be used by u_2). An additional important property of POL systems is *user privacy* against the issuers and verifiers. We make two contributions. First, we formalize the POL security and privacy properties, and construct the first system providing provable security and privacy against the issuer and the verifier, both. Second, we introduce a *geo-tampering attack* that completely breaks POL system security, by simply changing the location of a pol issuing node. The attack applies to portable infrastructure nodes that are not continually monitored. We propose an algorithm that is used by a pol issuer to provide a location integrity "proof", that will be embedded in a pol to protect against this attack. The proof relies on a novel application of Euclidean Distance Matrices. We implemented our POL on an off-the-shelf Android smartphone to show the practicality of the proposed algorithms.

Index Terms—Proof-of-Location, Distance bounding, Geo-tampering.

1 Introduction

A Proof-of-Location (POL) system issues *proof-of-location tokens* that can be carried by the user and later presented to, and verified by, the verifiers. POL systems [1], [2], [3] rely on a *trusted location infrastructure* that reliably determines the location of the claimant, using a set of *location infrastructure nodes* that cover the area of interest, and issue a proof-of-location token, pol, to a user who has "proved" their presence at a claimed location ℓoc. The issued token can be later used to prove to a (trusted) *verifier* that the user has been at ℓoc. A pol can be seen as a credential that can be used together with other credentials of the user to provide refined access control [4], [5] and supply chain management [6].

A pol can be with respect to a specific *geo-coordinate* that is obtained from a Global Positioning System (GPS), or act as a certification for the *proximity* of the prover to the *issuer*, which has a known (to the infrastructure) location. POL systems usually use the latter approach because of the unavailability and unreliability of GPS signals indoors, as well as a range of known attacks on the GPS systems [7], [8][1].

The only known method of verifying closeness to the issuer with provable cryptographic security, is by using Distance Bounding (DB) protocols [9], [10], [11] that use well designed challenge-and-responses to allow the (untrusted) prover to prove their proximity (being within a distance bound B) to a (trusted) verifier. Systems that use Radio Signal Strength [12] or Time of Flight [13], [14] for estimating proximity are not reliable and allow the distance to be shortened [15].

State of DB-based POL systems. We focus on POL systems that use DB protocols [16], [17], [18]. Such systems must provably provide a number of security properties including unforgeability, non-transferability and privacy. Existing POL systems however have two major limitations. Firstly, there is no known protocol that provides privacy in the sense that the pol issuer and pol verifier cannot learn the identity of the user, or be able to link multiple pols and trace the user. To generate a pol, the issuer (i) uses a DB protocol to verify a user u's position with respect to issuer's location that is assumed known and trusted, and (ii) digitally signs this location information. To provide non-transferability, the prover's identity must be included in the pol. This however results in full traceability of a user, both while interacting with the pol issuer and also while presenting the pol to a verifier. The only POL system that considers user privacy against the issuer and the verifier is due to Gambs et al. [17]. This POL uses a public key DB protocol [19] that does not provide adequate security level for POL applications. More specifically, secure DB protocols must protect against three main attacks, distance fraud, Mafia fraud and Terrorist fraud [10]. Although in some applications some of these properties can be tolerated, for secure POL systems, all these properties are required. The protocol in [19] was shown to be insecure against distance fraud and terrorist fraud attack [20], which renders the POL system in [17], insecure. It is worth noting that the construction of this POL system is not modular, and critically depends on the structure of its underlying DB protocol. Thus, *there is no known construction of private POL system.* A second important limitation of the works in this area is that security of POL systems is only argued informally. POL systems can be seen as anonymous credential systems that require formal cryptographic model

- *M. Akand and R. Safavi-Naini are with University of Calgary, Canada. E-mail: {mdmamunurrashid.akan, rei} @ucalgary.ca*
- *M. Kneppers is with Telus Communications, Canada. E-mail: marc.kneppers @telus.com*
- *M. Giraud and P. Lafourcade are with University Clermont Auvergne, France. E-mail: {matthieu.giraud, pascal.lafourcade} @uca.fr*

1. One can always consider a combination of the two to improve accuracy if reliable GPS signal is available

and analysis similar to other credential systems such as [21], [22], [23]. A formal model and analysis will form the foundation of essential properties such as ensuring that *pol* issuing infrastructure has not been tampered with, that we consider in this paper, as well as new properties such as combining multiple credentials to achieve high level properties.

Our contribution.

Our goal in this paper is to lay a solid foundation for design and analysis of POL systems, and design a secure POL system with provable security using a trusted location infrastructure. We then relax the trust assumption on the *the location of infrastructure nodes* and consider the case that some of the infrastructure nodes are *displaced by the attacker*. Such a physical displacement is a real threat for small and unprotected infrastructure nodes, and as will be shown in Section 4 can completely compromise security of the system. We show how such an attack can be efficiently detected, and extend POL systems to provide security in this extended model. In the following we outline our contributions that are to, *(i)* formalize security and privacy of POL systems, and construct a POL system that provably achieves these properties, assuming an infrastructure that has assured location for the infrastructure nodes; *(ii)* define geo-tampering attack and show its devastating effect on the security of POL systems and propose an efficient and effective approach to providing "proof" of infrastructure integrity, and show that the *pol* can be extended to include this extra infrastructure integrity information, while maintaining its provable security guarantee; and *(iii)* implement our cryptographic algorithm and infrastructure integrity generation algorithms, to show feasibility of our solution in practice. More details below.

(i) Security model and construction. We use a game-based approach to define two security properties, *unforgeability* and *non-transferability*, and an indistinguishability based approach to define *user full anonymity* in its interaction with the issuer and the verifier. We assume there is an identity issuer that stores sufficient amount of secret information that can be used to "open" transcripts of the user's interactions with the issuer and the verifier, if needed, and hence providing the required accountability.

The POL construction requires a DB protocol that (a) provides anonymity for the (DB) prover against the (DB) verifier, and (b) includes sufficient information in the DB protocol transcript that can be used in the *pol* to make it non-transferable. We construct such a DB, use it to construct a POL, and prove its security and privacy in our proposed model. To our knowledge none of the existing anonymous DB protocols [24], [25], [26], [27] satisfy both properties simultaneously.

(ii) Geo-tampering attacks and protection. In geo-tampering attack the hardware and the software of the access point will remain untouched, and all cryptographic protocols are run flawlessly[2]. The attacker however physically moves one or more access points. Figure 1 is an illustration of the attack:

2. Modern WiFi access points come with built-in encryption scheme such as WPA/WPA2 and are considered secure if a strong enough password or paraphrase is used. Therefore, attacker's ability to move the AP does not necessarily mean that they can break into the AP.

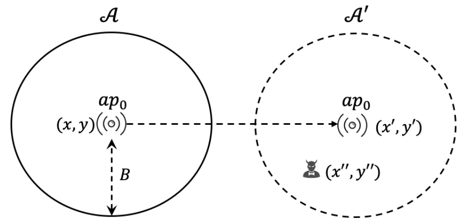


Fig. 1. A *geo-tampering* attacker moves ap_0 from (x, y) to (x', y'). This enables the attacker located at (x'', y'') in $\mathcal{A}'$ to claim locations in $\mathcal{A}$.

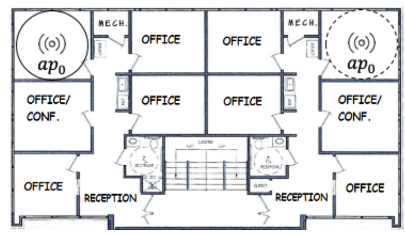


Fig. 2. ap_0 in the office on the top left is moved to the office on the top right. Alice can obtain a proof of being in the former office, while being in the latter.

the attacker moves ap_0 to a geo-coordinate (x', y'). This results in the set of "close-by" points $\mathcal{A}$ to be replaced by $\mathcal{A}'$, allowing the attacker (located at (x'', y'') to claim a location in $\mathcal{A}$. The attack is feasible because of the prevalence of small access points that are increasingly used as infrastructure nodes. The attack can stay undetected until, for example, when a *proof-of-location (pol)* issued to an honest user is not verified as expected. Fig. 2 shows an example of this attack in a typical setting, where an employee can obtain a proof-of-location without being in their designated office.

To detect the attack, an infrastructure node must perform a real-time integrity checking algorithm to provide verifiable information about its location. We sometimes informally refer to this information as "proof" of infrastructure integrity. In Section 4 we show that generating such information naively, for example by using trilateration, requires introduction of many additional infrastructure nodes. We then propose a novel method of achieving location integrity information by using geo-location of "neighboring" nodes, that are reachable from *pol* issuing node through (possibly) multiple hops. We construct an initial Euclidean Distance Matrix (EDM) (see Section 4.1) that records the pairwise physical (point-to-point straight line) distances among the nodes in the infrastructure. This matrix is then used to verify the "proof of integrity" of a *pol* issuing node, by comparing the real-time measured pairwise distances of neighbor nodes of the *pol* issuer, with the corresponding recorded values. The effectiveness of the approach is due to the fact that *all* physical distances between nodes in the neighbourhood are used, while trilateration based approaches rely on the distances of the neighbouring nodes to the issuing node, only.

(iii) Implementation and experiments.

a. Implementation. We give a proof-of-concept implementation of our proposed POL system using the Idemix Java library[3] for an off-the-shelf android smart-phone. The library is developed by IBM Security Research and is widely used for anonymous credentials. We use the library to implement the cryptographic components of the protocol including commitment, zero-knowledge proof and CL-signatures (see Section 2 for a description on these primitives), in three different security (RSA modulus length for CL-signature) settings.

b. Geo-tampering detection: We implement our detection algorithm to verify its correct detection of tampering for spars neighborhoods. We show that our approach can detect geo-tampering attack with reasonable "accuracy", that is defined as the minimum amount of movement of an access point before tampering is detected (see section 5). Our experiments clearly show effectiveness and superiority of using distance information of neighboring nodes, compared to trilateration which only uses the distances of the neighboring node to the issuing node.

Organization. Section 2 gives the system setting and definitions of proof-of-location schemes and their security properties. Our proof-of-location construction is in Section 3. Section 4 introduces geo-tampering attack on proof-of-location schemes, and presents an extended proof-of-location scheme proven to be secure against this attack. Section 5 details our experiments. We discuss related work in Section 6 and conclude the paper in Section 7. The supplemental material includes the security proofs.

2 Model and Definitions

2.1 Cryptographic Primitives

The following cryptographic primitives are used in our proof-of-location scheme.

Commitment. Commitment is a two-party protocol between a committer and a receiver. A commitment scheme ($\mathbb{C}$) has two stages - *Commitment* stage and *Reveal* stage. In the *Commitment* stage, the committer, for a value x produces a *commitment c*, and in the *Reveal* stage, opens the commitment to a value x'. A commitment protocol is *perfectly hiding* if the receiver cannot learn anything about the committed value x after the *Commitment* stage, and *perfectly binding* if in the *Reveal* stage, the committer can open the commitment only to the committed value $x(x' = x)$. The formal definitions of these properties can be found in [28]. A property can be satisfied against an unbounded adversary, resulting in statistical security, or a polynomially bounded adversary, resulting in computational security.

We use a commitment scheme proposed by Damagard and Fujisaki [28]. For a security parameter λ, in $\mathbb{C}.\texttt{KeyGen}$, a public key $PK_c = (n, g, h)$ is generated, where n is a special RSA modulus, $h \leftarrow QR_n, g \leftarrow \langle h \rangle$, where $\langle h \rangle$ is the group generated by h, QR_n denoting quadratic residue modulo n. The commitment $com = \mathbb{C}.\texttt{Commit}(x, r)$ for a string x uses a random string $r \in \mathbb{Z}_n$ and is computed as $com = g^x \times h^r \mod n$. In the reveal stage, the committer reveals the values x, r. The receiver can verify $com = \mathbb{C}.\texttt{Commit}(x, r)$. This commitment scheme is *statistically hiding*, and *computationally binding* assuming factoring is a hard problem.

3. www.zurich.ibm.com/idemix

Zero-knowledge proof of knowledge. Zero-knowledge proof of knowledge ($\mathbb{ZKPoK}$) is a protocol between a *prover* and a *verifier*, in which the prover convinces the verifier that they possess a certain quantity w that satisfies some polynomial-time computable relation R, without revealing any information about w. We use Camenisch and Stadler's [29] representation of proofs of knowledge of discrete logarithms, and proofs of the validity of statements about discrete logarithms. For example, $\mathbb{ZKPoK}\{(\alpha, \beta, \gamma) : y = g^\alpha h^\beta \wedge \tilde{y} = \tilde{g}^\alpha \tilde{h}^\gamma\}$ expresses a Zero-Knowledge Proof of Knowledge of integers α, β and γ s.t. $y = g^\alpha h^\beta$ and $\tilde{y} = \tilde{g}^\alpha \tilde{h}^\gamma$ are true, where $y, g, h, \tilde{y}, \tilde{g}$ and $\tilde{h}$ are elements of some groups $G = \langle g \rangle = \langle h \rangle$ and $\tilde{G} = \langle \tilde{g} \rangle = \langle \tilde{h} \rangle$. By convention, the Greek letters denote quantities the knowledge of which is being proved, while all other parameters are known to the verifier.

We also use Camenisch and Lysianskaya's digital signature scheme (CL-signature scheme [30]) that provides existential unforgeability, and a public key encryption scheme that provides indistinguishability under chosen ciphertext (IND-CCA) attack [31]. Backgrounds and notations for these primitives are in the supplemental material, in Appendix A.

2.2 Proof-of-Location

System model and entities. As shown in Fig. 3, we consider four types of entities in the system: *i) Users*, *ii) Infrastructure nodes* that issue *pol* to registered users, *iii) Verifiers* who can verify a *pol* that is presented by a registered user, and *iv) a trusted authority* who sets up the system parameters. The system works as follows.

User requests a proof-of-location (*pol*) from an infrastructure node. $\mathcal{U}$ denotes the set of users in the system. *Infrastructure* consists of a set of access points $\mathcal{AP} = \{ap_0, ..., ap_n\}$ that can issue *pol*s to the users. Infrastructure also includes a database server $DBase$ that stores the initial location information of these access points. *Verifiers* need to verify a user's past location information (e.g. to provide service to them). $\mathcal{V}$ denotes the set of verifiers in the system. *Trusted Authority (TA)* generates public parameters of the system and the keys, and registers the users.

Users, access points and verifiers are the *participants* in the system. Each participant has a public key that is issued by the TA, and a geo-location $\ell oc = (x, y) \in \mathbb{R} \times \mathbb{R}$ in a well defined coordinate system, with distances measured as planar Euclidean distance. The distance function $d(\ell oc_1, \ell oc_2)$ returns the distance between two locations ℓoc_1 and ℓoc_2. Given a distance threshold B, two participants located at $(\ell oc_1, \ell oc_2)$ are said to be *close-by* if $d(\ell oc_1, \ell oc_2) \leq B$, and *far-away* otherwise.

Clock. We assume nodes in the infrastructure and the verifier use a UTC (Universal Time Coordinate) to loosely synchronize their local clocks. UTC may not reach all the system entities at the same time due to atmospheric pressure, network(s) transmission time and software overhead in local OS [32]. We use interval timestamp to capture the uncertainty over measuring time using UTC. An issuer will use interval timestamp to specify the time interval of issuing a *pol*, and the verifier check their local interval time to check validity of the time information. The format of an

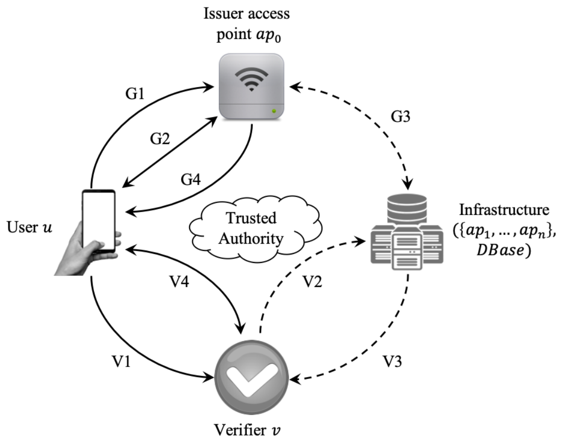


Fig. 3. Proposed proof-of-location system model. Trusted authority provides secret and public keys to the system entities. pol generation steps are: $G1$: user makes a pol request to ap_0; $G2$: ap_0 runs a DB protocol with u; G3: ap_0 generates proof of its own location integrity by communicating with neighboring APs; and G4: ap_0 issues the pol to the user. pol verification stage: V1: user presents the pol to a verifier; V2,V3: Verifier requests for and receives ap_0's neighborhood data from the server $DBase$; V4: ap_0 verifies location integrity and signature on the pol, runs a zero knowledge proof of knowledge protocol with the user to authenticate it. G3, V2 and V3 are for protection against geo-tampering - see Section 4.

interval timestamp t is $[t_1, t_2], t_2 > t_1$. The interval width is source-dependent - may vary from one entity to another. We will see in Section 4 that the infrastructure nodes also use interval timestamp to show the time of generation of "proof" of location integrity.

Trust assumptions. Users are dishonest; they can claim wrong location, or attempt to forge or transfer a proof-of-location. APs and the verifiers are honest-but curious and can attempt to link pols and users to infer location movement trajectories.

Each AP has a location loc that is stored in a database $DBase$. The location of a user is with respect to the location of the access point that issues the pol. When a user u requests a proof of location from an access point ap_0, the ap_0 runs a DB protocol with the user, which if successful guarantees that u is *close-by*. In its basic form, pol is the ap_0's digital signature on the statement "$[u]$ is within distance B from ℓoc_{ap_0}", ℓoc_{ap_0} being the location of ap_0. This information is captured in the transcript of the DB protocol.

User privacy is an important requirement of POL systems. This requires the users to use *pseudonyms*, to request pol. We use $[u]$ to denote this pseudonym that will be used to anonymously authenticate the user u. In the following, we first describe our computational model, *assuming the infrastructure nodes have correct location.*

Adversary (Computational). An adversary can corrupt a subset of participants $\mathcal{X}^* \subset \mathcal{U} \cup \mathcal{AP} \cup \mathcal{V}$. For each security property, the adversary has a defined goal, which is reflected as restrictions of $\mathcal{X}^*$; in *unforgeability* and *non-transferability*, $\mathcal{X}^* \subset \mathcal{U}$ and in *anonymity* $\mathcal{X}^* \subset \mathcal{AP} \cup \mathcal{V}$. We note that corrupting a participant refers to the adversary gaining full control on the participant's code.

In Section 4 we consider a *physical attacker* that tampers with physical location of infrastructure nodes.

Definition 1 (Proof-Of-Location Scheme POL). For a security parameter λ, a proof-of-location scheme (POL) is defined by a tuple ($\mathtt{POLInit}, \mathtt{POLJoin}, \mathtt{POLGen}, \mathtt{POLVer}$). $\mathtt{POLInit}(1^\lambda)$ generates the public and private parameters of the system (run by TA). POLJoin is an interactive protocol between user and TA, where TA registers a new user in the system and generates credentials for them. POLGen is an interactive protocol between a user $u \in \mathcal{U}$ and an access point $ap \in \mathcal{AP}$, that proceeds in two stages: i) POLGen.DB: a distance bounding protocol is run between u and ap_0, where ap_0 verifies if $d(\ell oc_{ap_0}, \ell oc_u) \leq B$, where B is the distance bound. ii) POLGen.issue: ap_0 issues a proof-of-location (pol) to the user u, which is ap_0's signature on a statement "$[u]$ is within distance B from ℓoc_{ap_0}", ℓoc_{ap_0} being ap_0's location. POLVer is an interactive protocol between a user $u \in \mathcal{U}$ and a *verifier* $v \in \mathcal{V}$. User u presents a proof-of-location pol to the *verifier* v. If the *verifier* is convinced that pol was issued by a valid issuer to the presenter of the proof, it outputs 1 (accept); otherwise it outputs 0 (reject).

POL Correctness. If all parties follow the protocols correctly, i.e. key generation and POLJoin are correctly executed, POLGen protocol is performed between an access point ap_0 that has trusted code and trusted location, and a close-by honest user u, the proof-of-location pol issued by the ap_0 and held by u, will be successfully verified by the *verifier* that runs POLVer protocol with u.

Proving security of cryptographic systems uses two main approaches, game-based and simulation based. While simulation based approach gives a more holistic view of security and allow composition of the proofs, we will use game-based approach because (i) security of distance bounding protocols that form a sub-protocol of our POL system has been studied using game based approach, and (ii) AP geo-tampering attack is an attack on physical location of APs and more amenable to modeling and analysis using game-based security. Simulation based models that include "physical" properties have been considered in [33]. However, no simulation-based model has been proposed for proximity verification.

In game-based approach the game is defined between the adversary and the challenger. The challenger initializes the system and provides oracle access to different parts of the system. The adversary's power is modelled by the set of their oracle accesses. We assume the adversary has the access to the following types of oracle queries.

1) $Corrupt(\mathcal{X}^*)$: An adversary can send a $Corrupt(\mathcal{X}^*)$ query to the challenger, asking to corrupt a subset of participants $\mathcal{X}^* \subset \mathcal{U} \cup \mathcal{AP} \cup \mathcal{V}$. The challenger returns the secret credentials of all participants in $\mathcal{X}^* \subset \mathcal{U} \cup \mathcal{AP} \cup \mathcal{V}$ to the adversary. Also, the codes and locations of these participants are set according to the adversary's instruction. The list $CorruptList$ stores the identities of the corrupted participants.
2) $POLGen(ap_0, u)$: The adversary selects an access point $ap_0 \in \mathcal{AP}$ and a user $u \in \mathcal{U}$, and makes an oracle query $POLGen(ap_0, u)$. The challenger runs the protocol POLGen as access point ap_0 with user u, and returns

either a proof-of-location pol, or $\perp$. If pol is returned, the tuple (pol, u) is appended to the list $GenList$.

3) $POLGenIssue(ap_0, u)$: The adversary selects an access point $ap_0 \in \mathcal{AP}$ and a user $u \in \mathcal{U}$, and makes an oracle query $POLGenIssue$ (ap_0, u). The challenger runs only the second stage of protocol `POLGen`, i.e., `POLGen.issue`, as access point ap_0 with user u, and returns a proof-of-location pol. The tuple (pol, u) is appended to a list $IssueList$.
4) $POLVer(v, u, pol)$: The adversary selects a verifier $v \in \mathcal{V}$ and a user $u \in \mathcal{U}$, and makes an oracle query $POLVer(v, u, pol)$ for a proof-of-location pol. The challenger runs the protocol `POLVer` as *verifier* v with user u on proof-of-location pol, and returns either 1 or 0. If 1 is returned, the tuple (pol, u) is appended to a list $VerList$.

We first define a general POL game and then show how it can be used to define each property.

Definition 2 (POL Game). For a security parameter λ, we define the following game between a challenger and an adversary.

1) *Initialize:* The challenger runs $\texttt{POLInit}(1^\lambda)$ and publishes the public parameters of the system. The challenger also initializes empty lists $CorruptList$, $GenList$, $IssueList$ and $VerList$.
2) *Generate participants:* The challenger generates a set of m users ($\mathcal{U}$), a set of n access points ($\mathcal{AP}$), and a set oof q verifiers ($\mathcal{V}$). The credentials (i.e., public/private key pairs) are generated for all the verifiers and access points. The locations of these participants are set arbitrarily. Then the challenger runs `POLJoin` for all the users. The challenger publishes the list $\mathcal{U}, \mathcal{AP}$ and $\mathcal{V}$ (i.e., the public credentials as well as location of each participant).
3) Queries: Adversary makes queries to oracles $Corrupt(\mathcal{X}^*)$, $POLGen(ap_0, u)$, $POLGenIssue(ap_0, u)$ and $POLVer(v, u, pol)$.
4) Adversary's output: The adversary outputs a proof-of-location pol_A.

The properties for POL are defined based on the POL Game. Conditions to win the game however vary depending on the property. We define three POL properties: unforgeability, non-transferability and anonymity. The following definition were motivated in Section 1, and *assume integrity of the access points' locations (access points are always assumed to correctly perform the computation).*

Property 1 (POL Unforgeability). Consider a POL scheme POL and a POL Game with the following restrictions: the adversary can only corrupt users, i.e., $\mathcal{X}^* \subset \mathcal{U}$ in the $\texttt{Corrupt}(\mathcal{X}^*)$ query. An adversary succeeds in the game, if there exists an entry $(pol, .) \in VerList$, s.t., $pol = pol_A$, and any of the following two holds: i) There does not exist an entry $(pol, .) \in GenList$ s.t. $pol = pol_A$, and there does not exist an entry $(pol, .) \in IssueList$ s.t. $pol = pol_A$; ii) There exists an entry $(pol, .) \in GenList$ s.t. $pol = pol_A$, and $d(\ell oc_{ap_0}, \ell oc_u) > B$. A proof-of-location scheme POL provides unforgeability, if the advantage of the adversary in succeeding in the above game, denoted by Adv_{UF}, is negligible.

Property 2 (POL Non-transferability). Consider a proof-of-location scheme POL and a POL Game with the following restriction: the adversary can only corrupt users, i.e., $\mathcal{X}^* \subset \mathcal{U}$ in the $\texttt{Corrupt}(\mathcal{X}^*)$ query. An adversary succeeds in the game, if there exists an entry $(pol, u) \in VerList$, s.t., $pol = pol_A$, and any of the following two holds: i) There exists an entry $(pol, u') \in GenList$ s.t. $pol = pol_A$, and $u' \neq u$; ii) There exists an entry $(pol, u') \in IssueList$ s.t. $pol = pol_A$, and $u' \neq u$. A proof-of-location scheme POL provides non-transferability if the advantage of the adversary in succeeding in the above game, denoted by Adv_{NT}, is negligible.

Property 3 (POL Anonymity). Consider a proof-of-location scheme POL and a POL Game, with the restriction that the adversary can only corrupt access points and *verifier*s, i.e., $\mathcal{X}^* \subset \mathcal{AP} \cup \mathcal{V}$ in the $Corrupt(\mathcal{X}^*)$ query. POL Anonymity is twofold.

1) *Anonymity with respect to the access point*: We remove step 4 in the POL game, and add the following steps. *i)* The adversary chooses a pair of users $(u_0, u_1) \in \mathcal{U}$, an access point $ap \in \mathcal{AP}$, and sends its choice of the participants to the challenger. However, the pair of users chosen by the adversary must be either both *close-by* the access point ap, or both being *far-away* from ap (otherwise, the adversary can win the game by simply looking at the output of the oracle in the following step). *ii*) The challenger randomly selects a bit $b_{ap} \leftarrow \{0, 1\}$, and simulates $POLGen(ap, u_{b_{ap}})$ oracle. The oracle returns either $\perp$ or a proof-of-location pol, which is forwarded to the adversary. In addition, the transcript of the protocol execution (i.e., in this case, all the messages exchanged between $(ap, u_{b_{ap}})$ in the `POLGen` protocol) is also sent to the adversary. *iii*) The adversary outputs a bit $\hat{b_{ap}} \in \{0, 1\}$.
2) *Anonymity with respect to verifier*: The game is the same as above, with the following modifications in the additional steps: *i)* the pair of users $(u_0, u_1) \in \mathcal{U}$ that are chosen by the adversary must be both *close-by* the access point ap (otherwise the game cannot proceed to pol verification stage), and *ii)* the challenger first simulates $POLGen(ap, u_{b_v})$ oracle, where b_v is a random bit picked by challenger, and then simulates $POLVer(v, u_{b_v}, pol)$ oracle, where $v \in \mathcal{V}$ is chosen by the adversary, and pol is the proof-of-location resulted from $POLGen$ query. The adversary's final output is a bit $\hat{b_v} \in \{0, 1\}$.

Adversary's advantage in winning the game is expressed as a tuple $(Adv^{ap}_{AN}, Adv^{v}_{AN})$, where, $Adv^{ap}_{AN} = |Pr[\hat{b_{ap}} = b_{ap}] - \frac{1}{2}|$ and $Adv^{v}_{AN} = |Pr[\hat{b_v} = b_v] - \frac{1}{2}|$. The proof-of-location scheme POL achieves anonymity with respect to the access point ap if Adv^{ap}_{AN} is negligible, and the *verifier* v if Adv^{v}_{AN} is negligible.

Note that anonymity with respect to ap implies anonymity with respect to the location infrastructure as this is the only part of the infrastructure that interacts with the user, and transcripts of different POL sessions are statistically independent. Following similar arguments, anonymity w.r.t v implies anonymity with respect to the set of verifiers in the system, $\mathcal{V}$.

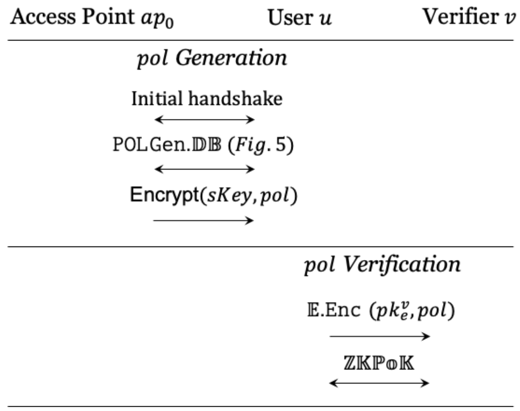


Fig. 4. An overview of pol generation (POLGen) and pol verification (POLVer) protocols in POLA scheme. Here $pol = \langle sig, msg \rangle, sig \leftarrow \mathbb{DS}.\texttt{Sig}(sk_s^{ap_0}, msg), msg = \langle com, pk_s^{ap_0}, \ell oc_{ap_0}, t \rangle$. See Fig. 5 for $com, sKey$ generation.

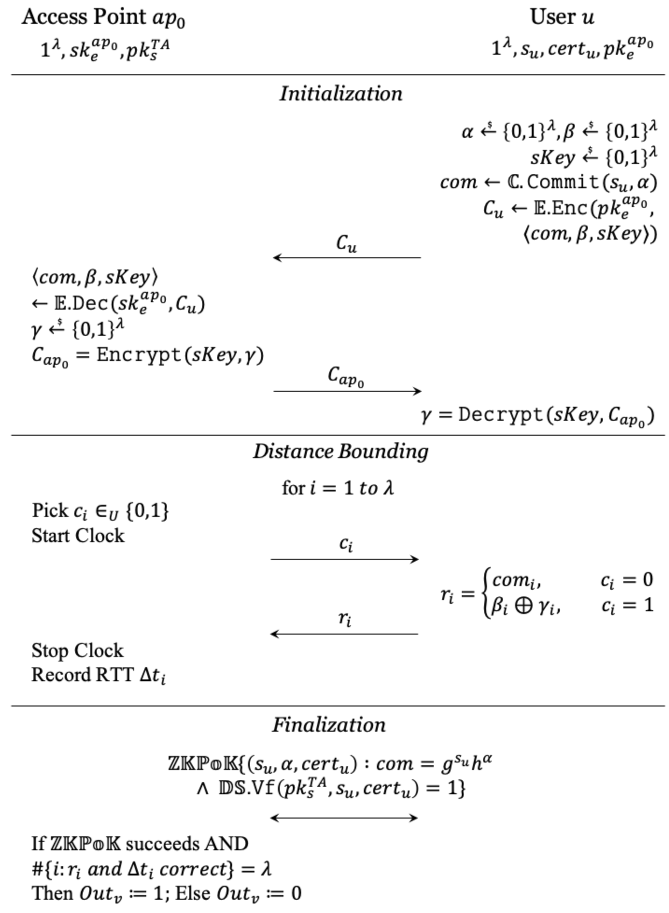


Fig. 5. The distance bounding protocol (POLGen.DB) in POLA scheme between user u and access point ap_0.

3 POLA: Proof-of-Location with Issuer and Verifier Anonymity

POLA is a proof-of-location scheme with anonymity against the issuer and the verifier. We assume each access point has a registered keypair (sk_s^{ap}, pk_s^{ap}) of the digital signature scheme $\mathbb{DS}$ that provides security against existential forgery. The public verifying key pk_s^{ap} is known by all the verifiers in the system. Also, each access point (respectively each verifier) has registered keypair (sk_e^{ap}, pk_e^{ap}) (resp. (sk_e^v, pk_e^v) for verifier) of the encryption scheme $\mathbb{E}$ that provides CCA security. The public encryption keys (pk_e^{ap}, pk_e^v) are known by all the registered users in the system. Fig. 4 gives an overview of pol generation and pol verification protocols in POLA scheme. Details of the scheme follows.

Initialization (POLInit).

TA calls $\mathbb{DS}.\texttt{KeyGen}(1^k)$ to generate (sk_s^{TA}, pk_s^{TA}), a private/public key pair of the digital signature scheme.

User registration (POLJoin).

Upon receiving join request from a user with identity u, TA generates a random binary string as the user's long term secret s_u. TA digitally signs s_u to generate the certificate $cert_u = \mathbb{DS}.\texttt{Sig}(sk_s^{TA}, s_u)$ and provides the private credential $(s_u, cert_u)$ to the user.

Proof-of-location generation (POLGen).

Access point discovery and initial handshake. APs can be discovered in several ways by the user. One way is that APs advertise their capability of providing proof-of-locations through periodically transmitted beacons, and user scans for available APs and identify them [34]. Once the public key of a *close-by* access point is known, the user performs an initial handshake with the access point (i.e., user requesting a pol and AP acknowledging the request), and proceeds to the distance bounding stage.

Distance bounding between u and ap_0 (POLGen.DB). The user and the access point start the distance bounding protocol as shown in Figure 5. The protocol has three phases. Let λ be the security parameter.

1) Initialization Phase: u chooses three random strings $\alpha, \beta, sKey$ each of length λ, where α is used to generate a commitment on s_u, $com = \mathbb{C}.\texttt{Commit}(s_u, \alpha)$. $sKey$ is a session key for encrypting (i.e., AES symmetric key encryption) subsequent messages sent by ap_0 to the user. User encrypts $\langle com, \beta, sKey \rangle$ using ap_0's public key $pk_e^{ap_0}$ and the result C_u is sent to ap_0, who decrypts C_u using $sk_e^{ap_0}$ and obtains $\langle com, \beta, sKey \rangle$. ap_0 chooses a random string γ of length λ, encrypts using $sKey$, and sends to u, who decrypts it to obtain γ. The values com, β and γ will be used by the user in responding to ap_0's challenges in the distance bounding phase.

2) Distance Bounding Phase: This phase has λ rounds. In the i-th round of this phase ($1 \leq i \leq \lambda$), ap_0 sends a uniformly chosen random bit c_i to u. User immediately replies with com_i (if $c_i = 0$) or $\beta_i \oplus \gamma_i$ (if $c_i = 1$). Here x_i represents the i-th bit of the binary string x of length λ. The round trip time Δt_i for the challenge-response is measured and stored by ap_0.

3) Finalization Phase: This stage starts with a zero-knowledge proof of knowledge protocol between user and access point, where the user proves that com is a valid commitment of a value s_u that is certified by the TA. We follow the protocol proposed by Camenisch and Lysyanskaya ([30], Sec 6.2, Fig. 2), which is a zero knowledge proof of knowledge of the values $(s_u, \alpha, cert_u)$ such that $com = g^{s_u} h^\alpha \mod \gamma$ and $\mathbb{DS}.\texttt{Vf}(pk_s^{TA}, s_u, cert_u) = 1$. If the zero-knowledge proof of knowledge protocol succeeds,

ap_0 checks the user's responses from distance bounding phase. If the estimated round trip times are within bound (for distance bound B) and responses are correct in all the rounds, then ap_0 outputs 1, otherwise 0.

Issuing proof-of-location (`POLGen.Issue`*).* If distance bounding protocol succeeds and outputs 1, ap_0 generates a signature on the message msg, described below, using signing key $sk_s^{ap_0}$ of the digital signature scheme, and concatenate msg to this signature to output proof-of-location pol. The message msg is,

$$msg = \langle com, pk_s^{ap_0}, \ell oc_{ap_0}, t\rangle$$

where, ℓoc_{ap_0} is the location of ap_0, t is the interval timestamp at the issuer. pol is encrypted using $sKey$ and sent to the user.

$ap_0 \rightarrow u : \texttt{Encrypt}(sKey, pol), pol = \langle sig, msg\rangle, sig \leftarrow \mathbb{DS}.\texttt{Sig}(sk_s^{ap_0}, msg)$

Proof-of-location verification (`POLVer`**).**

User u presents pol to the *verifier* v, encrypted with the *verifier*'s public key pk_e^v. The *verifier* rejects the claim (outputs 0) if, after decrypting the values, $\mathbb{DS}.\texttt{Vf}(pk_s^{ap_0}, sig, msg) = 0$. If the issuer's signature is verified, the verifier v runs the $\mathbb{ZKP}\text{o}\mathbb{K}$ protocol in [30] (similar to the finalization phase of `POLGen.DB`). Verifier rejects the claim if the protocol fails (outputs 0), otherwise v is convinced that u knows the secret s_u used to generate the commitment com and that this secret has been certified by the TA, and accepts the claim (outputs 1).

Discussion. In POLA, issuing a proof of location requires the issuer to run a DB protocol. To provide anonymity for the POL system, one can use an existing *anonymous* DB protocol [24], [25], [26], [27]. This however will not be sufficient because, during the verification phase, the generated pol should be linkable to the identity credential $[u]$ of the prover. These protocols do not provide this additional property. We will prove that while two *pol*s that are issued to the same user in two distinct sessions, remain unlinkable, each individually will be linkable to the credential of the user. Our proposed anonymous DB protocol `POLGen.DB` satisfies these properties.

3.1 Security Analysis

Correctness of POLA can be straightforwardly shown. For security, we first show that the DB protocol between u and ap_0 is secure against distance fraud, distance hijacking, mafia fraud and terrorist fraud attack (See the supplemental material, Appendix B for attack descriptions). We use this to prove security of the protocol in Theorem 1.

Theorem 1. Let $\mathbb{E}$ be a IND-CCA secure encryption scheme, $\mathbb{C}$ be a computationally binding and computationally hiding commitment scheme, $\mathbb{DS}$ be a digital signature scheme secure against existential forgery and the protocol $\mathbb{ZKP}\text{o}\mathbb{K}$ is sound and zero knowledge proof of knowledge of the values $(s_u, \alpha, cert_u)$. Then,

a) The distance bounding protocol (`POLGen.DB` in Fig. 5) in POLA between a user u and an access point ap_0, is secure against distance fraud, distance hijacking, mafia fraud and terrorist fraud attack.
b) If `POLGen.DB` in POLA is secure against above four types of attacks, then POLA is unforgeable.
c) Assuming the user is not willing to share their secret credential, POLA provides non-transferability.
d) POLA is anonymous with respect to both the issuer access point and the verifier.

Proof is in the supplemental material, Appendix C.

4 Geo-tampering attack on POL systems

Consider a POL system in Section 2 with a location infrastructure $\mathcal{AP}$ consisting of n access points $\{ap_0, ap_1, ..., ap_{n-1}\}$, each associated with a location ℓoc_{ap_i} $(i = 0, ..., n-1)$. The initial location map $LocMap = \{(ap_0, \ell oc_{ap_0}), ..., (ap_{n-1}, \ell oc_{ap_{n-1}})\}$ of the APs is stored in $DBase$.

In geo-tampering attack, an access point will have intact hardware and software but its location has been modified. Let ℓoc_{ap_0} be the geo-coordinate of an access point ap_0 in $LocMap$, and $\ell oc'_{ap_0}$ be its modified geo-coordinate. The distance bounding protocol will accept the claim of a user within $Circ(\ell oc'_{ap_0}, B)$, a circle of radius B around the location $\ell oc'_{ap_0}$, and issue a pol using its stored location ℓoc_{ap_0}. This is effectively forging a pol for a location that the user is not in.

One can use an approach such as trilateration to determine the location of the proof issuing AP, and compare it with the corresponding value that is stored in $LocMap$. This is Geo-tampering detection using *location determination*.

Geo-tampering detection using *location determination*. Assume there are three access points (ap_1, ap_2, ap_3), all with trusted location (e.g. physically secured devices) that are at the line-of-sight distance of ap_0. Assuming synchronized clocks, ap_0 sends a radio signal to ap_1, ap_2, ap_3, who record their signal arrival time, and send it to ap_0 who can use the received values to estimate the distances $d(ap_0, ap_i), i = 1, 2, 3$ using the travel time, the (constant) speed of radio signal. Using these, and the geo-coordinates of ap_1, ap_2, ap_3, one can find an estimate for the geo-coordinate of ap_0, using a trilateration algorithm (such as the one proposed in [35]). The new computed location of ap_0 will have some error because of inaccuracy of distance measurement, that is called *trilateration error*. A *geo-tampering* will be detected if the distance between the initial stored value of the ap_0's location in $LocMap$, and its new estimate exceeds a chosen *trilateration error*.

Drawback of *location determination* based approach. Trilateration requires at least three access points with trusted locations, that are at the line of sight distance (reach of radio signal) of the pol issuing AP. These requirements can be fulfilled by providing many APs with trusted location (e.g. physically protected areas) which makes the infrastructure nodes expensive for many applications.

4.1 Our approach

We propose a novel approach to detect geo-location tampering that does not require determining the location of pol issuing AP, but relies on detecting the change in the AP's relative position to its neighbors. Here the notion of "neighbor" can be defined in a flexible way, and is not restricted to nodes that are at the line-of-sight of the issuing AP. We define the neighborhood of ap_0 to be the set of all APs that are "reachable" from ap_0, where reachability means existence of a path consisting of edges that each correspond to a line of sight communication. That is, a message sent from ap_0 will reach all the nodes in the neighborhood of ap_0, possibly through multiple "hops". We assume a subset of APs in the ap_0's neighborhood have not been displaced and have correct (original) locations. We however *do not require these nodes with correct locations, to be identifiable*.

Definition 3 (Neighborhood N_{ap_0}). Let $G = [\mathcal{V}, \mathcal{E}]$ be an undirected graph where each vertex $v \in \mathcal{V}$ represents an AP in the infrastructure of the proof-of-location system, and an edge $e \in \mathcal{E}$ between two APs represents the two being at the line-of-sight of each other. A neighbourhood of ap_0 is a connected component of graph G that includes ap_0. If $N_{ap_0} = \{ap_1^0, ap_2^0, .., ap_{n-1}^0\}$ is the neighborhood of ap_0, then for each $ap_i^0 \in N_{ap_0}$, there is a path from ap_0.

For a neighbourhood N_{ap_0}, we define an Euclidean Distance Matrix (EDM [36]) $D_{N_{ap_0}}$ as follows.

Let $|N_{ap_0}| = n$ (Neighborhood includes ap_0). $D_{N_{ap_0}}$ is a $n \times n$ matrix, with rows and columns labelled by $ap_i^0 \in N_{ap_0}$ and $D(i,j)$ is the length of the straight line connecting ap_i^0 and ap_j^0. This matrix can be constructed by knowing the exact coordinates of the nodes (that can be found in $LocMap$ stored in $DBase$) and calculating the Euclidean distance between them. The geo-tampering detection algorithm has two steps: (i) construct $D'_{N_{ap_0}}$, a real-time estimate of the matrix $D_{N_{ap_0}}$, and (ii) compare the entries of the two matrices, and use a decision algorithm to detect tampering. An overview of each step is given below. Full details and algorithms are in Sec. 4.3.

(i) Constructing $D'_{N_{ap_0}}$ is through making real-time distance measurements between each pair of nodes in N_{ap_0}. The distance measurement will be by recording the arrival time of a radio signal that is sent by one node associated to an edge, and received by the second node of that edge. Note that this measurement can be performed for only nodes that are in the radio distance of each other. Thus only the entries of $D'_{N_{ap_0}}$ that correspond to nodes that are at radio-distance of each other can be measured (possibly with some error). The entries of $D'_{N_{ap_0}}$ that can not be measured can be "reconstructed" using *distance recovery* algorithm [36]. This recovery is due to the geometric properties of distances of neighboring nodes and is available to any distance matrix. The error that is introduced in this reconstruction depends on the error in the distance measurements using radio signal, and the number of entries that cannot be found through the radio signal measurement.

(ii) Given $D_{N_{ap_0}}$ and its estimation $D'_{N_{ap_0}}$, one can use various decision algorithms to detect tampering. We use a simple threshold algorithm which requires the corresponding entries of the two matrices to be within distance Δ of each other. That is, $|D_{N_{ap_0}}(i,j) - D'_{N_{ap_0}}(i,j)| \leq \Delta, \quad (i,j) \in \{0,..,n-1\}$.

Notice that small values of Δ implies reduced advantage for the attacker, in terms of its ability to displace the access points. Δ is called the *Geo-tampering detection threshold*, defined bellow.

Definition 4 (Geo-tampering threshold Δ). For a geo-tampering detection algorithm, Δ is the maximum distance by which the geo-tampering attacker is able to move an access point from its original location, without getting detected.

Δ is determined by (i) error in distance measurement between access points in the neighborhood, and (ii) error in reconstructing the entries in $D'_{N_{ap_0}}$ that could not be measured. Hence, for our detection algorithm, Δ is a tuple (Δ_M, Δ_R); Δ_M is the distance measurement error, and Δ_R is the distance reconstruction error.

The rest of this section provides details of our approach in geo-tampering detection, and the construction of POLA$^+$, an extension of POLA with protection against geo-tampering.

4.2 POLA$^+$: Protection against geo-tampering

Protection against geo-tampering requires the *pol* issuer to perform a real-time integrity checking algorithm to provide location integrity proof information. Let $LocIntInfo(ap_0)$ denote the location integrity information ("proof") that must be provided by the issuing AP ap_0 to convince the POL system that ℓoc_{ap_0} is correct.

We require the following properties for $LocIntInfo(ap_0)$.

(P1) It must be generated at the time of issuing *pol* to the user.
(P2) It must convince the POL system that the geo-location of ap_0 with respect to $LocMap$ is correct.

These requirements allow a modular approach to the construction of a POL system that provides security against tampering of the infrastructure node locations, based on a secure POL system that requires trusted location for the infrastructure nodes. Theorem 2 below extends Theorem 1 (POLA security) such that geo-tampering with the location of APs can be tolerated.

Theorem 2. Let $LocIntInfo(ap_0)$ satisfy P2 (have sufficient information to convince the verifier about the integrity of ap_0's geo-location). By generating and including $LocIntInfo(ap_0)$ at the time of generating *pol* (and thus satisfying P1), and including it in *pol* as shown below:

$$pol \leftarrow \langle sig, msg \rangle, sig \leftarrow \mathbb{DS}.\mathtt{Sig}(sk_s^{ap_0}, msg)$$

$$msg = \langle com, pk_s^{ap_0}, \ell oc_{ap_0}, t, LocIntInfo(ap_0) \rangle$$

we obtain a POL system that satisfies POL security properties, i.e., unforgeability (property 1), non-transferability (property 2) and anonymity (property 3), while providing security against *geo-tampering attack*.

Proof is in the supplemental material, Appendix A.

4.3 LocIntInfo(ap_0) Generation and Verification

Protection against geo-tampering is achieved by, (i) LocIntInfo(ap_0) Generation: at the time of pol generation, ap_0 constructs $D'_{N_{ap_0}}$, a real-time estimate of the matrix $D_{N_{ap_0}}$, and appends $D'_{N_{ap_0}}$ to the msg before digitally signing it to form the pol with the ap_0's location integrity information $LocIntInfo(ap_0)$, and (ii) LocIntInfo(ap_0) Verification: at the time of pol verification, verifier compares the entries of the two matrices, and uses a decision algorithm to detect tampering.

4.3.1 LocIntInfo(ap_0) Generation

Construction of $D'_{N_{ap_0}}$ requires real-time distance measurement between each pair of nodes in N_{ap_0}, that are at each other's line-of-sight. Any entry in $D'_{N_{ap_0}}$ corresponding to a pair of nodes that are not at LOS, will be left empty.

Measuring distance between two line-of-sight APs. Since both APs correctly follow the prescribed computation by the protocol (even if they are geo-tampered), we do not require cryptographically secure distance measurement techniques such as distance bounding. One can use distance measurement techniques that are commonly used in wireless sensor networks, including Time of Flight (TOF) or Time of Arrival (TOA), for this purpose. We use TOA as it uses a one-way signal for time measurement and has less communication overhead.

Algorithm GenLocIntInfo (Fig. 7) is used to estimate the distance between ap, ap' (an edge in $D'_{N_{ap_0}}$). In brief, ap sends a signed message to ap' that includes the sending time t_s. ap' verifies the signature, and if valid computes the distance between the two as $d'(ap, ap') = (t_r - t_s - \delta_p) \times c$, where t_r is the arrival time of the message, c is the speed of light and δ_p is the computation delay of the sender (digital signature). The computation delays are assumed publicly known.

TOA distance measurement can be affected by clock drift between two access points. To reduce the clock drift techniques such as those proposed in "PinPoint" [37] can be used to improve distance measurement accuracy. In PinPoint, multiple rounds of timestamp exchange between the two access points is used to estimate and remove the difference in the two clock readings. PinPoint provides an average accuracy of 4.18 feet, with a standard deviation of 4.4 in a complex indoor environment.

Selecting Effective Neighborhood. In practice, only a selected subset of the actual neighborhood N_{ap_0} may be used in generating location integrity information. This subset is selected such that the performance of the geo-tampering detection algorithm is improved in terms of geo-tampering threshold Δ. Also, to improve computation time of the location integrity information, it is important to select access points that have shorter path-lengths to issuer ap_0. Below we define effective neighborhood for ap_0.

Definition 5 (Effective Neighborhood EN_{ap_0}). An effective neighborhood EN_{ap_0} of access point ap_0 is a subset of ap_0's neighborhood N_{ap_0} and includes ap_0. This is a connected subgraph of N_{ap_0} (see Def. 3 for N_{ap_0}). EN_{ap_0} includes all the nodes that are used to form ap_0's location integrity information $LocIntInfo(ap_0)$.

By "connected subgraph of N_{ap_0}", we mean a subgraph of the connected component N_{ap_0}, such that each pair of vertices in it are connected by a path.

The distance matrix formed by EN_{ap_0} is $D_{EN_{ap_0}}$. As will be shown in theorem 4, the geo-tampering attack is detected with high probability if at least three untampered access points are present in the effective neighborhood. Therefore, when the risk of nodes being displaced in a system is higher (such as applications where APs are placed in public places), it will require having a larger effective neighborhood – that will increases the chance of having at least three untampered APs in the effective neighborhood.

Effective Neighbourhood Selection. An effective neighborhood selection algorithm will first choose *the size of the effective neighbourhood, m,* and then select an effective neighborhood of that size. The effective neighborhood size will depend on the estimated probability of neighbourhood nodes being moved by an adversary, with higher probabilities corresponding to larger size of m. Choosing the value of m will depend on other factors including the required security guarantee of the system and will not be further studied here. For a given value of m, the effective neighbourhood will be a set of size at least m nodes, each connected to ap_0 through a path. The choice of the set will affect (i) the geo-tampering error threshold Δ, and (ii) the delay in the generation of location integrity proof, determined by the node with the longest path to ap_0 (assuming equal transmission time on each link). In the following we consider the problem of selecting a set that minimizes Δ, and show that it reduces to the hard problem of finding a *clique* of size k in an undirected graph (see below). Selecting a set that takes both objectives of reducing Δ and maximum path length will be an extension of this work.

Minimizing Δ for a size m Neighborhood . Finding an effective neighborhood of size m that minimizes Δ implies minimizing the EDM reconstruction error, which in turn implies minimizing the number of missing elements in the matrix $D_{EN_{ap_0}}$, therefore maximizing the number of edges in the effective neighborhood EN_{ap_0}. Now, finding an effective neighborhood of size m with maximum number of edges, is NP-hard, as stated in the following theorem.

Theorem 3. Finding an effective neighborhood of ap_0 of size m that has maximum number of edges, is NP-hard.

Proof:

A clique is a subset of vertices in an undirected graph such that every two vertices are connected by an edge, forming a complete subgraph. The k-clique problem is stated as follows: *given an undirected graph G, find a clique of size k, where k is the number of vertices.* The k-clique problem has been shown to be NP-complete [38].

Finding ap_0's effective neighborhood of size m that maximizes number of edges, can be stated as follows: "Given a neighbourhood of ap_0 consisting of all nodes that are connected through a path to ap_0, find a subset of size m, each connected to ap_0, where the number of edges is maximum."

Assume there is a polynomial time algorithm that solves the above neighborhood selection problem. This algorithm can also find a subgraph of size m vertices that include ap_0, and includes all edges. That is, it will find a clique of size m that includes ap_0, if there exists one. Now it is easy to see that by repeating the algorithm for each vertex of the graph, one obtains an efficient algorithm that finds a clique of size m in the graph. This is a contradiction to the NP-completeness of k-clique problem. □

Our approach to selecting effective neighborhood of size m. For our experiment we consider a heuristic approach (Algorithm in Fig. 6) that works as follows. Let M be a $n \times n$ matrix of all APs in the neighborhood of ap_0, including ap_0. In M, each element $M(i,j)$ can take value in $\{0,1\}$, 1 denoting the event that real-time distance measurement is possible between (ap_i, ap_j), and 0 denoting the opposite - a missing element, and must be reconstructed. We sort the APs in N_{ap_0} according to the number of missing elements involved with the AP. The algorithm sorts the APs in N_{ap_0} according to the number of zeros in the row/column of M indexed by the APs. Finally, given the required effective neighborhood size m, simply select the first $(m-1)$ APs (excluding ap_0) from the sorted list in ascending order, such that there is a path from ap_0 to each of these APs. In order to find if two nodes are connected (i.e, if there is a path between these nodes) in an undirected graph, one can simply use a graph traversal algorithm such as Depth-First-Search as shown in [39]. Note that the issuing AP ap_0 is by default included in the effective neighborhood.

Input: ap_0's neighborhood N_{ap_0} of size n, location map $LocMap$, required size of effective neighborhood: m
Output: Effective neighborhood EN_{ap_0} of size m

1: Include issuer AP ap_0 to the effective neighborhood EN_{ap_0}.
2: Create a $n \times n$ matrix M, for neighborhood N_{ap_0}, with rows and columns labelled by $ap_i \in N_{ap_0}$. Each element $M(i,j)$ will take value in $\{0,1\}$. 0 denotes an edge between ap_i, ap_j, and 1 denotes absence of the edge.
3: Count the number of zeros in each column (or row) of M, and assign this number to the AP that labels this column (or row, respectively).
4: Excluding ap_0, sort the APs in N_{ap_0} according to the assigned number (i.e., count of zeros).
5: Select the first $(m-1)$ APs in ascending order, such that there is a path from ap_0 to each of these APs. Add these APs to EN_{ap_0}.
6: **return** EN_{ap_0}.

Fig. 6. Effective neighborhood selection algorithm.

Generating location integrity information. The nodes in the chosen effective neighborhood set of ap_0 measure the pairwise distances among themselves, and send the results back to ap_0. A path from ap_i to ap_j is a finite sequence of edges between ap_i and ap_j. Algorithm GenLocIntInfo (Fig. 7) uses TOA technique to generate $D'_{EN_{ap_0}}$. We define Adj_{ap} as the adjacent list of ap, that is the list containing all access points in radio range of ap. An edge between two APs (ap_i, ap_j) is denoted by $\{ap_i, ap_j\}$.

(Step 1) ap_0 determines its neighborhood using $LocMap$, then selects its effective neighborhood EN_{ap_0} by running algorithm in Fig. 6. A distance matrix $D'_{N_{ap_0}}$ is initiated with all zero entries. Then ap_0 generates and broadcasts a signed location integrity request Req, that contains a sequence number $ReqID$, `Requester` and `Sender` name (both set as ap_0), the effective neighborhood list `ENList` (set as EN_{ap_0}), adjacent list `SenderAdjList` (set as Adj_{ap_0}), and the time of sending this message `SenderTime` (set as ap_0's local time t_{ap_0}).

(Step 2) Access point ap_i receives Req, records the receive time (`ReceiverTime`) and checks if itself and the `Sender` both are members of `ENList`. If so, it computes the length of the edge between them (`EdgeLength`) using `ReceiverTime`, `SenderTime` and sender processing delay δ_p. Then it computes a path P back to the Requester, and transmits a signed response Res containing `Destination` (set as `Requester`), `Responder` (set as ap_i), `Path` (set as P), `Edge` (set as $\{$`Sender`, `Responder`$\}$) and `EdgeLength`.

ap_i also rebroadcasts the message Req so that it will reach all the member of effective neighborhood of ap_0. However, instead of using blind flooding (i.e., each node broadcast the message whenever it receives it) which would waste wireless resource considerably, we use the "Self-pruning" method in [40]. Following this method, ap_i checks if all its adjacent nodes have already received Req. If so, it refrains from re-broadcasting Req. Otherwise, it resets `Sender` as ap_i, `SenderAdjList` as its own adjacent list, `SenderTime` as its own local clock time, then signs and forwards the modified Req.

(Step 3) ap_0 receives Res, it updates the element in $D'_{N_{ap_0}}$ corresponding to the `Edge` in Res and discards future Res with same `Edge` value. If all the elements in $D'_{EN_{ap_0}}$ are updated (excluding the elements that corresponds to absence of an edge), ap_0 outputs location integrity data that includes $EN_{ap_0}, D'_{EN_{ap_0}}$ and the algorithm completion time t_{int}.

4.3.2 LocIntInfo(ap_0) Verification

Verifier, upon receiving $LocIntInfo(ap_0)$ as part of the pol from a user, runs algorithm VerLocIntInfo (Fig. 8). Basically this algorithm reconstructs any missing element in $D'_{EN_{ap_0}}$, then compares the entries of the two matrices ($D_{EN_{ap_0}}, D'_{EN_{ap_0}}$), and uses a decision algorithm to detect tampering.

Reconstructing missing entries. The distance matrix $D'_{EN_{ap_0}}$ is an EDM. As noted earlier, the real-time distance measurements using Algorithm GenLocIntInfo (Fig. 7) may not obtain all the distances and corresponding entries in $D'_{EN_{ap_0}}$ will be missing. EDM geometric constraints allow recovery of missing distances.

We used the algorithm "Alternating Descent" in [36] (see the supplemental material, Appendix D) that can complete an EDM with high success probability when the number of missing elements in the matrix is bounded.

Comparing the two matrices. Once all distances of $D'_{EN_{ap_0}}$ that correspond to the stored distances of $D_{EN_{ap_0}}$ are recovered, a matching algorithm is used to decide if the effective neighborhood of ap_0 has been tampered with. Note $D'_{EN_{ap_0}}$ does not match $D_{EN_{ap_0}}$ could be because of changes in one or more node in the neighborhood. The matching algorithm in our experiments is by simply comparing every pair of

Input: $LocMap$ containing system AP locations, sender processing delay δ_p for TOA, Effective neighborhood size m
Output: Location integrity data $LocIntInfo(ap_0)$

1: Issuing AP ap_0 does the following:
   i. Determines its neighborhood N_{ap_0} from $LocMap$.
   ii. Selects effective neighborhood EN_{ap_0} of size m by running Algorithm in Fig. 6.
   iii. Initializes an empty $m \times m$ distance matrix $D'_{N_{ap_0}}$. Elements in this matrix that correspond to absence of an edge between the APs indexing the element, are set as 0, representing a missing element.
   iv. Selects a unique sequence number $ReqID$ for the message it is going to broadcast. Sets $\texttt{Requester} = ap_0, \texttt{Sender} = ap_0, \texttt{ENList} = EN_{ap_0}, \texttt{SenderAdjList} = Adj_{ap_0}, \texttt{SenderTime} = t_{ap_0}$. Then broadcast a request: $Req = \langle m, \mathbb{DS}.\texttt{Sig}(sk_s^{ap_0}, m)\rangle, \quad m = \langle ReqID, \texttt{Requester}, \texttt{Sender}, \texttt{ENList}, \texttt{SenderAdjList}, \texttt{SenderTime}\rangle$

2: ap_i that receives a message Req, records the reception time ReceiverTime, and does the following:
   i. If $ap_i \in EN_{ap_0}$ and Sender $\in EN_{ap_0}$, then computes $\texttt{EdgeLength} = (\texttt{ReceiverTime} - \texttt{SenderTime} - \delta_p) \times c$. Then computes $\texttt{Path} = P$ to Requester, sets $\texttt{Destination} = \texttt{Requester}, \texttt{Responder} = ap_i, \texttt{Edge} = \{\texttt{Sender}, \texttt{Responder}\}$, selects a unique sequence number $ResID$, and transmits following response: $Res = \langle m', \mathbb{DS}.\texttt{Sig}(sk_s^{ap_i}, m')\rangle, \quad m' = \langle ResID, ReqID, \texttt{Responder}, \texttt{Destination}, \texttt{Path}, \texttt{Edge}, \texttt{EdgeLength}\rangle$. This response will be forwarded along the path P to ap_0.
   ii. Computes own adjacent list Adj_{ap_i}. If $Adj_{ap_i} - \texttt{SenderAdjList} - \{\texttt{Sender}\}$ is empty, sets $\texttt{Sender} = ap_i, \texttt{SenderAdjList} = Adj_{ap_i}, \texttt{SenderTime} = t_{ap_i}$. Then forwards the modified request: $Req = \langle m, \mathbb{DS}.\texttt{Sig}(sk_s^{ap_i}, m)\rangle, \quad m = \langle \texttt{ReqID}, \texttt{Requester}, \texttt{Sender}, \texttt{ENList}, \texttt{SenderAdjList}, \texttt{SenderTime}\rangle$

3: ap_0 receives a response Res, and does the following:
   i. Extracts Edge from Res. If it has not already received a Res containing same Edge, updates the element in $D'_{N_{ap_0}}$ indexed by APs in the Edge.
   iii. If all the elements of $D'_{N_{ap_0}}$ (excluding the ones that were initially set as 0) are updated, or a predefined waiting period has passed (in which case the elements that were not updated are set as 0), ap_0 outputs $LocIntInfo(ap_0) = \langle EN_{ap_0}, D'_{EN_{ap_0}}, t_{int}\rangle$, t_{int} is the timestamp of the algorithm completion, at ap_0.

Fig. 7. GenLocIntInfo: Location integrity information generation algorithm

Input: $LocIntInfo(ap_0)$, $LocMap$, geo-tampering detection threshold $\Delta = (\Delta_M, \Delta_R)$. Δ_M: distance measurement error, Δ_R: distance recovery error
Output: $Out = \{0, 1\}$

1: Extract the distance matrix $D'_{EN_{ap_0}}$ and effective neighborhood EN_{ap_0} from $LocIntInfo(ap_0)$.
2: If $D'_{EN_{ap_0}}$ has missing elements, let $\mathcal{M}$ be the set of missing elements. Apply the *Alternating Descent* on $D'_{EN_{ap_0}}$ to recover distances in $\mathcal{M}$.
3: Use $LocMap$ and EN_{ap_0} to compute the distance matrix $D_{EN_{ap_0}}$.
4: For each $D_{N_{ap_0}}(i,j) \in \mathcal{M}$, check if $|D_{N_{ap_0}}(i,j) - D'_{N_{ap_0}}(i,j)| \leq \Delta_R$.
5: For each $D_{N_{ap_0}}(i,j) \notin \mathcal{M}$, check if $|D_{N_{ap_0}}(i,j) - D'_{N_{ap_0}}(i,j)| \leq \Delta_M$.
6: If all the checks succeed, $Out \leftarrow 1$, otherwise $Out \leftarrow 0$
7: **return** Out

Fig. 8. VerLocIntInfo: Location integrity information verification algorithm.

corresponding distances in $D_{EN_{ap_0}}$ and $D'_{EN_{ap_0}}$, and detect a change if the difference is larger than geo-tampering detection threshold Δ. As noted earlier, this value is the tuple of EDM recovery error, and the distance measurement error.

Theorem 4 states that location integrity data satisfies P2.

Theorem 4 (Sufficiency of $LocIntInfo(ap_0)$). Assuming *Alternating Descent* can complete EDM with high success probability, and at least three untampered node are present in the effective neighborhood of ap_0, $LocIntInfo(ap_0)$ generated by algorithm GenLocIntInfo satisfies P2 with high probability: convinces the POL system that the geo-location of ap_0 with respect to $LocMap$ is correct.

Proof is in the supplemental material, Appendix C.

4.4 Putting it together: Constructing POLA$^+$

This section puts all the parts together and extends our proposed proof-of-location scheme POLA to construct POLA$^+$ that provides security against geo-tampering attack.

Initialization (POLInit). As in POLA in Section 3.

User registration (POLInit). As in POLA in Section 3.

Proof-of-location generation (POLGen). *Access point discovery and initial handshake.* As in POLA in Section 3.

*Distance bounding between u and ap_0 (*POLGen.DB*).* As in POLA in Section 3.

Generating location integrity data for ap_0. If POLGen.DB returns 1, ap_0 begins GenLocIntInfo algorithm (see Fig. 7) and generates $LocIntInfo(ap_0)$.

*Issuing proof-of-location (*POLGen.Issue*).* This is as in POLA, but with $LocIntInfo(ap_0)$ included in the issued proof-of-location pol.

$$pol \leftarrow \langle sig, msg\rangle, sig \leftarrow \mathbb{DS}.\texttt{Sig}(sk_s^{ap_0}, msg)$$
$$msg = \langle com, pk_s^{ap_0}, \ell oc_{ap_0}, t, LocIntInfo(ap_0)\rangle$$

$$LocIntInfo(ap_0) = \langle EN_{ap_0}, D'_{EN_{ap_0}}, t_{int} \rangle$$

Proof-of-location verification (`POLVer`). This is as in POLA, with following additional checking by the verifier:

- Extracts $LocIntInfo(ap_0)$ from pol, retrieves $LocMap$ from $DBase$.
- Extracts timestamps t from msg and t_{int} from $LocIntInfo(ap_0)$. Checks that t and t_{int} are at most a predefined value Γ apart from each other.
- Runs algorithm VerLocIntInfo (Fig. 8).
- If the algorithm outputs 0, verifier rejects the claim and aborts, otherwise accepts the claim.

Setting a small Γ ensures that $LocIntInfo(ap_0)$ is generated immediately before pol is issued to the user, and so P1 is satisfied. P2 is also satisfied as proved in theorem 4. Now we can derive following corollary from theorem 2 and 4.

Corollary 1. POLA$^+$ is secure against *geo-tampering*, while preserving POL security properties (property 1, 2 and 3).

5 Experiments and Evaluation

5.1 Proof-of-concept implementation of POLA

Our implementation goal is to estimate the processing time of user, issuer, and verifier. For all these we use a mobile phone (Samsung Galaxy S9) to represent the resource limitations of the user (requester) as well as the small portable base stations that represent the issuer and the verifier.

We use the Java implementation version v2.3.43 of Idemix (Identity Mixer) [41], a cryptographic protocol suite that is designed for providing anonymity for authentication, and unlinkability for transactions, using CL signature and hash functions. Idemix uses SHA-256 hash function. Idemix commitment scheme is based on the hardness of discrete logarithm (DL) problem, while the CL signature security relies on the hardness of factorization problem. To show the effect of key size (security level) on the computation time and storage, we consider three different RSA modulus sizes for the CL-signature, 1024 bits, 1536 bits and 2048 bits. For these sizes, Idemix uses appropriate group sizes for the DL problem so that the overall cryptographic security will be equivalent to that of the RSA modulus size. More details are in the Idemix specification document given as Table 2 and 3 in [42]. More specifically, Idemix uses 768 bit commitment modulus with 1024 bit RSA modulus, and 1632 bits commitment modulus with both 1536 and 2048 bit RSA modulus.

We examine the computational time and storage that are needed to run POLA with our implementation. The results that are shown in Figure 9 are based on 10 independent runs of each test. During the tests, we ensured that no other background processes were running in parallel.

In our testbed, the user, who already possesses a certificate from TA on its secret key (a CL-signature from TA on user's secret), requests for a pol to an issuer that contains a commitment on user's secret and a non-interactive zero-knowledge proof stating that user holds a valid certificate from the TA on the committed value. Issuer decides if user is within an acceptable distance and validates the zero-knowledge proof. This is the *initialization phase* in Figure 9. After the initialization phase the user and the issuer take part in generating the pol credential, which is essentially the issuer's CL-signature on user's commitment, issuer's location and time. This is *pol generation* phase in Figure 9. Finally, in *pol verification* phase, the verifier and user takes part in the verification of *pol* which is a non-interactive zero-knowledge proof allowing verifier to validate pol without revealing user's identity.

Initialization phase, that corresponds to user making commitment on its secret and generating zero-knowledge-proof, and issuer verifying the proof, takes $24.7, 36.3$ and 55.6 milliseconds for user, and $16.5, 24.4$ and 38.7 for issuer (for RSA modulus size of 1024, 1536 and 2048 bits, respectively). RSA modulus size comes into play when prover generates zero-knowledge proof of knowledge on its certificate (CL-signature) from TA, and when issuer verifies that proof, and thus affecting the computation time.

pol generation phase is most costly (amount of time) phase in all cases, that generates CL-signature on user attributes (location, time) and requires the user and the issuer to perform a protocol consisting of multiple rounds (see specification of Idemix [42], Section 6.1.1 for a description of the protocol). For three RSA modulus sizes (1024, 1536 and 2048 bits), the computation times for this phase are $170.4, 185.6$ and 242.9 ms for users, and $228.7, 234.7$ and 288.2 ms for the issuer. *pol* verification protocol at the user and verifier takes $21.8, 32, 51.1$ ms and $13.6, 22.3, 36.4$ ms, respectively.

pol size for three different RSA modulus lengths are $1840, 2087, 2391$ bytes, respectively, that are definitely acceptable considering the storage capacity of today's mobile devices.

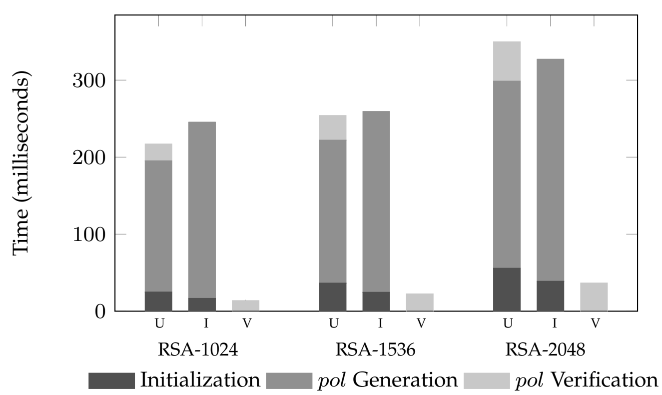


Fig. 9. Computation time of different phases of POLA for user (U), issuer (I) and verifier (V), with 1024 bits, 1536 bits and 2048 bits RSA modulus sizes for CL-signature.

5.2 *Geo-tampering* detection

The goal of this experiment is to analyze the geo-tampering detection threshold Δ (Def. 4) of our geo-tampering detection approach, and to evaluate the performance of our proposed algorithm to select effective neighborhood (Algorithm in Fig. 6) in terms of lowering this threshold. We also compare our geo-tampering detection algorithm with location determination (trilateration) based approach (see Sec. 4 for descriptions of both approaches).

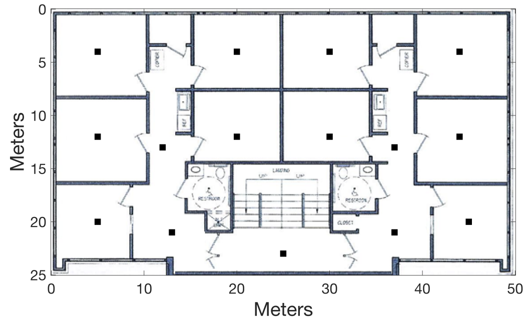


Fig. 10. Floor plan of an office building of 50×25 sq. meter area. A total of 15 access points (black squares) have been deployed at various locations in the building. In experiments, we ran each test for 15 times, to consider all fifteen possible locations as the location of the issuer AP. To ensure integrity of the experiments, we considered the worst possible (maximum) value for number of missing edges (Fig. 1) or geo-tampering threshold Δ (Fig. 11), among these 15 runs.

We noted earlier that geo-tampering detection threshold (Δ) is the tuple of EDM Recovery Error (Δ_R) and distance measurement error (Δ_M). However, in experiments, distance measurement error is a parameter that we "choose", and observe the EDM Recovery Error for the chosen values of Δ_M. Hence when we mention geo-tampering threshold (Δ) in this section, we mean only the EDM recovery error (Δ_R).

5.2.1 *Experiment Setup*

We considered an indoor office environment of 50×25 square meters dimension, that deploys fifteen access points at different locations (e.g., each of the 10 rooms has one AP in it, 5 more are deployed in the passageway, see Fig. 10). We used MATLAB to simulate this environment.

Coverage range instead of line-of-sight. In indoor environment, Line-Of-Sight (LOS) is hardly achievable due to walls and other obstacles. However, unavailability of LOS does not render TOA based distance measurement techniques (which is used by our approach) impractical in indoor environment. TOA based ranging between two access points in indoor environment is practical as long as the APs are within *Coverage range* of each other. Coverage range is a distance that is related to *maximum tolerable path loss of the direct path*. A direct path cannot be detected after the Coverage range due to high attenuation and scattering of signal in indoor environment [43]. As shown in [43], in typical indoor environments (such as office, laboratory, factory), coverage range is between 25 to 60 meters.

5.2.2 *Selecting effective neighborhood*

We implemented Algorithm in Fig. 6, and for comparison considered a direct (baseline) approach where, given an effective neighborhood size m, a total $(m-1)$ APs are selected arbitrarily (randomly) from the available APs in the neighborhood N_{ap_0}. ap_0 is included by default, and thus completing the effective neighborhood.

Comparison of direct and proposed approach. We compared the direct approach with the proposed approach in terms of number of missing edges in the resulting effective neighborhood EN_{ap_0}, and the comparison is shown in figure 1.

TABLE 1
Effective neighborhood selection: Direct Vs Proposed approach.
Proposed approach significantly reduces the number of missing edges in EN_{ap_0}. When proposed approach selects an EN_{ap_0} of size 8 for coverage range of 25 meters, there are only 4 missing edges out of total possible 28 edges. Direct approach gives an EN_{ap_0} with 12 missing edges, for the same setting.

			Coverage Range (m)				
			25	30	35	40	45
Effective neighborhood size	4	Proposed	2(6)	0(6)	0(6)	0(6)	0(6)
		Direct	4(6)	3(6)	2(6)	1(6)	0(6)
	8	Proposed	4(28)	1(28)	0(28)	0(28)	0(28)
		Direct	12(28)	10(28)	6(28)	2(28)	0(28)
	12	Proposed	17(66)	8(66)	3(66)	1(66)	0(66)
		Direct	28(66)	18(66)	11(66)	3(66)	0(66)

Missing edges (Total edges)

We fixed the distance measurement error to 2 meters. We ran the test 15 times, each time we selected a different AP, located at different location in the building, to play the role of ap_0, and took worst possible value, that is, the highest number of missing edges in EN_{ap_0} among this 15 runs. We considered effective neighborhood size of 4, 8 and 12, for Coverage range $25, 30, 35, 40, 45$ meters, for both approaches (See Table 1).

Table 1 shows that our proposed selection algorithm performs better than the direct one in terms of missing edges, for all three sizes of EN_{ap_0}. For instance, when the required effective neighborhood size is 8 and the coverage range is 25 meters, proposed approach outputs an effective neighborhood with only 4 missing edges out of possible 28 edges, while the direct approach outputs an effective neighborhood with 12 missing edges. Since EDM reconstruction error is directly dependent on the number of missing elements [44], the proposed selection approach significantly reduces geo-tampering threshold Δ.

5.2.3 *Trade-off between Δ and EN size.*

Figure 11 shows how the value of Δ changes when we vary: (i) Size of the effective neighborhood (4, 8 and 12), (ii) Coverage range of access points (ranging from 25 to 45 meters, in 1 meter interval), and (iii) distance measurement error (0.5 to 3 meters, in 0.5 meter interval). We consider the AP layout as in Fig. 10, (i.e., 15 APs in a 50×25 sq. meters office building). We used algorithm in Fig. 6 to select the effective neighborhood in all cases. As in the last experiment, we ran each test for all 15 possible location of ap_0, and took the maximum value of Δ.

Observe that for all sizes of effective neighborhood, when the coverage range gets bigger, the geo-tampering threshold Δ approaches zero. This is due to the fact that bigger coverage range results in EDMs with very small percentage of missing elements, and consequently smaller recovery error. Also notice that smaller the effective neighborhood size, faster the Δ becomes zero in terms of coverage range. Smaller sizes of EN are therefore more suitable for applications that have devices with comparatively weaker radio strength, and/or operates in more complex indoor scenarios where coverage ranges are affected by obstacles.

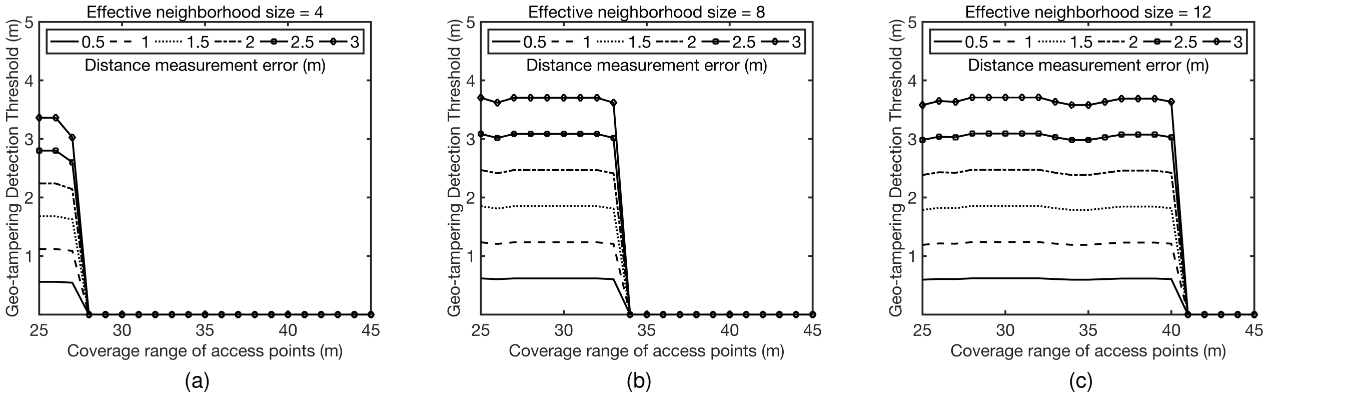


Fig. 11. Geo-tampering threshold (Δ) for varying coverage range (25 to 45 meters), distance measurement error (0.5 to 3 meters) and size of effective neighborhood (4, 8, 12). For all cases, after a certain value for coverage range of access points, Δ becomes zero. For example, in (a), this happens when coverage range increases from 27 to 28 meters. This jump to zero is due to (i) we chose the coverage range interval for our experiment as 1 meter, and (ii) number of missing edges in EN_{ap_0} is zero when coverage range is 28 meters in this case.

Approach	Scenario 1	2	3
Location determination	0.58	NA	NA
Proposed	0	2.1	2.2

Δ (m)

Fig. 12. **Comparison with location determination based approach.** Star represents the issuer AP, circles represent neighboring APs, and a solid line between two APs means that these are in coverage range of each other. Location determination (Trilateration) based approach can only detect geo-tampering in Scenario 1, where all 4 APs are in each other's coverage range. Our approach is applicable in all three scenarios.

On the other hand, bigger sizes of EN are more suitable for applications that operates in less secure or public area, such as shopping malls, where the possibility of geo-tampering is higher; so bigger neighborhood will increase the chance of having at least three untampered APs in it, and thus guaranteeing the detection of geo-tampering attack (see Theorem 4).

5.2.4 Comparison with location determination based detection.

Figure 12 compares our approach in detecting geo-tampering attack with the location determination (trilateration) based detection approach.

We consider three scenarios. In all scenarios, we consider the presence of only 4 access points in the office building. We assume that all other access points are absent (i.e., temporarily down or were not deployed in the first place). In scenario 1, there are three access points at the coverage range the issuer AP. In scenario 2, two access points are at the coverage range of the issuer AP, and the remaining one is two hops away. In scenario 3, only one access point is at the coverage range of the issuer AP, and rests are two and three hops away, respectively.

We fixed a distance measurement error of 2 meters, and tried to compute the geo-tampering detection threshold Δ in all scenarios, for both approaches. For location determination based detection approach, the trilateration error, which is the distance between actual and trilaterated location of issuer AP, is the amount that the AP needs to be moved for a successful geo-tampering detection, and according to the definition 4, this is the geo-tampering detection threshold.

Observe that trilateration based approach is only applicable in scenario 1, where all three APs are at the coverage range of the issuing AP, and detects a geo-tampering with $\Delta = 0.58$ meters. Our approach is applicable in all three scenarios, and and detects geo-tampering with $\Delta = 0, 2.1$ and 2.2 meters, respectively.

6 Related Work

Geo-tampering attack that is introduced in this paper is complementary to the traditional attacks in localization systems such as [45], [46] where the access points remain intact and the prover's goal is to claim a different location. In secure localization systems the prover intends to change the timing of the signals such that the verifying nodes have measurements that are constrained by their physical locations that are assumed correct. For example in [45] it is shown that the prover's restriction is that they can add a single delay to all distances and this will limit the places that are possible to claim. In our setting the locations of the access points are malleable and the change, if any, must be detected in real time and by performing new distance measurements and comparing them with some base measurements. Note that this attack is increasingly possible as the number of access points gets smaller and can be easily moved/relocated. Also note that tampering with location (moving) is a much easier attack compared to breaking into such nodes and modify their codes.

A number of proof-of-location systems have been proposed to date but none of them considers location integrity of the infrastructure.

Location proof system in Javali et al. [34] utilizes CSI (Channel State Information) data extracted from packets sent by users to access points to determine their location, and a fuzzy vault technique to preserve user location privacy. However, this is vulnerable to relay attack that enables a user obtaining proof for locations that they have not been. VeriPlace [3] is another proof-of-location system that employs three different trusted third parties to provide proof-of-location to the users. However, user location verification capability of this scheme is significantly limited since the scheme can only detect location cheating when the two locations claimed by a user in consecutive proof-of-location requests are impossible in space-time domain for that individual. APPLAUS [2] is a proof-of-location system where bluetooth enabled users mutually generate proof-of-location that are uploaded on a server, that can later by queried by a *verifier* to validate a user's location claim. Periodically changing pseudonyms are used to provide user anonymity, that can introduce high storage and management overhead for the Certificate Authority. To achieve user anonymity with respect to the issuer, Nosouhi et al. [47] use a modified version of DB in [27] and encrypt the user ID to achieve POL security and privacy. They however do not provide user anonymity with respect to the *pol* verifier. Also, *pol* generation in the scheme is verifier specific; that is a user must use the public key of the verifier to encrypt their identity during the *pol* generation phase to preserve anonymity against issuer, resulting in a *pol* that can be used only by the target verifier.

Our model of POL system assumes existence of a trusted location infrastructure consisting of APs with correct locations and honest behaviour. In [2], [16], [17], [47], [48] however the *pol* is constructed by untrusted "witnesses", each consisting of a neighbouring device. These works, however, do not cryptographically model and prove security. Rather they use informal security arguments such as "trust ranking" [2], [16], [17] or incentive and penalty models [47], [48] to show security of the system.

7 Concluding Remarks

We provided a formal model and security definition for proof-of-location systems that provide privacy for the user against *pol issuer* and *verifier*, and proposed a proof-of-location system with provable properties in our model. We introduced a novel attack that targets the physical integrity of the proof issuing infrastructure, and results in forged *pol*. We showed how EDM can be effectively used to provide provable protection against this attack. Using EDM for infrastructure integrity is a novel approach and can have applications to other location systems. We also implemented our proof-of-location scheme on android smart-phones to observe computational time and storage requirement. Optimal placement of access points given the physical restrictions of indoor environments to naturally protect against geo-tampering is an interesting research direction.

Acknowledgment

The work of Mamunur Akand and Reihaneh Safavi-Naini was in part supported by Natural Sciences and Engineering Research Council of Canada and Telus Communications Industrial Research Chair Program.

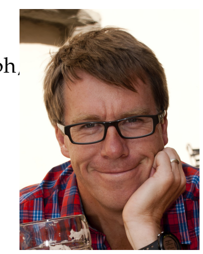
References

[1] B. Waters *et al.*, "Secure, Private Proofs of Location," Princeton University, Technical, 2002.
[2] Z. Zhu *et al.*, "APPLAUS: A Privacy-Preserving Location Proof Updating System for location-based services," in *2011 Proceedings IEEE INFOCOM*, 2011, pp. 1889–1897.
[3] W. Luo *et al.*, "VeriPlace: A Privacy-aware Location Proof Architecture," in *Proc. of the 18th SIGSPATIAL*, ser. GIS '10. New York, NY, USA: ACM, 2010, pp. 23–32.
[4] C. Ardagna *et al.*, "Privacy-enhanced location-based access control," in *Handbook of Database Security*. Springer, 2008, pp. 531–552.
[5] I. Ray, M. Kumar, and L. Yu, "Lrbac: a location-aware role-based access control model," in *International Conference on Information Systems Security*. Springer, 2006, pp. 147–161.
[6] R. Khan *et al.*, "A secure location proof generation scheme for supply chain integrity preservation," in *IEEE HST'13*, vol. 13, 2013, pp. 446–450.
[7] C. Bonebrake *et al.*, "Attacks on gps time reliability," *IEEE Security & Privacy*, vol. 12, no. 3, pp. 82–84, 2014.
[8] T. Nighswander *et al.*, "Gps software attacks," in *Proc.s of the 2012 ACM CCS*. ACM, 2012, pp. 450–461.
[9] S. Brands *et al.*, "Distance-bounding protocols," in *Workshop on the Theory and Application of of Cryptographic Techniques*. Springer, 1993, pp. 344–359.
[10] I. Boureanu *et al.*, "Secure and lightweight distance-bounding," in *International Workshop on LightSec*. Springer, 2013, pp. 97–113.
[11] G. Hancke *et al.*, "An rfid distance bounding protocol," in *Proc. of the 2005 SecureComm*. IEEE, 2005, pp. 67–73.
[12] R. Vaghefiand *et al.*, "Rss-based sensor localization with unknown transmit power," in *Proc. of 2011 ICASSP*. IEEE, 2011, pp. 2480–2483.
[13] E. Xu *et al.*, "Source localization in wireless sensor networks from signal time-of-arrival measurements," *IEEE Transactions on Signal Processing*, vol. 59, no. 6, pp. 2887–2897, 2011.
[14] C. Zhang *et al.*, "Accurate uwb indoor localization system utilizing time difference of arrival approach," in *2006 IEEE RWS*. IEEE, 2006, pp. 515–518.
[15] A. Ranganathan *et al.*, "Are we really close? verifying proximity in wireless systems," *IEEE S&P*, 2017.
[16] X. Wang *et al.*, "STAMP: Ad hoc spatial-temporal provenance assurance for mobile users," in *Proc. of 21st ICNP)*, 2013, pp. 1–10.
[17] S. Gambs *et al.*, "PROPS: A PRivacy-Preserving Location Proof System," in *Proc. of the 33rd SRDS*, 2014, pp. 1–10.
[18] M. Graham *et al.*, "Protecting Privacy and Securing the Gathering of Location Proofs – The Secure Location Verification Proof Gathering Protocol," in *MobiSec*. Springer, 2009, pp. 160–171.
[19] L. Bussard *et al.*, "Distance-bounding proof of knowledge to avoid real-time attacks," in *IFIP SEC*. Springer, 2005, pp. 223–238.
[20] A. Bay *et al.*, "The bussard-bagga and other distance-bounding protocols under attacks," in *Proc. of the 2012 ICISC*. Springer, 2012, pp. 371–391.
[21] J. Camenisch and A. Lysyanskaya, "An efficient system for non-transferable anonymous credentials with optional anonymity revocation," in *International Conference on the Theory and Applications of Cryptographic Techniques*. Springer, 2001, pp. 93–118.
[22] J. Camenisch *et al.*, "Efficient attributes for anonymous credentials," in *Proc. of the 15th ACM CCS*. ACM, 2008, pp. 345–356.
[23] G. Persiano and I. Visconti, "An efficient and usable multi-show non-transferable anonymous credential system," in *International Conference on Financial Cryptography*. Springer, 2004, pp. 196–211.
[24] A. Ahmadi *et al.*, "New attacks and secure design for anonymous distance-bounding," in *Proc. of the 2018 ACISP*. Springer, 2018, pp. 598–616.
[25] X. Bultel *et al.*, "A prover-anonymous and terrorist-fraud resistant distance-bounding protocol," in *Proc. of the 9th ACM WiSec*. ACM, 2016, pp. 121–133.

[26] S. Gambs *et al.*, "Prover anonymous and deniable distance-bounding authentication," in *Proc. of the 9th ASIACCS*. ACM, 2014, pp. 501–506.
[27] G. Avoine *et al.*, "A terrorist-fraud resistant and extractor-free anonymous distance-bounding protocol," in *ASIACCS'17*. ACM, 2017, pp. 800–814.
[28] I. Damagard and E. Fujisaki, "An integer commitment scheme based on groups with hidden order," *IACR Cryptology ePrint Archive 2001/064*, 2001.
[29] J. Camenisch *et al.*, "Efficient group signature schemes for large groups," in *Annual International Cryptology Conference*. Springer, 1997, pp. 410–424.
[30] J. Camenisch and A. Lysyanskaya, "A signature scheme with efficient protocols," in *International Conference on Security in Communication Networks*. Springer, 2002, pp. 268–289.
[31] C. Rackoff *et al.*, "Non-interactive zero-knowledge proof of knowledge and chosen ciphertext attack," in *CRYPTO'91*. Springer, 1991, pp. 433–444.
[32] J. Bacon and K. Moody, "Time in distributed systems," https://www.cl.cam.ac.uk/teaching/0910/ConcDistS/10a-Time.pdf, (Accessed on 06/26/2020).
[33] B. Fisch *et al.*, "Physical zero-knowledge proofs of physical properties," in *Annual Cryptology Conference*. Springer, 2014, pp. 313–336.
[34] C. Javali *et al.*, "I Am Alice, I Was in Wonderland: Secure Location Proof Generation and Verification Protocol," in *Proc. of the 2016 LCN*, 2016, pp. 477–485.
[35] A. Norrdine, "An algebraic solution to the multilateration problem," in *Proc. of the 15th IPIN*, vol. 1315, 2012.
[36] I. Dokmanic *et al.*, "Euclidean distance matrices: A short walk through theory, algorithms and applications," *CoRR*, vol. abs/1502.07541, 2015.
[37] M. Youssef *et al.*, "Pinpoint: An asynchronous time-based location determination system," in *Proc. of the 4th SIGMOBILE*. ACM, 2006, pp. 165–176.
[38] R. M. Karp, "Reducibility among combinatorial problems," in *Complexity of computer computations*. Springer, 1972, pp. 85–103.
[39] "Find if there is a path between two vertices in an undirected graph," https://www.geeksforgeeks.org/find-if-there-is-a-path-between-two-vertices-in-an-undirected-graph, (Accessed on 08/20/2020).
[40] H. Lim and C. Kim, "Flooding in wireless ad hoc networks," *Computer Communications*, vol. 24, no. 3-4, pp. 353–363, 2001.
[41] "Msp implementation with identity mixer," https://hyperledger-fabric.readthedocs.io/en/release-1.3/idemix.html\#what-is-idemix, (Accessed on 08/24/2020).
[42] J. Camenisch *et al.*, "Specification of the identity mixer cryptographic library," *IBM Research—Zurich*, pp. 1–48, 2010.
[43] N. A. Alsindi *et al.*, "Measurement and modeling of ultrawideband toa-based ranging in indoor multipath environments," *IEEE Transactions on Vehicular Technology*, vol. 58, no. 3, pp. 1046–1058, 2008.
[44] R. Parhizkar, "Euclidean distance matrices: Properties, algorithms and applications," Ph.D. dissertation, Ph. D. dissertation, Ecole Polytechnique Federale de Lausanne (EPFL), 2013.
[45] V. Shmatikov *et al.*, "Secure verification of location claims with simultaneous distance modification," in *ASIAN*. Springer, 2007, pp. 181–195.
[46] J. Chiang *et al.*, "Secure and precise location verification using distance bounding and simultaneous multilateration," in *ACM WiSec'09*. ACM, 2009, pp. 181–192.
[47] M. R. Nosouhi *et al.*, "Pasport: A secure and private location proof generation and verification framework," *IEEE Transactions on Computational Social Systems*, vol. 7, no. 2, pp. 293–307, 2020.
[48] M. Amoretti *et al.*, "Blockchain-based proof of location," in *2018 IEEE International Conference on Software Quality, Reliability and Security Companion (QRS-C)*. IEEE, 2018, pp. 146–153.
[49] U. Dürholz *et al.*, "A formal approach to distance-bounding rfid protocols," in *ISW*. Springer, 2011, pp. 47–62.

Mamunur Akand is a Ph.D. candidate in the Department of Computer Science, University of Calgary, Canada, and Dr. Reihaneh Safavi-Naini is his current supervisor. He received his Bachelor and Master in Computer Science from Islamic University of Technology, Bangladesh and University of Calgary, Canada, respectively. Akand has been working with ISPIA Lab, University of Calgary since 2014. His research interests include cryptography, information security, and location-based security and privacy.

Reihaneh Safavi-Naini holds the NSERC/Telus Industrial Research Chair and Alberta Innovates Chair in Information Security. She is the (co-) Founding Director of Institute for Security, Privacy and Information Assurance at the University of Calgary, during 2009-2019, and currently the Director of Information Security Lab in the Department of Computer Science at the University of Calgary, Canada. Before joining University of Calgary in 2007, she was a Professor of Computer Science, and the Director of Telecommunication and Information Technology Research Institute (TITR) and the Centre for Information Security, all at the University of Wollongong, Australia. She has a Ph.D. in coding theory from University of Waterloo, Canada. Her current research interests include information-theoretic security, provable cryptography, network security, and security of distributed and decentralised systems.



Marc Kneppers received his Master in Astronomy from University of Western Ontario (Western University). He has 20 years of experience in IT/networking security and was appointed a TELUS Fellow and is now the Chief Security Architect for TELUS Communications. His responsibilities include security oversight and strategy across all of TELUS' technologies and portfolios. He represents TELUS on Canadian national infrastructure forums and industry boards with membership in international security forums and vendor advisory boards.

Matthieu Giraud received his Master in Cryptography in 2016 from Universite de Bordeaux, France, and Ph.D. in 2019 from Laboratory LIMOS of University Clermont Auvergne, France, on the security of data storage in the cloud. Currently he is working as an Engineer in the cryptography department of Thales Security and Communications (Gennevilliers, France). His research interests include cryptography, mathematics, and protection of privacy. He has worked on symmetric searchable encryption schemes, particularly on their leakage.

Pascal Lafourcade obtained his Ph.D. in 2006 from ENS Cachan (laboratory LSV) on verification of cryptographic protocols in presence of algebraic properties. He spent 1 year at the ETH Zurich in David Basin group, where he worked on WSN. Then, he was associate professor at Verimag during 7 years, where he developed automatic techniques for verifying cryptographic primitives and analyzed security protocols. Currently, he holds an industrial chair on Digital Trust in Laboratory LIMOS (Team Networks and Protocols) and is associate professor at University of Auvergne (Clermont-Ferrand, France).

SUPPLEMENTAL MATERIAL

APPENDIX A
CRYPTOGRAPHIC PRIMITIVES

Digital signature. A digital signature scheme ($\mathbb{DS}$) consists of three algorithms: a key generation ($\mathbb{DS}.\texttt{KeyGen}$), a signature generation ($\mathbb{DS}.\texttt{Sig}$) and a signature verification ($\mathbb{DS}.\texttt{Vf}$) algorithm. For a security parameter λ, the algorithms $\mathbb{DS}.\texttt{KeyGen}(1^\lambda) \rightarrow (pk_s, sk_s)$, $\mathbb{DS}.\texttt{Sig}\ (sk_s, m) \rightarrow \sigma$, $\mathbb{DS}.\texttt{Vf}(pk_s, m, \sigma) \rightarrow 0/1$, generate a private signing a public verifying key, digitally signs a message using signing key, and verifies a signature using public verification key. The correctness requires that for every pair of keys output by $\mathbb{DS}.\texttt{KeyGen}$, for every message m signed by $\mathbb{DS}.\texttt{Sig}$, $\mathbb{DS}.\texttt{Vf}$ outputs 1. We consider the standard notion of security against existential forgery, where the attacker needs to find a signature σ for at least one new message m, such that σ is valid for m with respect to a given verifying key [**?**]. In this paper, we use Camenisch and Lysianskaya's digital signature scheme (CL-signature scheme) [30].

Encryption. A public key encryption scheme ($\mathbb{E}$) provides three algorithms, a key generation ($\mathbb{E}.\texttt{KeyGen}$), an encryption ($\mathbb{E}.\texttt{Enc}$) and a decryption ($\mathbb{E}.\texttt{Dec}$) algorithms. For a security parameter λ, the algorithms $\mathbb{E}.\texttt{KeyGen}(1^\lambda) \rightarrow (pk_e, sk_e)$, $\mathbb{E}.\texttt{Enc}(pk_e, m) \rightarrow C$, $\mathbb{E}.\texttt{Dec}(sk_e, C) \rightarrow m$, generate a pair of public and private key, encrypt a message using the public key, and decrypt a ciphertext using the private key. The correctness requires that for every pair of key generated by $\mathbb{E}.\texttt{KeyGen}$, the decryption ($\mathbb{E}.\texttt{Dec}$) of any message m encrypted by $\mathbb{E}.\texttt{Enc}$, returns the original plaintext m. We consider the standard notion of indistinguishability under chosen ciphertext (IND-CCA) attack [31].

APPENDIX B
DISTANCE BOUNDING PROTOCOLS

A DB protocol is a two-party protocol between a prover (user) and a verifier (AP), that uses round trip time of signals to provide guarantee for the verifier that the prover is within its distance bound B. DB protocols are cryptographic protocols and enjoy well defined security models and proofs. Three common attacks are defined in DB protocol. Our security model and attack description follows [27].

Distance Fraud. In distance fraud attack, attacker is the far-away dishonest user u' who attempts to convince the access point ap_0 that they are within distance bound B. In a more general attack called *distance hijacking*, the far-away malicious user u' may use the presence of a close-by honest prover u to accomplish this.

Mafia Fraud. In Mafia Fraud attack, attacker is a close-by external entity who communicates with a far-away honest user u and the access point ap_0, to convince ap_0 that u is within distance bound B.

Terrorist Fraud. In terrorist fraud attack, the attacker has the same goal as in the mafia-fraud attack, but here the user u' is dishonest that colludes with the attacker up to the non-disclosure of significant information, such as, whole or part of its secret keys, that may enable the attacker in later impersonations of u.

APPENDIX C
PROOFS

Proof of Theorem 1. The proof has four parts. First we prove that if $\mathbb{E}$ is an IND-CCA secure encryption scheme and $\mathbb{C}$ is a computationally binding commitment scheme, then the distance bounding protocol of POLA between a user u and an access point ap_0 ($\texttt{POLGen.DB}$ in Fig. 5), is secure against distance fraud, distance hijacking, mafia fraud and terrorist fraud attacks, as defined in Appendix B. Next, for $1.b$, using $1.a$ and the assumption that the digital signature scheme $\mathbb{DS}$ is secure against existential forgery, we show that POLA is unforgeable. We then in $1.c$ show that if the $\mathbb{ZKP_oK}$ is a sound zero-knowledge proof-of-knowledge protocol,then POLA provides non-transferability [1] Lastly, in $1.d$ we prove that if commitment scheme $\mathbb{C}$ is computationally hiding and $\mathbb{ZKP_oK}$ is a zero knowledge proof of knowledge (of the values $(s_u, \alpha, cert_u)$, then POLA is anonymous with respect to both access point and the verifier.

Proof of $1.a$*:*

Distance Fraud (DF) Resistance. Distance Hijacking (DH) security, implies DF security because a DH attackers has the same capability of a DF attacker with an additional ability for the adversary: the far-away dishonest user u' can use the presence of a close-by honest prover u to convince the access point ap_0 that they are within distance bound B (See DB security model in [27]). Therefore, it is sufficient to prove Distance Hijacking (DH) Resistance of the DB protocol.

Distance Hijacking (DH) Resistance.

Firstly, consider the case where the close-by honest user u sends the initial message C_u to ap_0. This makes the authentication to fail for dishonest user u', since $com = \mathbb{C}.\texttt{Commit}(s_u, \alpha)$ does not correspond to the identity of u'. Therefore, consider that u' sent the initial message C'_u that is associated with com', β'. Let (α, β) be the values picked by honest user u, and com be the commitment generated using α. Now, for each challenge c_i, u' either lets u to respond, or they respond themselves using (com', β'). In the former case, the response is correct with probability $1/2$, since this is the probability to have $com_i = com'_i$ (or, $\beta_i = \beta'_i$). In the latter case, either $com'_i = \beta'_i \oplus \gamma_i$, and u' can commit to a correct response with probability 1, or $com'_i \neq \beta'_i \oplus \gamma_i$, and they must guess the challenge to commit to the correct response. Because γ is uniformly distributed and unknown to u' at the time they pick (α, β) and generates com using α, $Pr[com'_i = \beta'_i \oplus \gamma_i] = 1/2$. Therefore, the probability to commit correct response is $Pr[com'_i = \beta'_i \oplus \gamma_i].1 + Pr[Pr[com'_i \neq \beta'_i \oplus \gamma_i].1/2 = 3/4$.

Therefore, u''s best strategy is to respond themselves. For λ challenges u''s probability to succeed is at most $(3/4)^\lambda$, which is negligible.

Mafia Fraud Resistance. During Mafia Fraud attack the adversary acts as a man-in-the-middle between the honest far-away user u and the access point ap_0 and attempts to authenticate by interacting with both. However, there must be a restriction on adversary in relaying information between u and ap_0 during time-critical (distance bounding) phase. Otherwise, the attack would be successful by simply

1. Assuming the user will not share their secret credential

relaying the challenge from ap_0 to u and the corresponding response from u to ap_0. To discard such attacks, a "tainted time-critical" phase is introduced in [27]: in the time-critical (distance bounding) phase, the attacker's response to the i-th challenge from the ap_0, must be independent of the far-away honest user's response to that particular challenge.

Firstly we prove that the initialization phase of our proposed DB protocol does not leak any information about the response table, that consists of binary strings com and β . Then we show that the attacker's best strategy is to guess the challenge/response in the time critical phase, which makes the success probability of MF attack negligible.

We start by arguing that the probability of outputting same value of $\langle com, \beta \rangle$ in more than one protocol session, is negligible. Recall that α, β are generated randomly at each session, and com is the commitment randomized by α on prover's secret s_u. Therefore, in i-th session, the probability to have a collision with any of the previous $i-1$ $\langle com, \beta \rangle$ values is bounded by $\frac{i}{2^{2n}}$, which is negligible.

Next we rule out the possibility that the attacker generates a valid commitment value com without the key s_u, which is trivially forbidden by the "binding" property of the commitment scheme.

Then, the IND-CCA property of the encryption scheme $\mathbb{E}$ ensures that there is no leakage of $\langle com, \beta \rangle$ from C_u that is sent to the access point. Therefore, we can say that $\langle com, \beta \rangle$ only appears during the DB phase.

Now, in the DB phase, the attacker has no way to predict the response value r_i for any round i, since neither of com_i or β_i appears before round i. Given the restriction we stated at the start of this proof, the attacker can only try to guess the challenge c_i or response r_i. They succeed with probability $\frac{1}{2}$ in the second case. They succeed in the first case upon guessing the challenge correctly, since only then they can obtain the response from the prover. Attacker also succeeds if they wrongly guess the challenge but receives correct response (in case r_i is same for both challenges). For each round, the success probability is $\frac{1}{2} \cdot 1 + \frac{1}{2} \cdot \frac{1}{2} = \frac{3}{4}$. For λ such rounds, the probability becomes $(3/4)^{\lambda}$, which is negligible.

Terrorist Fraud Resistance.

We follow the simulation-based TF-resistance notion defined by Dürholz et al. [49], where a far-away dishonest user u' attempts to authenticate to the DB verifier (access point) with the help of an *accomplice* that is close to the verifier. The assumption is, no rational user u' would leak enough information to their accomplice during the TF attack that will enable the accomplice in authenticating himself to the verifier without the help of u' at a later stage. Let Pr_A be the success probability for the malicious user u' (with the help of accomplice A) in the first stage, and let a simulator S take the internal view of accomplice A and run an authentication attempt without any interaction with any user. Let Pr_S be the success probability of the simulator. According to simulation based TF-resistance model, a protocol is TF attack resistant if $Pr_S \geq Pr_A$. Similar the restriction we stated for MF attack, the dishonest user u' is not allowed to communicate with the accomplice during the time-critical phase of the protocol.

Now we will show that if the verifier's challenges (c_i) are picked uniformly at random, then our proposed DB protocol is terrorist fraud resistant in the simulation-based definition. In particular, we will prove that for any user u' helping an accomplice A to authenticate in a DB session (let ses_A denote this session) with probability of success Pr_A, there exists a simulator S that can authenticate itself in a later DB session (let ses_S denote this session) using A as a black-box, with a success probability equal or greater than Pr_A.

The idea is to put A in the same situation as when it was helping u' to authenticate. The simulator simulates the malicious user u' as well as the DB verifier (access point) for A. Firstly, A is rewind to the state when it already received information from u' and sent C'_u to the verifier in session ses_A. Simulator relays this C'_u to its own verifier (to whom simulator is trying to authenticate) and receives γ'. However, simulator sends the message γ to A that was actually sent to A in session ses_A, even if $\gamma' \neq \gamma$. This is done to simulate session ses_A for A.

Afterwards, in the time critical session, simulator forwards the challenges from the verifier of ses_S to A. Simulator simply forwards the response r_i from A to the verifier when $c_i = 0$. When $c_i = 1$, simulator sends $r'_i = r_i \oplus \gamma_i \oplus \gamma'_i$.

It is evident that the simulator can respond to any challenge from the verifier in session ses_S with the success probability at least equal to the success probability of the accomplice and user u' in session ses_A. That is, we prove that $Pr_S \geq Pr_A$.

Proof of $1.b$*:*

Winning conditions for the adversary to succeed in POL forgery is given in property 1. Let us assume that advantage of the adversary, Adv_{UF} is non-negligible. We can have two cases.

Case 1: Winning condition 1 is fulfilled. That is, given that there exists an entry $(pol, .) \in VerList$, s.t., $pol = pol_A$, we have the following:

i) There does not exist an entry $(pol, .) \in GenList$ s.t. $pol = pol_A$, and ii) there does not exist an entry $(pol, .) \in IssueList$ s.t. $pol = pol_A$.

Therefore, pol_A must not have been issued by any access point $ap_0 \in \mathcal{AP}$.

Property 1 restricts adversary from corrupting any access point, therefore adversary does not possess the secret signing key of any access point.

The only way adversary could generate the proof-of-location pol_A that is verified by the *verifier*, is that adversary forged the signature of an access point. However, $\mathbb{DS}$ is secure against existential forgery. Therefore, probability of adversary fulfilling winning condition 1 is negligible.

Case 2: Winning condition 2 is fulfilled. That is, given that there exists an entry $(pol, .) \in VerList$, s.t., $pol = pol_A$, we have the following:

i) There exists an entry $(pol, .) \in GenList$ s.t. $pol = pol_A$, and ii) $d(\ell oc_{ap_0}, \ell oc_u) > B$.

Therefore, a query to $POLGen(ap_0, u)$ oracle was made by the adversary with u located far-away from ap_0. However, the proof-of-location $pol = pol_A$ was only issued after `POLGen.DB` was successful. That is, ap_0 was convinced that u was located nearby. The only way this is possible is by breaking the security of distance bounding protocol, which is negligible by definition. Therefore, probability of adversary fulfilling winning condition 2 is also negligible.

Proof of 1.*c:*

Winning conditions for the adversary to succeed in transferring a *pol* is given in property 2. Let us assume that advantage of the adversary, Adv_{NT} is non-negligible. We can have two cases.

Case 1: Winning condition 1 is fulfilled. That is, given that there exists an entry $(pol, .) \in VerList$, s.t., $pol = pol_A$, we have the following:

i) There exists an entry $(pol, u') \in GenList$ s.t. $pol = pol_A$, and ii) $u' \neq u$.

We have assumed that long term secret are not shared among users, therefore, we rule out the possibility that u' colluded with u and shared their secret $s_{u'}$ with u.

Therefore, u takes part in the protocol `POLVer` without the secret $s_{u'}$. pol_A is presented to the *verifier*, which embeds a commitment done by u', namely $com' = \mathbb{C}.Commit(s_{u'}, r')$. A zero-knowledge proof of knowledge between v and u takes place, and the *verifier* is convinced that u knows the secrets $s_{u'}, r'$ used to make the commitment com'. Therefore, we can use this adversary to break the soundness of zero knowledge proof of knowledge protocol.

Case 2: Winning condition 2 is fulfilled. That is, given that there exists an entry $(pol, .) \in VerList$, s.t., $pol = pol_A$, we have the following:

i) There exists an entry $(pol, u') \in IssueList$ s.t. $pol = pol_A$, and ii) $u' \neq u$.

The argument is same as in the previous case.

Proof of 1.*d:*

Anonymity w.r.t access point: In this case, the access point ap_0 responsible for issuing the proof-of-location is controlled by adversary A. At the end of the POL game for anonymity w.r.t to access point (see property 3), A's view will consist of the output (i.e., either $\perp$ or *pol*) of `POLGen` protocol, as well as the transcript of the protocol, i.e., adversary will have access to following values: $com, \beta, sKey, r_i (1 \leq i \leq \lambda), View_A^{\mathbb{ZKPoK}}$, the last one representing the view of the adversary from the zero-knowledge proof of knowledge protocol that takes place in `POLGen.DB`'s verification phase.

Let us start with the output of the `POLGen` protocol. First of all, the output $\perp$ does not give away any information about the user other than that they are far away from the access point, which is already known by the adversary (i.e., adversary chooses the locations of the users). If the protocol outputs *pol*, then we should note that *pol* is ap_0's signature on $\langle com, PK^{ap_0}, \ell^{ap_0}, t\rangle$. The only information of our interest is com, since this is generated by the user. com is user's commitment on its secret s_u, $com = \mathbb{C}.Commit(s_u, \alpha)$, α is a binary string randomly generated by u. Now, u is honest in this game and follows the protocol, and the commitment scheme is statistically hiding. Therefore, A cannot gain any information on secret s_u from com.

$\beta, sKey$ are chosen randomly by the user, and so doesn't provide any information on u to A. r_i is response bit in round i, that is based on com and β.

What remains is $View_A^{\mathbb{ZKPoK}}$. The zero-knowledge property of the protocol $\mathbb{ZKPoK}$ is proven in [22], which ensures that the adversary does not get any any knowledge other than the fact that the commitment com is formed properly, i.e., user knows a secret s_u that they have committed to, and s_u is certified by the trusted authority.

Therefore, adversary A does not gain any information that can aid him to win the anonymity game w.r.t access point with non-negligible probability.

Anonymity w.r.t verifier: In this case, the *verifier* v responsible for verifying the proof-of-location is controlled by adversary A. At the end of the POL game for anonymity w.r.t to *verifier* (see property 3), A's view will consist of the output (i.e., either 0 or 1) of `POLVer` protocol, as well as the transcript of the protocol, i.e., adversary will have access to following values: $com, PK^{ap_0}, \ell oc_{ap_0}, t, View_A^{\mathbb{ZKPoK}}$, the last one representing the view of the adversary from the zero-knowledge proof of knowledge protocol that takes place in `POLGen.Ver`.

The arguments are similar to the previous case. A does not get any information about the secret s_u from com as the commitment scheme is statistically hiding. $\ell oc_{ap_0}, t$ does not provide any information on the identity of u. The zero-knowledge property of the protocol $\mathbb{ZKPoK}$ ensures that adversary does not gain any useful information from $View_A^{\mathbb{ZKPoK}}$.

Therefore, adversary A does not gain any information that can aid him to win the anonymity game w.r.t *verifier* with non-negligible probability.

Proof of Theorem 2. If P1 and P2 are satisfied by $LocIntInfo(ap_0)$, then it is guaranteed that at the time the proof-of-location *pol* was generated, the *pol issuer* ap_0's actual location $\ell oc'_{ap_0}$ was close enough to its original location ℓoc_{ap_0} (in $LocMap$) such that location integrity was preserved. That is, circle $Circ(\ell oc_{ap_0}, B)$ and circle $Circ(\ell oc'_{ap_0}, B)$ are effectively the same circle given that B is the distance bound, and $d(\ell oc_{ap_0}, \ell oc'_{ap_0})$ is negligible. By negligible we mean that this is within an error tolerance value (which we later defined as geo-tampering threshold Δ). Therefore, it is ensured that there is no geo-tampering attack.

Inclusion of $LocIntInfo(ap_0)$ in pol preserves unforgeability of POLA. Let, $LocIntInfo(ap_0)$ is included in *pol* as shown in theorem 2. We show that if a *pol* forgery successful in this *extended* POL, one can use the attacker to successfully forge a *pol* in the original scheme POLA.

Let the *extended* POL is A'. At the end of the POL game, A' outputs pol'_A. It includes, along with everything else, $LocIntInfo(ap_0)$. All the original POL attacker A needs to is remove $LocIntInfo(ap_0)$ from pol'_A, and output the resulting pol_A to the challenger. All the checks in the verification oracle $POLVer$ will succeed, given that similar checks are supposed to succeed in the game with A'. Thus, a successful *pol* forgery is performed by A on original POL.

Inclusion of $LocIntInfo(ap_0)$ in pol preserves non-transferability of POLA. A similar reduction proof can be shown as above.

Inclusion of $LocIntInfo(ap_0)$ in pol preserves anonymity of POLA. This follows the proof of theorem 1(iv). In addition to $com, \beta, sKey, r_i (1 \leq i \leq \lambda), View_A^{\mathbb{ZKPoK}}$, adversary will have access to value $LocIntInfo(ap_0)$. However, $LocIntInfo(ap_0)$ is the location integrity information of access point ap_0, and generation of this information must not depend on the location or identity of any user, since users

Input: The incomplete EDM M, the observation matrix W, the dimension of coordinates d, the upper bound of iteration $MaxIter$.
Output: The completed EDM $\hat{M}$

```
while No Convergence or c < MaxIter do
    for i ∈ {1, ..., n} do
        for j ∈ {1, ..., d} do
            θ_{i,j} ← GetQuadricCoeffs(W, M, d)
            m̂_{i,j} ← arg min_x f(x; θ_{i,j})
        end for
    end for
    c = c + 1
end while
return M̂
```

Fig. 1. *Alternating Descent*: EDM completion algorithm given in [36].

are mobile, and untrusted. Therefore, $LocIntInfo(ap_0)$ does not contain such information of the users, and including it in pol preserves anonymity of user w.r.t access point and the verifier.

Proof of Theorem 4. The theorem assumes the following:

1) The event that after running the completion algorithm the real-time computed EDM $D'_{N_{ap_0}}$ has no missing elements, has a high probability.
2) There are at least three untampered APs in the neighborhood of ap_0. Let these be ap_0^1, ap_0^2, ap_0^3.

Above two assumption implies, $D'_{N_{ap_0}}$ will have elements with values representing distances $d'(ap_0, ap_0^1), d'(ap_0, ap_0^2)$ and $d'(ap_0, ap_0^3)$.

Since distances from three geo-locations determine the location of a target access point in 2D space, any tampering on the location of ap_0 will affect at least one of the distances among $d(ap_0, ap_0^1), d(ap_0, ap_0^2)$ and $d(ap_0, ap_0^3)$, which will be detected in line 4 or 5 of algorithm 8 ($VerLocInt$). Therefore, $LocIntInfo(ap_0)$ that contains $D'_{N_{ap_0}}$, satisfies P2 with high probability: convinces the POL system that the geo-location of ap_0 with respect to $LocMap$ is correct.

Appendix D
EDM Completion Algorithm

We provide the EDM completion algorithm given in [36]. Given the incomplete EDM M, the observation matrix W, the dimension of coordinates d, the upper bound of iteration $MaxIter$, this algorithm returns a completed EDM $\hat{M}$. For a complete understanding on the parameters and how the algorithm works, we refer readers to [36].

Summary of Changes

Below we provide responses to all the review comments, as well as the changes we made to the original manuscript. References we make here are listed at the end of this document.

Responses to Editor comments:

1. In the revised paper we discussed and compared our work with the reviewers' referred work [2], and provide reference to another recent work [3].

2. We move basic definitions of Encryption and Digital signature to the supplementary material (Appendix A). They can be removed if needed. We kept definitions and notations of zero-knowledge-proof for readability because several steps of the construction rely on them (e.g., final stage of the DB protocol).

3. We discuss motivation for using game-based security model for POL in the revised paper, on page 4 (after POL correctness). The main motivation in using game-based security model for POL system is to use existing game-based security modelling of distance bounding protocols, as well as being able to extend the security model and proof to include location tampering of APs, and integrity checking that includes physical properties of environment.

4. The focus of the paper is secure design of POL with provable security, and so in our implementation we only considered implementation of POLA that uses CL-signature, commitment scheme and zero-knowledge proof of knowledge cryptographic primitives. In the revised paper, we updated our proof-of-concept implementation of POLA to show the effect of key size on computation time and storage, and considered three RSA modulus sizes for the CL-signature, 1024 bits, 1536 bits and 2048 bits, and other system parameters are also adjusted according to the size recommendation and restrictions set by the Idemix Library Specification (i.e., the crypto library we used).

   Implementation of distance bounding on smart phone is a research topic of its own. To our knowledge, the only work that explores feasibility of DB on smart phones is [1]. In this implementation DB protocols run at the Hardware Abstraction Layer and Near field communication (NFC) that implies the distance of the user to the issuer to be 4cm, which is not meaningful for proof-of-location systems.

   Nosouhi et al [2] claims to have implemented the DB protocol on smartphone, but no details of the implementation are provided. The implementation results of key size vs DB execution time (Figure 7c in [2]) shows that the execution time of DB is in seconds, while this is expected to be in nanoseconds.

Response to Reviewer 1 comments:

1. Please see the response to Editor Comment 1.
   Second and third reference from this review comment are not directly related proof-of-location and so our work. The second reference (Zhou et al.) is an authentication scheme based on a secure outsourced computation scheme, and the third reference (Chen et al.) gives a framework for location-based services, where users can privately query for location-based services.

2. Please see the response to Editor Comment 2.

3. Consider a pol issuing AP. We require (Theorem 4) at least three APs in the "effective neighborhood" (Def. 5) of the AP to provide integrity check for its location. The three APs only need to have a connected path to the pol issuing AP (be reachable through possibly multiple hops). A POL verifier uses the integrity information that is generated by the neighbors of the issuing AP, and the initial map "$LocMap$" of the area to detect any change in the location of the issuing AP.
   In our security model, APs are considered honest in unforgeability and non-transferability games, and honest but curios in anonymity game. The physical attack only changes the physical location of APs but maintains the above security model.

4. In our security model users are dishonest with respect to their location claim, and this is modelled through different types of attacks (For attack descriptions see DB attacks in appendix B in supplemental material). They are also dishonest with respect to pol's unforgeability and non-transferability properties (Property 1 and 2), and try to forge a pol, or transfer a proof of location. The verifiers are considered dishonest with respect to anonymity property and linking a user identity to a pol. We allow the verifiers to deploy privacy attacks (see property 3) on users. Theorem 1 proves that our proposed POL is secure against all these attacks.

5. Our revised paper discusses and compare with recent works, including a paper suggested by the reviewer [2]. We have improved readability of the paper by removing basic definitions such as digital signature and encryption scheme.

Response to Reviewer 2 comments:

1. Please see response to Editor Comment 3.
   Role of the challenger and the oracle are explained in the revised paper, page 4.
   Definition 2 uses terms such as "queries", "oracle" and "challenger" that must be explained before. This is done in paragraph immediately before the definition.

2. Witness-based POL systems, as well as the referred work [2] are discussed in the related work section of the revised paper.

Response to Reviewer 3 Comments:

1. Please see the response to Editor Comment 4.

2. We require user anonymity w.r.t pol issuer and the pol verifier (see property 3, POL Anonymity, on page 5). Paper [2], reveals the user identity (ID_P) to the verifier and does not consider user anonymity w.r.t the pol verifier. We design a distance bounding protocol, that preserve user anonymity with respect to the pol issuer and the pol verifier, both. This is the first distance bounding protocol that achieves this.

3. We have followed this suggestion in the revised paper.

Reference:

[1] Gambs S, Lassance CE, Onete C. "The not-so-distant future: Distance-bounding protocols on smartphones." In 2015 International Conference on Smart Card Research and Advanced Applications. Springer, Cham, 2018, pp. 209-224.

[2] Nosouhi MR, Sood K, Yu S, Grobler M, Zhang J. "PASPORT: A Secure and Private Location Proof Generation and Verification Framework." In IEEE Transactions on Computational Social Systems. 2020 Jan 23;7(2):293-307.

[3] M. Amoretti, G. Brambilla, F. Medioli, and F. Zanichelli, "Blockchain-based proof of location." In 2018 IEEE International Conference on Software Quality, Reliability and Security Companion (QRS-C). IEEE, 2018, pp. 146–153.