# Location-Proof System based on Secure Multi-Party Computations

Aurélien Dupin[1,3], Jean-Marc Robert[2], and Christophe Bidan[3]

[1] Thales Communications & Security, Gennevilliers, France
[2] École de Technologie Supérieure, Montréal, Canada
[3] Centrale-Supélec, Rennes, France

**Abstract.** Location-based services are now quite popular. The variety of applications and their numerous users show it clearly. However, these applications rely on the persons' honesty to use their real location. If they are motivated to lie about their position, they can do so. A *location-proof system* allows a *prover* to obtain proofs from nearby *witnesses*, for being at a given location at a given time. Such a proof can be used to convince a *verifier* later on. Yet, provers and witnesses may want to keep their location and their identity private. In this paper, a solution is presented in which a *malicious adversary*, acting as a prover, cannot cheat on his position and remain undetected. It relies on *multi-party computations* and *group-signature schemes* to protect the private information of both the prover and the witnesses against any *semi-honest participant*.

## 1 Introduction

Location-based services are now ubiquitous, mostly through our vehicles and smartphones. These services generally rely on the persons' honesty to transmit their real location. Thus, they are limited to situations in which the persons do not have any motivation to lie. However, for some services such as electronic voting, location-based access control, and law enforcement investigation, this is not the case. These services must be based on a *location-proof system* that allows a participant, called *prover*, to obtain proofs from nearby participants, called *witnesses*, asserting that he is at a given location at a given time. Such a proof can be presented later on to convince a service provider, called *verifier*.

Any location-proof systems based on the interaction between a prover and his neighbors have some privacy issues. The prover may not want to broadcast his identity and his location every time he needs location-proofs. Similarly, witnesses may want to hide their identity and their location. Hence, private information must be kept secret from all the participants but not from an independent trusted third-party, called *judge*. Indeed, the judge must be allowed to retrieve the identities of the participants, in order to detect collusions with the prover. An ideal location-proof system must then have the following properties [4].

1. **Correctness** : location proofs generated honestly by a prover with the collaboration of honest witnesses must always be accepted by the verifier.

2. **Unforgeability** : a prover cannot obtain/modify valid location proofs for a location he is not, or at a different time.
3. **Non-transferability** : location proofs are valid only for the prover who generated them. They cannot be exchanged.
4. **Traceability** : given a proof, the judge must be able to retrieve the identity of the witness who signed it. *New property - not in [4].*
5. **Location and identity privacy** : the location and the identity of the witnesses and the prover must be kept secret from other participants.
6. **Unlinkability** : given two distinct location proofs, a participant cannot guess whether they have been generated by the same witness, nor whether they concern the same prover. This obviously does not stand for the judge.
7. **Storage sovereignty** : the prover is responsible for storing his own location proofs. No one is able to access them without the prover's agreement.

In this paper, we propose a privacy-aware location-proof system that fulfills these properties. It relies on two protocols: a *location-proof gathering* protocol (allowing a prover to obtain proofs from witnesses) and a *location-proof verifying* protocol (allowing a verifier to validate the correctness of a proof). The first one ensures that both the prover and the witnesses keep secret their identity and their location. Once the location proofs have been obtained from witnesses, a prover must keep them securely and may use them later on to convince verifiers. For efficiency reasons, no centralized server is used during the gathering protocol.

Our scheme relies on *multi-party computations* and *group signature schemes* to protect the identity and the location of all participants. It assumes also that the participants have mobiles/vehicles with directional antenna to locate their neighbors. Such a solution can complement the classical distance-bounding protocols [1].

The security of our solution is analyzed against *malicious* and *semi-honest* adversaries. The former is a prover trying to obtain invalid location proofs, whereas the later is any participant (prover, witness or verifier) trying to obtain the private information on other participants. *Static collusions* between a prover and some of the witnesses against other witnesses are also considered.

Several solutions that partially fulfill the objectives were proposed. Unfortunately, most of them require to broadcast participants' identity and/or location. Using distance-bounding protocols, Singelee *et al.* [13] manage to obtain location proofs from witnesses, but without addressing the privacy issue. Sastry *et al.* [12] rely on impersonal local access points to locate participants in a given region. The location and the identity of the participants are transmitted and the access points simply grant access to nearby location-based services. Saroiu *et al.* [11] have the same goal. They use short range wireless communications to obtain location proofs and give access to location-based services. In this case, proofs can be used later on, as needed. Similar approaches are presented in [8,10]. In a more general setting, Zhu *et al.* [14] propose to use any participant as a witness instead of impersonal devices. Identities are protected by a set of pseudonyms, but all proofs (including pseudonyms and locations) are stored in a centralized authority, raising some privacy and efficiency issues. Finally, Gambs *et al.* [4] propose

a solution to get rid of the central authority. Unfortunately, only identities are protected through a group-signature scheme.

## 2 Problem Statement

Let us suppose the participants have devices (phone or vehicle) equipped with directional antennas, allowing to locate a transmitting device in 90°-quadrants with respect to their position and orientation. Depending on his location and orientation, a witness would be able to locate a prover in only one of the four reference orthogonal half-planes (*North*, *East*, *South*, *West*), as shown in Fig. 1.

Thus, our location problem can be stated as follows. Consider $n$ witnesses having located a prover $\mathcal{P}$ in half-planes with respect to their position. Looking for a location proof, $\mathcal{P}$ wants to obtain an authenticated description of the intersection of these half-planes, as shown in Fig. 2, while the witnesses want to protect their identity and their private information. This can be reduced to find the maximum (or minimum) of the private $x-$ or $y-$coordinates of the witnesses.

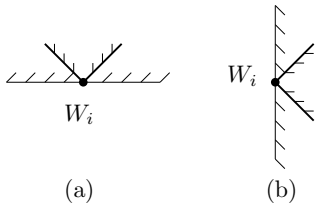


**Fig. 1.** Half-planes from 90-quadrants

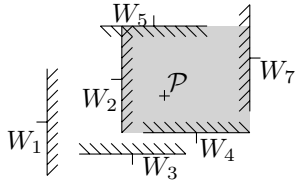


**Fig. 2.** Intersecting half-planes

### 2.1 Location-Proof Generation Protocol Outline

Protocol 1 presents the outline of our approach. After sharing the ephemeral additive homomorphic public key and the direction of each witness (**Step 1**), the idea is to find the intersection of the witness-defined orthogonal half-planes approximating the prover's position (**Steps 2-3**), and generate location proofs from it (**Steps 4-5**).

Our method relies on an *additive homomorphic encryption scheme*, such as Paillier's cryptosystem [9], and a *unique group-signature* scheme [3], that possesses the following properties : correctness, unforgeability, anonymity, traceability, unlinkability and uniqueness. The last property assures that two signatures on the same message by a given group member share a lot of common bits, breaking the unlinkability property in this very particular case. The uniqueness property prevents that an accomplice of the prover $\mathcal{P}$, or $\mathcal{P}$ himself, simulate the presence of multiple witnesses (Sybil attack).

In our scheme, groups can be dynamically managed and each participant $U$ has a signing key $gsk_U$. Let $\mathrm{GS}(gsk_U, m)$ denote the private signature function

**Input:** Each witness $W_i$ knows his position $(x_i, y_i)$ and his signature key $gsk_{W_i}$. The encryption function of the verifier $E_V(\cdot)$ is public.
**Output:** $\mathcal{P}$ obtains an authenticated location proof from his neighbor witnesses.
**Step 1** : **Initialization**
  $\mathcal{P}$ broadcasts a request: "*I'd like location proofs at time* $\tau$".
  **forall** *accepting witness* $W_i$ **do**
    Find the direction $d_i$ of $\mathcal{P}$ ($N$, $S$, $W$ or $E$) and generate an ephemeral public key $N_{W_i}$ (used in the min/max computation protocol).
    Send back $(d_i, N_{W_i})$ to $\mathcal{P}$.
  $\mathcal{P}$ broadcasts to all witnesses $\mu = (\tau, (d_i, N_{W_i})_{1 \leq i \leq n})$ and $GS(gsk_{\mathcal{P}}, \mu)$
  **forall** *accepting witness* $W_i$ **do**
    Find the presence of his key $N_{W_i}$ – if not, abort.
    Return the signature $GS(gsk_{W_i}, \mu)$.
  $\mathcal{P}$ broadcasts $\{GS(gsk_{W_i}, \mu) | 1 \leq i \leq n\}$.
  **forall** *accepting witness* $W_i$ **do**
    Find if all the signatures are valid and different. If not, abort.
**Step 2** : **forall** *accepting witness* $W_i$ **do**
  Run a min/max computation protocol with other witnesses (Protocol 3).
**Step 3** : $\mathcal{P}$ gets $E_V(x_{i_{min}})$, $E_V(x_{i_{max}})$, $E_V(y_{i_{min}})$ and $E_V(y_{i_{max}})$ (Protocol 5).
**Step 4** : $\mathcal{P}$ transfers these encrypted results to all witnesses.
**Step 5** : All $W_i$ sign the proof, using $gsk_{W_i}$, and send it to $\mathcal{P}$ (Protocol 2).

**Protocol 1:** Location proof generation

of the message $m$ with $gsk_U$ and $\mathrm{GV}(g_v, m, \sigma)$ the public verification function, that allows anyone to verify the signature $\sigma$ of $m$ with the public group key $g_v$.

### 2.2 Adversary Models

The following definitions of adversary models are extracted from [6] :

**Definition 1 (semi-honest adversary model).** *A semi-honest adversary follows the protocol specification exactly, but it may try to learn more information than allowed by looking at the messages that it received and its internal state. This model is also known as the passive or honest-but-curious adversary model.*

**Definition 2 (malicious adversary model).** *A malicious adversary may use any efficient attack strategy and thus may arbitrarily deviate from the protocol specification. This model is also known as the active adversary model.*

In this paper, we stress that there are two different motivations for the prover. First, the main motivation of a malicious prover is to obtain a valid proof that he is at a given location at a given time, when in fact he is somewhere else. In this case, the prover has to deviate from the protocol, while remaining undetected. Otherwise, legitimate witnesses would abort and alert the judge.

On the other hand, a curious prover may be interested in getting information on his neighbors (identity or precise location). Since the identity of a witness

relies on the security of the group-signature scheme used, the potential risk is low. At best, the prover can expect to get the location of an unknown participant.

The witnesses could be interested in discovering more information on their neighbors. However, since a witness has quite less possibilities than a prover, a malicious witness would be better to act as a prover with his neighbors.

Similarly, the verifier does not participate in the gathering location protocol and thus can only follow the semi-honest adversary model to try to get more information on the witnesses and the prover.

Finally, notice that a prover can always obtain a valid but faked location proof from accomplices. The verifier and the judge can always determine the number of witnesses, having participated in the protocol. If they determine that this number is too low, they may reject the valid proof anyway.

To sum up, our scheme is secure against the following adversaries :

- A malicious prover willing to obtain fake location proofs.
- A semi-honest prover, witness or verifier trying to violate other participants' privacy.

In the rest of the paper, Section 3 presents how to build encrypted location proofs against a malicious adversary, and how to verify them. In Section 4, a new solution to the secure multi-party maximum computation problem is described. It relies on a modified version of a classical two-party comparison protocol presented in Section 5.

## 3 Location-proof Gathering and Verifying

Let us first assume that the prover $\mathcal{P}$ has obtained somehow the four encrypted optimum values $E_V(x_{i_{min}})$, $E_V(x_{i_{max}})$, $E_V(y_{i_{min}})$ and $E_V(y_{i_{max}})$ describing the rectangle in which he lies from his neighbor witnesses (Section 4 will show how to obtain them). Unfortunately, nothing proves that he has not chosen these values himself and encrypted them with the verifier public key. The goal of **Steps 4-5** of Protocol 1 is specifically to prevent that a prover can do so. In this section, we design a protocol allowing each group of witnesses to certify the value of their corresponding optimum value. In the remaining of this section, we will focus only on $E_V(x_{i_{max}})$, defining the western side of the locating rectangle.

### 3.1 Location-Proof Gathering

We assume (w.l.o.g.) that the public key $N_V$ of the verifier is 2048 bit-long and that the witnesses are at most at one kilometre from the prover. If the scale of the grid system is one meter, the difference $x_{i_{max}} - x_i \leq 2^{10}$ uses at most $l_x = 10$ bits. We define $l_k = |N_V| - (l_x + 1)$. Our method for generating the location proofs is presented in Protocol 2. If a witness follows the protocol, the verifier would be able to retrieve the value $k_i(x_{i_{max}} - x_i) + r_i$. This value is such that:

$$k_i(x_{i_{max}} - x_i) + r_i > \quad 2^{l_x} \text{ iff } x_i < x_{i_{max}} \tag{1}$$

$$-2^{l_x} < k_i(x_{i_{max}} - x_i) + r_i < \quad 2^{l_x} \text{ iff } x_i = x_{i_{max}} \tag{2}$$

$$k_i(x_{i_{max}} - x_i) + r_i < -2^{l_x} \text{ iff } x_i > x_{i_{max}} \tag{3}$$

**Input:** $\mathcal{P}$ knows $E_V(x_{i_{max}})$. Each witness $W_i$ has his value $x_i$ and his signature key $gsk_{W_i}$. Each witness knows the number of participants $n$, $\mathrm{GS}(gsk_{\mathcal{P}}, \mu)$, and the key $N_V$ of the verifier with function $E_V(\cdot)$.
**Output:** $\mathcal{P}$ obtains a location proof from each witness.
**Step 1** : $\mathcal{P}$ broadcasts the randomized version of $E_V(x_{i_{max}})$.
**Step 2** : **forall** *witness* $W_i$ **do**
  Choose randomly $k_i \in_R [\![2^{l_x+1}; 2^{l_k} - 1]\!]$ and $r_i \in_R [\![-2^{l_x} + 1; 2^{l_x} - 1]\!]$.
  Compute $E_V(k_i(x_{i_{max}} - x_i) + r_i) = (E_V(x_{i_{max}}) \cdot E_V(-x_i))^{k_i} \cdot E_V(r_i)$.
  Send $E_V(k_i(x_{i_{max}} - x_i) + r_i)$ to $\mathcal{P}$.
**Step 3** : $\mathcal{P}$ broadcasts to all $(E_V(k_i(x_{i_{max}} - x_i) + r_i))_{1 \leq i \leq n}$.
**Step 4** : **forall** *witness* $W_i$ **do**
  Check the presence of $E_V(k_i(x_{i_{max}} - x_i) + r_i)$ – if not, abort.
  Define $\nu = ((E_V(k_i(x_{i_{max}} - x_i) + r_i))_{1 \leq i \leq n}, E_V(x_{i_{max}}), n, \mathrm{GS}(gsk_{\mathcal{P}}, \mu))$.
  Sign $\sigma_i = \mathrm{GS}(gsk_{W_i}, \nu)$ and send it to $\mathcal{P}$.
**Step 5** : $\mathcal{P}$ stores $\nu$, $\mathrm{GS}(gsk_{\mathcal{P}}, \nu)$ and all witness signatures $\sigma_i$.

**Protocol 2:** Location-proof gathering protocol

If all the participants follow the protocols, Case (2) must happen at least once and Case (3) never. This can be confirmed by the verifier $V$. Thus, $V$ can detect if a malicious prover deviates in **Step 1** and uses an invalid value. On the other hand, if a malicious prover deviates in **Step 3** and drops (or alters) some values, at least one witness can abort the protocol and alert the judge, by sending it any value signed by the prover (such as $GS(gsk_{\mathcal{P}}, \mu)$ of Protocol 1), which the judge can trace thanks to the properties of the group-signature scheme. Finally, the prover cannot deviate in **Step 5** due to the unique group-signature scheme.

### 3.2 Security Properties of the Overall Process

We have now to argue that the overall process to obtain the location proofs respects all the security properties listed in the introduction.

Since the unique group signature scheme [3] is unforgeable, the prover $\mathcal{P}$ cannot forge new proofs, except with his own key. In **Step 5** of Protocol 1, such an opportunity is impossible. $\mathcal{P}$ would have to generate two distinct signatures on the same message, contradicting the uniqueness property of the signature scheme. In fact, the judge would identify any transgressing participant in this step, due to the traceability property of the signature scheme. Thus, the unforgeability and traceability properties of our location-proof protocol are ensured.

In **Step 1** of Protocol 1, the prover broadcasts a message $\mu$ and its signature $\mathrm{GS}(gsk_{\mathcal{P}}, \mu)$. This links the timestamp and the $n$ ephemeral keys of the witnesses. Since this signature is included in the final proofs signed by the witnesses, the location proof is valid only for the participant able to produce the valid signature $\mathrm{GS}(gsk_{\mathcal{P}}, \mu)$, confirming the non-transferability property of the protocol.

Due to the unlinkability property of the group signature scheme, the location proof associated to $\mathrm{GS}(gsk_{\mathcal{P}}, \mu)$ would not be linkable with another location proof associated to a different signature $\mathrm{GS}(gsk_{\mathcal{P}}, \mu')$ done by the same prover.

Similarly, the signatures of the witnesses in Protocol 2 would also not be linkable. Thus, the unlinkability property of our location-proof protocol is guaranteed.

The privacy of the identity follows from the property of group signature scheme. Similarly, the privacy of the positions $(x_i, y_i)$ relies on the semantic property of the encryption scheme and the randomization process (see Section 5.2). Unfortunately, the last step of Protocol 2 leaks some information through $E_V(k_i(x_{i_{max}} - x_i) + r_i)$. The verifier can guess some bits of $x_i$. However, we can show that the Shannon entropy $H(X|Y = k_i(x_{i_{max}} - x_i) + r_i)$ is still close to $H(X|X \leq x_{i_{max}})$.

The prover obtains his location proofs during **Step 5**. Then, he stores them until he needs to convince the verifier, ensuring the storage sovereignty property.

### 3.3 Location-Proof verifying

Finally, the correctness property has to be shown. The prover $\mathcal{P}$ wants to convince the verifier $V$ that $E_V(x_{i_{max}})$ is indeed the maximum value. So, he sends:

- His position $x$, the message $\mu$ and his signature $\mathrm{GS}(gsk_{\mathcal{P}}, \mu)$. The message contains the timestamp $\tau$ and the number of witnesses $n$ (see Protocol 1).
- The randomized value of maximum $E_V(x_{i_{max}})$ (see Protocol 5).
- The $n$ proofs $E_V(k_i(x_{i_{max}} - x_i) + r_i)$ and the witness signatures $\sigma_i$ of $\nu = [(E_V(k_i(x_{i_{max}} - x_i) + r_i))_{1 \leq i \leq n}, E_V(x_{i_{max}}),$ $n, \mathrm{GS}(gsk_{\mathcal{P}}, \mu)]$ (see Protocol 2).

The verifier proceeds to several verifications. He first decrypts $E_V(x_{i_{max}})$ and checks if $x_{i_{max}} < x$. Then, he checks that the $n$ proofs are generated by $n$ distinct participants, different from $\mathcal{P}$. This verification is based on the uniqueness property of the group signature scheme. All the signatures of the message $\nu$ must be different. The verifier also asks the judge to check that $\mathrm{GS}(gsk_{\mathcal{P}}, \mu)$ was generated using $gsk_{\mathcal{P}}$, ensuring that $\mathcal{P}$ took place in the proof generation protocol. The final step is to make sure that $E_V(x_{i_{max}})$ is really the maximum value of the witnesses. From the values of $E_V(k_i(x_{max} - x_i) + r_i)$ in $\nu$, the verifier can check that there is an index $j$ s.t. $-2^{l_x} \leq k_j(x_{i_{max}} - x_j) + r_j < 2^{l_x}$, and that there is no index $j$ s.t. $k_j(x_{i_{max}} - x_j) + r_j < -2^{l_x}$.

If all these verifications succeed, the verifier should be convinced that $\mathcal{P}$ was indeed at the east of $x_{i_{max}}$ at the given time. If any of these steps fails, it reveals a malicious action by either the prover or a witness. But unlike the prover, witnesses do not have any incentive to cheat. If some proofs are missing, the prover might have deleted them on purpose, or a witness may have aborted because of a deviation of the prover.

## 4 Secure Multi-party Maximum Protocol

In this section, we introduce a new approach for secure multi-party maximum computations. The main purpose is to enable a third party (the prover) to determine the owner of the maximum value among a set of $n$ participants (or witnesses). The prover is the only party who gets a result from this protocol.

**Input:** The witnesses $S_1 = \{W_1, W_2, \cdots, W_n\}$. Each $W_i$ has a private value $x_i$.
**Output:** $\mathcal{P}$ determines $i_{max} = \arg\max\{x_i | 1 \leq i \leq n\}$.
**for** $i = 1$ **to** $\lceil \log(n) \rceil$ **do**
  **for** $j = 2^{i-1}$ **to** $2^i - 1$ **do**
    **Step 1** : $\mathcal{P}$ does the following steps :
      $\mathcal{S} = \emptyset$
      **if** $|S_j|$ *is odd* **then**
        Select $Single \in_R S_j$ s.t. $Single$ is not marked.
        Mark the witness $Single$ and add it to $\mathcal{S}$.
      Pair the elements of $S_j \setminus \mathcal{S}$ − pair the marked witnesses.
    **Step 2** : Each pair of witnesses uses Protocol 4, and $\mathcal{P}$ obtains the index of the owner of the greater value.
    **Step 3** : $\mathcal{P}$ selects $k \in_R \{0, 1\}$ and computes the following sets:
      $S_{2j+k} = \mathcal{S} \cup \{\text{the set of the losing witnesses}\}$
      $S_{2j+\overline{k}} = \mathcal{S} \cup \{\text{the set of the winning witnesses}\}$
**Step 4** : $\mathcal{P}$ determines the index Set $i_{max}$ of this witness.

**Protocol 3:** Secure maximum computation based on binary tree

The basic idea of our protocol is to use iteratively a dedicated secure two-party comparison protocol, that (i) enables the prover $\mathcal{P}$ to know which one of the two witnesses owns the greater private value without having to know this value, and (ii) guarantees that if one of the witnesses has already lost a comparison against another witness, the prover would not get any further information. This protocol (Protocol 4) is presented in Section 5.

### 4.1 The Protocol Description

Protocol 3 presents our approach for maximum computations. The prover regroups subsets of witnesses in a binary tree. In each node, the witnesses of the associated subset are paired and run the dedicated secure two-party comparison protocol (Protocol 4). At the end of each round, the prover gets the results of these comparisons and can eliminate half of the remaining witnesses. If a witness does not participate in any further comparison, he can deduce that he was farther away from the prover than his latest paired witness. Similarly, if one keeps participating in the protocol, he knows he has won every previous comparisons. Thus, the protocol should be adapted to ensure that witnesses keep participating in the protocol even if they have been eliminated. However, the comparisons with eliminated witnesses must be randomized and meaningless for the prover.

First assume that the number of witnesses is a power of 2. In the initial round, the prover pairs the $2^k$ witnesses all together. Each of these pairs runs the two-party comparison protocol. At the end of a round, the winners and the losers are regrouped independently. This process is then applied recursively on each subset. Hence, two witnesses would never be paired twice together. After $i$ iterations, there would be $2^i$ subsets of $2^{k-i}$ witnesses. One of these subsets

would be composed of only winners and all the other subsets would be composed of witnesses which have lost in a previous round.

Consider now the general case of $n$ witnesses. The prover pairs the witnesses. If there is an odd number of witnesses in a subset, one of them (called *Single* in Protocol 3) would be *doubled*, and considered as both a winner and a loser.

Finally, notice that the witnesses do not communicate with each other directly. Otherwise, it would be simple to find out which one is closer to the prover due to the directional antennas. Communications must go through $\mathcal{P}$.

### 4.2 The Protocol Security

The security of our maximum computation protocol relies on these objectives:

1. the prover cannot get any information from the two-party comparison protocol if at least one of the witnesses has been already eliminated previously,
2. the prover cannot get any information on the value of any witness, and
3. the witnesses cannot get any information from the comparison protocol.

The prover does the pairing and acts as the intermediary for the two-party comparison protocol. He can then observe all the messages exchanged between the witnesses. Thus, the objectives (1) and (2) rely on the security of the two-party comparison protocol. This will be addressed in Section 5.

Objective (3) relies on the indistinguishability of the subsets $\mathcal{S}_j$ in the round $i$ of Protocol 3, for $2^{i-1} \leq j \leq 2^i - 1$. If the two-party comparison protocol is secure, the only way for a semi-honest witness to get any information on the comparisons is to find if he is in the subset of the winners. All of the subsets in the round contain the same number of witnesses. Since the indices of the subset are chosen randomly, any of these subsets can be the subset of the winners.

### 4.3 The Protocol Analysis

The secure maximum computation problem have already been studied (e.g., [2, 5]). However, as far as we know, their computational and communication complexity are in $\mathcal{O}(n^2)$. Such complexities are not suitable for portable or embedded devices. Our method only requires approximately $\frac{n}{2}\lceil\log(n)\rceil$ two-party comparisons, at the cost of leaking $n-1$ comparison results. This follows directly from the underlying binary tree orchestrating the comparisons.

In order to show the complexity of Protocol 3, a few facts have to be proven. Since some witnesses may have been doubled, they may have to be compared at least twice in any given round of comparisons. We consider that the comparisons of a marked witness are resolved consecutively. We may then wonder if a round has to be split into more than two consecutive stages (i.e. if a witness has to be doubled several times).

The first step is to show that in any subset of witnesses at any round, there are at most two marked witnesses. This can be seen as an invariant of the protocol. Let us first assume that a subset $S_j$ contains at most two marked witnesses at

the beginning of the round. If $|S_j|$ is even, the subsets $S_{2j}$ and $S_{2j+1}$ may contain at most one marked element. Otherwise, if $|S_j|$ is odd, one new witness would be marked and the subsets $S_{2j}$ and $S_{2j+1}$ may contain at most two marked elements - the new one and one of the old ones. This implies that, for any subset of odd cardinality in a non-final round, there are at least one unmarked witness that can be marked and doubled if needed. Marking twice the same witness is unnecessary. As a corollary of this analysis, we have the following fact:

**Fact 1** *Sets having two marked witnesses at the end of a round would contain one previously marked witness and a newly doubled witness.*

The second step is to show that any combination of comparisons can always be split into at most two stages in any given round. Consider the hypothetical cycle $\{W_{i_1}, W_{i_2}\}, \{W_{i_2}, W_{i_3}\}, \cdots, \{W_{i_k}, W_{i_1}\}$ of comparisons between marked witnesses in a given round. Each of these pairs belongs to a different subset of witnesses. If $k$ is even, these comparisons can be split easily into two independent stages. This is optimal since a marked witness may have to be compared with two other witnesses. Now, if $k$ is odd, consider the chain $W_{i_1} - W_{i_2} - W_{i_3} - \cdots - W_{i_k} - W_{i_1}$. By Fact 1, alternate witnesses would have been just doubled in the round. This is impossible since the length of the cycle is odd. Therefore, no cycle of comparisons of odd length may exist. This implies that two stages per round are enough to order the comparisons. As a result, the total number of stages is greater than $\lceil \log(n) \rceil$ and lower than $2\lceil \log(n) \rceil$.

## 5 Secure Two-party Comparison Protocol

In this section, we propose a specific two-party comparison protocol (Protocol 4) that enables a third party (the prover $\mathcal{P}$) to know which one of the two participants (the witnesses $A$ and $B$) owns the greater private value without having to know this value explicitly. This can be used iteratively, so that if one of the participants has already lost a comparison against another participant, he should not give any further information to the third party. Such a protocol can be obtained by adapting the protocol of Lin and Tzeng [7], chosen for efficiency.

Given an integer $x$, let us define the following sets for our comparison protocol: $T_0^x = \{x_1 x_2 ... x_{i-1} 1 | x_i = 0\}$ and $T_1^x = \{x_1 x_2 ... x_i | x_i = 1\}$. Let $T_j^x[i]$ denote the $i^{th}$ element of $T_j^x$, if it exists. Lin and Tzeng's protocol relies on this lemma:

**Lemma 1.** *[7] For $x, y \in \mathbb{N}$, $x > y$ if and only if $T_1^x \cap T_0^y \neq \emptyset$.*

Our comparison protocol has been developed to be used in our multi-party maximum protocol presented in the previous section. It relies heavily on a probabilistic additive encryption scheme such as Paillier's cryptosystem [9]. As mentioned earlier, there should be no direct communication between the participants.

### 5.1 The Protocol Correctness

Let us first assume that the private values $s_A$ and $s_B$ have been initialized to zero by $A$ and $B$, respectively. To simplify the notations, let us assume w.l.o.g.

**Input:** The $l$-bit private values $a$ and $b$ of $A$ and $B$. Keys $N_A$, $N_B$ and $N_{\mathcal{P}}$ with functions $E_A(\cdot)$, $E_B(\cdot)$ and $E_{\mathcal{P}}(\cdot)$. The private values $E_{\mathcal{P}}(s_A)$ and $E_{\mathcal{P}}(s_B)$ of $A$ and $B$, respectively. The public hash function $h(\cdot)$.

**Output:** $\mathcal{P}$ determines whether $a > b$ or $a \leq b$.

**Step 1** : $A$ does the following steps :

Compute $T_1^a$ and the $l$-element vector $\gamma$, so that $\gamma_i = h(T_1^a[i])$ if it exists, otherwise, $\gamma_i$ is simply a random value.
Pick a random $c \in_R \mathbb{Z}_{N_B}$.
Return $(E_A(\gamma_1), \cdots, E_A(\gamma_l))$ and $E_B(c)$ to $B$ through $\mathcal{P}$.

**Step 2** : $B$ does the following steps after decrypting $E_B(c)$:

Compute $T_0^b$ and the $l$-element vector $\delta$, so that
$E_A(\delta_i) = E_A(k_i(h(T_1^a[i]) - h(T_0^b[i])) + r_B)$
$= (E_A(\gamma_i) \cdot E_A(-h(T_0^b[i])))^{k_i} \cdot E_A(r_B)$
where $k_i, r_B \in_R \mathbb{Z}_{N_A}$ s.t. $(k_i, N_A) = 1$. Otherwise, $\delta_i$ is a random value.
Pick randomly a permutation $\pi_B(\cdot)$ and $\alpha, \beta \in_R \mathbb{Z}_{N_{\mathcal{P}}}$ s.t. $(\alpha, N_{\mathcal{P}}) = 1$.
Return $E_{\mathcal{P}}(s_B - r_B + c)$, $E_A(\alpha)$, $E_A(\beta)$ and
$(E_A(\delta_1^*), \cdots, E_A(\delta_l^*)) = \pi_B(E_A(\delta_1), \cdots, E_A(\delta_l))$ to $A$ through $\mathcal{P}$.

**Step 3** : $A$ does the following steps :

Decrypt the elements $E_A(\delta_i^*)$ and compute the vector $\mu$ homomorphically, so that
$E_{\mathcal{P}}(\mu_i) = E_{\mathcal{P}}((\delta_i^* - r_B + s_B + s_A + r_{A,i}) \cdot {r_{A,i}}^{-1})$
$= (E_{\mathcal{P}}(\delta_i^* + r_{A,i}) \cdot E_{\mathcal{P}}(s_A) \cdot E_{\mathcal{P}}(s_B - r_B + c) \cdot E_{\mathcal{P}}(-c))^{{r_{A,i}}^{-1}}$
where $r_{A,i} \in_R \mathbb{Z}_{N_{\mathcal{P}}}$ s.t. $(r_{A,i}, N_{\mathcal{P}}) = 1$.
Return $(E_{\mathcal{P}}(\mu_1^*), \cdots, E_{\mathcal{P}}(\mu_l^*)) = \pi_A(E_{\mathcal{P}}(\mu_1), \cdots, E_{\mathcal{P}}(\mu_l))$,
where $\pi_A(\cdot)$ is a random permutation, to $\mathcal{P}$.

**Step 4** : $\mathcal{P}$ decrypts the cyphertexts $E_{\mathcal{P}}(\mu_i^*)$.

If one of the elements of $\mu^*$ is equal to 1, then $a > b$ and $\mathcal{P}$ sets $s'_A = 0$. Otherwise, $a \leq b$ and $\mathcal{P}$ sets $s'_A = 1$. $\mathcal{P}$ returns $E_{\mathcal{P}}(s'_A)$ to $A$.

**Step 5** : $A$ does the following steps, once $\alpha$ and $\beta$ have been retrieved :

Update $E_{\mathcal{P}}(s_A) \leftarrow E_{\mathcal{P}}(s_A + k_A \cdot s'_A)$ using $E_{\mathcal{P}}(s_A) \cdot E_{\mathcal{P}}(s'_A)^{k_A}$, where $k_A \in_R \mathbb{Z}_{N_{\mathcal{P}}}$.
Return $E_{\mathcal{P}}(\alpha s'_B + \beta) = (E_{\mathcal{P}}(1) \cdot E_{\mathcal{P}}(s'_A)^{-1})^{\alpha} \cdot E_{\mathcal{P}}(\beta)$ to $B$ through $\mathcal{P}$, since $s'_B = 1 - s'_A$.

**Step 6** : $B$ does the following steps :

Retrieve $E_{\mathcal{P}}(s'_B) = (E_{\mathcal{P}}(\alpha s'_B + \beta) \cdot E_{\mathcal{P}}(-\beta))^{\alpha^{-1}}$.
Update $E_{\mathcal{P}}(s_B) \leftarrow E_{\mathcal{P}}(s_B + k_B \cdot s'_B)$ using $E_P(s_B) \cdot E_P(s'_B)^{k_B}$, where $k_B \in_R \mathbb{Z}_{N_{\mathcal{P}}}$.

**Protocol 4:** Secure two-party comparison protocol determining which participant has the greater private value.

that the permutation functions are the identity function. At the end of **Step 2**, there is an index $i^*$ such that $\delta_{i^*} = r_B$, iff $a > b$. This follows from Lemma 1 and the fact that the hash function is collision-free. Consequently, at the end of **Step 3**, if $s_A$ and $s_B$ are both still equal to 0, there would be an element $\mu_{i^*} = r_{A_{i^*}} \cdot r_{A_{i^*}}^{-1} = 1$, iff $a > b$. Thus, $\mathcal{P}$ would know the result of the comparison. On the other hand, if at least one of the participants has randomized his private

value $E_{\mathcal{P}}(s_*)$, due to a previous comparison, no element of the vector $\mu$ would be equal to 1, except if $\delta_i - r_B + s_A + s_B \equiv 0 \mod \mathbb{Z}_{\mathcal{P}}$. In any case, the result would be meaningless.

### 5.2 The Protocol Security

To prove the security of Protocol 4 w.r.t. *semi-honest* polynomially-bounded adversaries trying to get more information on other participants, we have to show that these objectives are achieved: **(1)** $A$ cannot find $b$, **(2)** $B$ cannot find $a$, **(3)** $\mathcal{P}$ cannot find neither $a$ nor $b$, **(4)** the result of the comparison is known only to $\mathcal{P}$, **(5)** no one knows the first index $i^*$ that differentiates $a$ and $b$, **(6)** $\mathcal{P}$ eliminates $A$ or $B$, **(7)** there is no information leaking if $A$ or $B$ has been already discarded, and **(8)** $\mathcal{P}$ cannot simulate $A$ or $B$ and have a coherent result.

First, consider the information sent by $A$ in **Step 1**. $T_1^a$ gives a bit-encoding of $a$. Due to the semantic security of Paillier's cryptosystem, $\mathcal{P}$ and $B$ cannot get any information on $a$ (Objectives (2) and (3)). Notice that the exact same $\gamma$ (including random values) must be produced by $A$ at any iteration. Otherwise, a collusion of $\mathcal{P}$ and $B$ can set $E_A(\delta_i) = E_A(\gamma_i) \cdot E_A(\gamma'_i)^{-1}$ and have an encoding of $a$. Either $\delta_i$ would be equal to 0, if $a_i = 1$, or be a random value, if $a_i = 0$.

Now, let us demonstrate that the vector $\delta$ in **Step 2** does not leak any information on $b$ to $A$. Since $r_B \in_R \mathbb{Z}_{N_A}$, knowing $\delta_i$ does not give any information on $k_i(h(T_1^a[i]) - h(T_0^b[i])$. Besides, even if $y = k_i(h(T_1^a[i]) - h(T_0^b[i]))$ could be inferred, for any value $x$ of $h(T_1^a[i]) - h(T_0^b[i])$ s.t. $(x, N_A) = 1$, there is a unique $k^*$ s.t. $y \equiv k^* \cdot x \mod N_A$. In other words, the vector $\delta$ follows a uniform distribution of $\mathbb{Z}_{N_A}^l$ and independent of $b$ (Objectives (1) and (5)).

We now show that the vector $\mu^*$ in **Step 3** does not leak any information, except whether $a > b$ or not. If neither $A$ nor $B$ has already been discarded, $\mu_i^* = (k_j \cdot r_{A,j'}^{-1})(h(T_1^a[j]) - h(T_0^b[j])) + 1$, for some $j$ and $j'$. The value $k_j \cdot r_{A,j'}^{-1}$ can be seen as a random value selected in $\mathbb{Z}_{N_{\mathcal{P}}}$. If $a > b$, one of these values is 1, and the others are random values of $\mathbb{Z}_{N_{\mathcal{P}}}$. If $a \leq b$, $\mu^*$ is a uniformly random vector of $\mathbb{Z}_{N_{\mathcal{P}}}^l$ (Objective (3)). Due to the permutations, no information on the index differentiating $a$ and $b$ can be inferred (Objective (5)).

The semantic security of Paillier's cryptosystem assures that $A$ cannot retrieve neither the values of $r_B$ nor $\mu_i^*$ in **Step 3** as well as $A$ and $B$ cannot retrieve the value of $s'_A$ in **Step 4** (Objective (4)).

In the last two steps, $s_A$ and $s_B$ are updated. Since $\mathcal{P}$ cannot infer the values of $\alpha$ and $\beta$ in **Step 2**, it cannot manipulate the value of $s'_B$ in **Step 6** in such a way that $s'_B = 0$. At least one of the participants would then have his value $s_* \neq 0$, achieving Objective (6). Finally, notice that once $s_A$ or $s_B$ is a random number different than 0, $\mu^*$ follows an independent uniform distribution of $\mathbb{Z}_{N_{\mathcal{P}}}^l$. Hence, no conclusion follows from the value of $\mu^*$ in **Step 4** (Objective (7)).

Finally, note that $\delta$ and $c$ are necessary to obtain the result and that they are encrypted respectively with $A$'s and $B$'s public keys. Assuming that these keys have been properly exchanged and have not been tampered with by $\mathcal{P}$, a polynomially-bounded $\mathcal{P}$ cannot simulate $A$ or $B$ successfully. In such a case, the result of the protocol would then be meaningless (Objective (8)).

**Input:** $\mathcal{P}$ knows $i_{max}$. Each witness $W_i$ has his values $x_i$ and $E_{\mathcal{P}}(s_{W_i})$. Public keys $N_{\mathcal{P}}$ and $N_V$ with functions $E_{\mathcal{P}}(\cdot)$ $E_V(\cdot)$.
**Output:** $\mathcal{P}$ obtains $E_V(x_{i_{max}})$.
**Step 1** : **forall** *witness* $W_i$ **do**
Generate a random number $\alpha_i \in_R \mathbb{Z}_{N_{\mathcal{P}}}$.
Compute $E_{\mathcal{P}}(\alpha_i + s_{W_i}) = E_{\mathcal{P}}(\alpha_i) \cdot E_{\mathcal{P}}(s_{W_i})$ and return it to $\mathcal{P}$.
**Step 2** : $\mathcal{P}$ does the following steps:
Compute $\alpha_{i_{max}}$ from $E_{\mathcal{P}}(\alpha_{i_{max}} + s_{W_{i_{max}}})$ received from $W_{i_{max}}$.
Broadcast to all witnesses $E_V(\alpha_{i_{max}})$.
**Step 3** : **forall** *witness* $W_i$ *receiving* $E_V(\alpha_{i_{max}})$ **do**
Compute $E_V(\alpha_{i_{max}} - \alpha_i + x_i) = E_V(\alpha_{i_{max}}) \cdot E_V(-\alpha_i) \cdot E_V(x_i)$.
and return it to $\mathcal{P}$, only if it is the first request for that proof generation.
**Step 4** : $\mathcal{P}$ does the final steps:
Receive $E_V(x_{i_{max}})$ from $W_{i_{max}}$.
Randomize it $E_V(x_{i_{max}}) \leftarrow E_V(x_{i_{max}}) \cdot r^{N_V}$, for $r \in_R \mathbb{Z}^*_{N_V}$.

**Protocol 5:** Maximum transfer protocol

Let us briefly consider the collusion between $A$ and $\mathcal{P}$ against $B$. In such a case, $A$ and $\mathcal{P}$ accept to exchange all their private information. Due to $\pi_B(\cdot)$, $A$ and $\mathcal{P}$ cannot obtain the index of the bit that differentiates $a$ and $b$. Moreover, due to the multiplication of each element of $\mu$ by a distinct $k_i$, $A$ and $\mathcal{P}$ cannot compute $h(T_1^a[i^*]) - h(T_0^b[i^*])$, except if the hashes are equal, which has been discarded anyhow. Thus, $\mathcal{P}$ does not discover more information with the help of $A$. Similarly, $B$ and $\mathcal{P}$ do not gain more information neither. The index of the bit that differentiates $a$ and $b$ is hidden by the permutation $\pi_A(\cdot)$, and it is impossible to compute $\delta^*$ without knowing the values $r_{A,i}$ generated by $A$.

### 5.3 The Protocol Complexity

Following the fact that communications are made through $\mathcal{P}$, any message sent between $A$ and $B$ is counted twice. The size of a public key $N$ is denoted by $|N|$. Notice that ciphertexts are $2|N|$-bit long in Paillier's cryptosystem.

For any iteration, there are eight communications and $(10l + 22)|N|$ bits transferred. A maximum of $4l + 6$ cryptographic operations are computed by $A$, $2l+8$ by $B$ and only $l+1$ by $\mathcal{P}$. By cryptographic operations, we mean encryption, decryption and modular exponentiation. If either $A$ or $B$ was eliminated, $\mathcal{P}$ does not have to decrypt the result in **Step 4**: only one encryption is needed.

### 5.4 The Maximum Transfer

Using Protocols 3 and 4, the prover $\mathcal{P}$ knows the index $i_{max}$ of the witness that has the maximum value. However, $\mathcal{P}$ needs to obtain $E_V(x_{i_{max}})$, which corresponds to the maximum value encrypted with the verifier's public key. $\mathcal{P}$ does not want to inform which witness has been selected, but the discarded

| | Each witness | Prover | Communications | Bits sent |
|---|---|---|---|---|
| Protocol 1 (overall system) | negl + Protocols 2, 3 and 5 | negl | $2m+2$ + 4× Protocols 2, 3 and 5 | negl |
| Protocol 2 | negl | negl | $2n+2$ | $(4n+2)\|N\|$ |
| Protocol 3 | $< 2\lceil \log n \rceil$ ×Protocol 4 | $\approx \frac{n}{2}\lceil \log n \rceil \times$ Protocol 4 | | |
| Protocol 4 | $\leq 4l+6$ | $l+1$ or 1 | $\leq 8$ | $\leq (10l+22)\|N\|$ |
| Protocol 5 | negl | negl | $2n+1$ | $(4n+2)\|N\|$ |

**Table 1.** Complexity of the system

witnesses do not want to provide their location uselessly. Protocol 5 manages to reach both objectives. It relies on the fact that $W_{i_{max}}$ ends up with the internal value $E_{\mathcal{P}}(s_{W_{i_{max}}}) = E_{\mathcal{P}}(0)$ at the end of Protocol 4 (which correspond to $s_A$ or $s_B$ in Protocol 4). The other witnesses have a random $s_{W_i}$.

The security of Protocol 5 is easy to show. The security of all encrypted messages relies on the semantic security of the cryptosystem. In **Step 1**, $\mathcal{P}$ receives only random values from the witnesses. In **Step 2**, he picks one of them and broadcasts it back to all witnesses encrypted with the verifier's public key. A witness would return a meaningful value in **Step 3** only if his internal random value $\alpha_i$ is the additive inverse of the value sent by $\mathcal{P}$. In this case, the witness would return his encrypted position. Otherwise, he would return a random encrypted value. Finally, $E_V(x_{i_{max}})$ is randomized to conceal it from the witness $W_{i_{max}}$. In term of complexity, if broadcasting generates only one communication, $2n+1$ messages of $2|N|$ bits are exchanged during the protocol.

## 6 Complexity of the overall system

We have detailed the computational and communication complexity in each sub-protocols, but we are now interested in the the complexity of the entire of location-proof system (Protocol 1), depending on the number of witnesses. For simplicity, let us assume there are $m = 4n$ witnesses and $n$ in each directions.

Table 1 presents the number of cryptographic operations processed by the prover and by each witness, the number of communications and the bits exchanged during the different protocols. We only deal with the worst case : we consider a marked witness for the computational complexity in Protocol 3 and we consider only witness $A$ in Protocol 4. This can obviously be optimized by giving role $B$ to marked witness as most as possible. The complexity of Protocol 3 is an approximation of the total number of comparisons. An exact formula is given in Section 4.3. In Protocol 4, it has been shown that $\mathcal{P}$ runs $l+1$ operations in $n-1$ comparisons, and only 1 otherwise, then the number of operations run by the prover in Protocol 3 and 4 is approximately $(n-1)l + \frac{n}{2}\lceil \log n \rceil$.

To summarise, the global complexity, both in terms of computations and communication, is in $\mathcal{O}(n \log n)$ for the prover and $\mathcal{O}(\log n)$ for a witness. In comparison, most previous location-based systems have a complexity for the

prover in $\mathcal{O}(n)$ and $\mathcal{O}(1)$ for a witness. This is due to the fact that witnesses do not need to interact with each other. However, location-privacy requires such interactions, and thus we do not reach the same objectives.

## 7 Conclusion

In this paper, we have presented a privacy-aware location-proof system, allowing a prover to generate location-proofs with the cooperation of nearby witnesses, without having to know explicitly any participant's identity and position. Our scheme relies on a novel secure multi-party computation protocol, allowing a third-party to find which participant has the maximum value and approximating the region in which he is. The proofs are then signed with a group signature scheme, protecting the identity of the participants and allowing to detect any adversary trying to impersonate multiple witnesses. Finally, our scheme assumes that participants' devices are equipped with directional antennas. Although this is not a technological challenge, it would be interesting to get rid of it.

## References


1. X. Bultel, S. Gambs, D. Gérault, P. Lafourcade, C. Onete, and J.-M. Robert. A prover-anonymous and terrorist-fraud resistant distance-bounding protocol. In *Proc. of WISec*. ACM, 2016.
2. R. Cramer, S. Fehr, Y. Ishai, and E. Kushilevitz. Efficient multi-party computation over rings. In *Proc. of EUROCRYPT*, pages 596–613. Springer, 2003.
3. M. Franklin and H. Zhang. Unique group signatures. In *Proc. of ESORICS*, pages 643–660. Springer, 2012.
4. S. Gambs, M.-O. Killijian, M. Roy, and M. Traoré. Props: A privacy-preserving location proof system. In *Proc. of SRDS*, pages 1–10. IEEE, 2014.
5. O. Hasan, L. Brunie, and E. Bertino. Preserving privacy of feedback providers in decentralized reputation systems. *Computers & Security*, 31(7):816–826, 2012.
6. C. Hazay and Y. Lindell. *Efficient Secure Two-party Protocols: Techniques and Constructions*. Springer Science & Business Media, 2010.
7. H.-Y. Lin and W.-G. Tzeng. An efficient solution to the millionaires' problem based on homomorphic encryption. In *Proc. of ACNS*, pages 456–466. Springer, 2005.
8. W. Luo and U. Hengartner. Veriplace: a privacy-aware location proof architecture. In *Proc. of SIGSPATIAL*, pages 23–32. ACM, 2010.
9. P. Paillier. Public-key cryptosystems based on composite degree residuosity classes. In *Proc. of EUROCRYPT*, pages 223–238. Springer, 1999.
10. A. Pham, K. Huguenin, I. Bilogrevic, I. Dacosta, and J.-P. Hubaux. Securerun: Cheat-proof and private summaries for location-based activities. In *Proc. of TMC*, pages 2109–2123. IEEE, 2015.
11. S. Saroiu and A. Wolman. Enabling new mobile applications with location proofs. In *Proc. of HotMobile*, pages 1–6. ACM, 2009.
12. N. Sastry, U. Shankar, and D. Wagner. Secure verification of location claims. In *Proc. of WISEC*, pages 1–10. ACM, 2003.
13. D. Singelee and B. Preneel. Location verification using secure distance bounding protocols. In *Proc. of MASS*, pages 7–14. IEEE, 2005.
14. Z. Zhu and G. Cao. Applaus: A privacy-preserving location proof updating system for location-based services. In *Proc. of INFOCOM*, pages 1889–1897. IEEE, 2011.