2023 62nd IEEE Conference on Decision and Control (CDC)
December 13-15, 2023. Marina Bay Sands, Singapore

Robust decentralised proof-of-position algorithms for smart city applications

Aida Manzano Kharman amm3117@ic.ac.uk[1], Pietro Ferraro[1], Anthony Quinn[1,2], and Robert Shorten[1]

[1]*Imperial College London, Dyson School of Design Engineering*
[2]*Trinity College Dublin, Electronic and Electrical Engineering*

***Abstract*— Motivated by the ever growing use of location-based services, we present a decentralised class of algorithms called Tree-Proof-of-Position (T-PoP). Most of the current proofs of location are centralised, thus forgoing verifiablity and privacy for the users. Decentralised solutions also exist, but they suffer from drawbacks that make them unsuitable for realistic, adversarial use cases. T-PoP algorithms rely on the web of interconnected devices in a smart city to establish how likely it is that an agent is in their claimed position. T-PoP operates under adversarial assumptions, where some agents are incentivised to be dishonest. We present a theoretical model for T-PoP and its security properties, and we validate this model through a large number of Monte-Carlo simulations. We specifically focus on two instances of T-PoP and analyse their security and reliability properties under a range of adversarial conditions. Use-cases and applications are discussed towards the end of the paper.**

I. Introduction

A basic problem across a range of application areas is the need for decentralised agents to be able to certify their position in a trustworthy and certifiable manner. For example, in crowd-sourcing applications arising in the context of smart cities, the need for agents to certify their position in a trustworthy manner is essential; one such use-case arises when vehicle cameras are used to identify available parking locations or electric charge points [1]. Other examples are emerging in the context smart mobility applications in which vehicles need to prove their location to avail of certain services; for example, in the case of hybrid vehicles using their electric engine mode in a city to avoid an environmental charge (as in London); when making use of a fast or slow lane on a highway and paying the associated charge; or when 'infotainment' services are offered to vehicles when adopting certain positions.

Our objective in this paper is to propose a suite of algorithms whereby agents may certify their position collaboratively, but in a decentralised manner. Our algorithms are designed to be robust in the sense that they do not require the use of centralised infrastructure, and in the sense that they are designed to operate successfully in an adversarial environment (in the presence of agents that are interested in coercing the system for their own personal objectives). The need to be independent of a centralised authority is fundamental to our work, as such authorities may be compromised or a subject to data and privacy leaks [2]. While our original motivation arises from automotive applications, the work presented here is relevant and may find application in other disciplines and applications, and may also help to encode basic elements of fairness, social justice and civil rights. More specifically, in an era characterised by fake news, and deep fake technology, the ability to associate sensing information with a verifiable geographic position, is not only essential in establishing the veracity of sensed information, but also in developing robust decision making analytics based on these data. Currently, across many such applications, sensed information is assumed more trustworthy if a number of people agree on it. In scenarios where we cannot verify what happened ourselves, we search for 'truth' by listening to our peers and believing what a majority claims [3]. Hence, our research question becomes: how can we provide agents with the ability to claim that they are at a given place in time, without the security of our protocol depending on the honesty of a centralised authority? While we are not the first to address this research question, existing solutions do not address the requirements of applications in smart city contexts. Specifically, the solution must be truly decentralised, and it must be robust to attacks whilst preserving user privacy.

Our work is motivated by recent developments in distributed ledger technologies (DLT); in particular, in the design of distributed acyclic graph (DAG)-based distributed ledgers. However, while the design of such ledgers is concerned with architectures that can provide peer-to-peer trustworthy record keeping, we are interested in realising DAG-based algorithms that encode reliable position information.

A. Related Work

Several papers have been published on the topic of proof-of-position; see for example [4], [5], [6], [7], [8]. Most are unsuitable for the applications we are interested in due to unrealistic trust assumptions and *de facto* centralisation in the proposed systems. We now provide a snapshot of some of this prior work.

An early example of a decentralised proof-of-location scheme, termed APPLAUS, was presented in [9]. The APPLAUS scheme makes a number of valuable contributions; namely it looks to address collusion attacks using graph clustering and computing a 'betweeness' metric. In [10], nodes in the graph that are weakly connected are considered less trustworthy. They also present a weight function that decays with time, and compute the trustworthiness of a node by calculating a node's ratio of approvals to neighbours. These contributions serve as a starting point for our approach. However, in their work, users must register their public/private keys with a trusted certificate authority, and so it is not a truly decentralised solution. A focal point of our work is that we do not assume a trusted centralised authority. Indeed, we argue that introducing this assumption makes a system *de facto* centralised and poses security and privacy risks. Another algorithm, known as SHARP, is introduced in [11]. Here, the authors present a private proximity test that does not require a user to reveal their actual location to a server, and furthermore, they present a secure handshake method wherein users do not need to have a pre-shared secret. A notable contribution is that a witness[1] may only extract the session key if they are indeed in the vicinity of the prover [2]. Security is ensured by requiring that location tags be unforgeable, thus implying that the protocol is robust against location cheating. A weakness of the protocol is that a user in a given location can generate a valid proof and relay it to a malicious agent in a different location. Another algorithm, known as Vouch+, is presented in [12]. This is another decentralised approach to proving location, with a focus on addressing high speed and platooning scenarios. The major disadvantage is that its security relies on selecting a proof provider which is honest. This assumption, in our opinion, is too strong. We aim to develop a protocol wherein the prover could lie, and the system would still have a probabilistic guarantee of detecting this. Another protocol, SPARSE [4], does not allow the prover to pick their own witnesses, making collusion significantly harder. Furthermore, SPARSE does address necessary security concerns, and achieves integrity, unforgeability and—very importantly—non-transferability. However, as in [12], the prover is assumed to be a trusted entity which supposedly does not publish users' identity and data.

B. Contributions

We present a generalised model for a class of decentralised proof-of-position algorithms, called T-PoP. We present a mathematical model to describe the operation of this class of algorithm and to facilitate further analysis. Simulations are presented that validate our mathematical model, and we present a framework for users to tailor the operating conditions of the algorithm to satisfy their security and reliability requirements. We also provide probabilistic guarantees of detecting dishonest provers and collusion attacks.

Comment: T-POP can also be implemented in a privacy preserving manner, since it does not require the agent to reveal their true position. Instead, a cryptographic commitment [13] to one's position suffices. Depending on the security requirements of the application, T-PoP users can pick a commitment scheme with varying binding and hiding, as long as the commitment scheme supports the computation of Euclidean distance between two points [3].

[1]An agent that verifies that they see another agent wishing to prove their position.

[2]An agent that wishes to prove their position.

[3]We showcase a proof of concept in our GitHub repository whereby the Euclidean distance between two agents can be computed in a privacy preserving manner using the Zama fully homomorphic encryption library [14]

Finally, we do not constrain the freedom of adversarial agents to misbehave. We consider not only the possibility that they are dishonest about their own position, but also that they are colluding to lie about other agents' position(s).

C. Structure of the paper

Our paper is structured as follows: first we introduce the T-PoP protocol and explain its functioning in section II; next, we present a theoretical model for the T-PoP class of algorithms in section III; finally, we simulate T-PoP in a more realistic scenario in section IV, thereby also validating our theoretical model.

II. Tree - Proof of Position (TPoP) protocol

We begin by providing a high level explanation of how the protocol operates. Subsequently, we will provide the necessary definitions for each stage, and explain them in detail. We assume that agents willing to participate in the protocol are situated in $T \subseteq \mathbb{R}^2$ (the protocol can be seamlessly extended to a three-dimensional space). Each agent a_i is characterised by their *true* position $s_i = (x_i, y_i) \in T$ and by their *claimed* position $\hat{s}_i = (\hat{x}_i, \hat{y}_i) \in T$, while the set of all agents is denoted by A. Notice that it is possible that $\hat{s}_i \neq s_i$ (in the event an agent is lying). An agent, a_j, is (allegedly) a_i's neighbour if $||\hat{s}_i - \hat{s}_j|| < r_i$, where $r_i > 0$ is each agent's range of sight. T-PoP is performed in three steps, as depicted in Figure 1:

- **Commit**: At the beginning of T-PoP, each agent, $a_i \in A$, commits to their claimed position, $\hat{s}_i$ nd publishes $\hat{s}_i$ on a distributed ledger (DL). This ensures that the agent's commitment[4] cannot be changed later.
- **Tree Construction**: Each agent, a_i, then constructs a tree of depth $d \in \mathbb{N}^+$, incorporating the committed positions of agents, called *witnesses*, at levels $l \in \{0, \ldots, d\}$. A specific a_i—which we denote as g—is the root of the resulting tree. These $g \in A$-indexed trees are also committed to the DL as they are part of the proof-of-position protocol. For every *prover*, g, the tree is constructed as follows:
  - g is is the root node at level 0.
  - For each $l \in \{1, ..., d\}$, each node at level $l-1$ will name w_l *witnesses*. A witness at level l is an agent, a_j, that is a neighbour (see above) of a witness, $a_i \in W_{l-1}$, at level $l-1$ (note that, if $\hat{s}_i \neq s_i$, and a_i is lying about their position, it is possible that a_i and a_j might not actually be *true* neighbours).

[4]The only necessary requirement for our protocol is that the commitment is binding [13] To ensure user privacy, we favour schemes that allow for the computation of the Euclidean distance between two points which can be achieved by leveraging encryption schemes that are fully homomorphic. It is also necessary to achieve non-repudiation, which can be done through the use of digital signatures. Frequently used examples include [15] and [16]. This ensures an agent cannot later deny having claimed to be in a given position [17]. Finally, non-transferability is needed to ensure that if an honest prover generated a valid location proof through T-PoP, they cannot then transfer their honest proof to a malicious actor. A user's identity is unique upon being issued, and should this be in the form of a private key, we introduce the assumption that users do not share it.

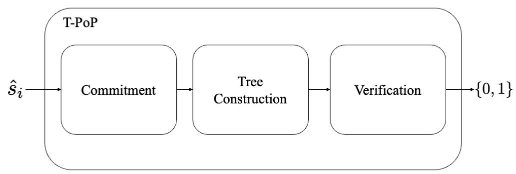


Fig. 1: High-level Overview of the T-PoP protocol

a_i is called the *parent* of witness a_j. The set of all witnesses at level l is called W_l, with $|W_l| \equiv n_l$.

– If a_j, was named a witness at some point in the tree, it should not be named again by another agent. If this happens, the prover will be considered dishonest.

In practice, the root node, g, names w_1 witnesses who in turn name w_2 witnesses and so on, until we reach depth d. The number of witnesses per level, n_l, can therefore be computed recursively:

$$n_l = w_l n_{l-1}, \; l = 1, \ldots, d, \tag{1}$$

with $n_0 \equiv 1$. Figure 2 depicts the operation of this process.

- **Verification**: The agent wishing to prove their position runs the verification stage with the tree as an input, initialized with $l = d$.
  1. Each witness at level l states whether their parent at level $l-1$ is their neighbour or not. If the answer is yes, and the witness has not yet been named in the tree, this witness becomes a confirmed level l witness. The total number of confirmed level l witnesses is denoted as $M_l \leq n_l$, and the total number of witnesses that confirm parent b at any level, l, is denoted by $K_b \leq w_l$. It follows that
  $$M_l = \sum_{b \in W_l} K_b \leq n_l \tag{2}$$
  2. If $K_b < tw_l$, $t \in (0, 1]$, parent b is eliminated from the tree. Here, t is a parameter of T-PoP, called the *threshold*, which is used to regulate the security and reliability properties of the algorithm, defined in Section III.
  3. If $M_l < tn_l$ then the algorithm interrupts and outputs that root g is lying about their position. Otherwise, we move on to level $l-1$ and we repeat this process. Note that any parent removed by the previous step will not be included in this next iteration of T-PoP.

T-PoP is therefore an algorithm depending on a set of parameters, $\theta \equiv \{t, d, w_1, ..., w_d\}$. The influence of these parameters on the performance of the algorithm will be explored in Section IV, via two examples. The pseudo-code for the *Tree Construction* and *Verification* stages of the protocol can be found in Algorithms 1 and 2 respectively.

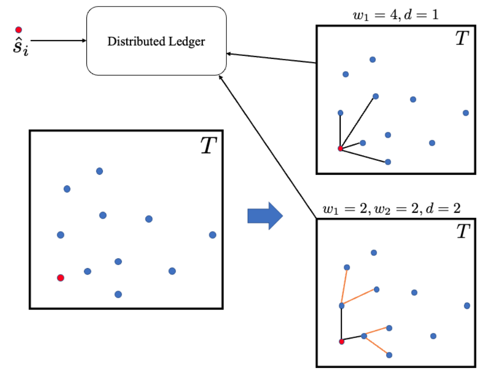


Fig. 2: Tree building examples. Agent a_i commits their alleged position $\hat{s}_i$ to a distributed ledger. The panel on the top right shows the construction of a tree for $d = 1$ and $w_1 = 4$, while the panel on the bottom right shows the construction of a tree for $d = 2, w_1 = 2, w_2 = 2$.

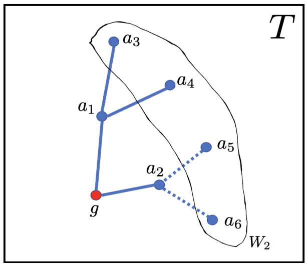
Example: Consider the T-PoP example in Figure 3, in which $\theta = \{t = 0.5, d = 2, w_1 = 2, w_2 = 2\}$, and so $n_1 = 2$ and $n_2 = 4$ (1). Solid arrows mean that a witness approves their parent and dotted lines mean that a witness does not approve their parent. Agents a_5 and a_6 are dishonest agents, so that their committed positions, $\hat{s}_5$ and $\hat{s}_6$, are different from their true positions. However, agent a_2 does not know this, it saw those cars next to it and it picked a_5 and a_6 as witnesses. So, a_5 and a_6 do not confirm that a_2 is a neighbour of theirs, whereas a_3 and a_4 confirm that a_1 is a neighbour of theirs. In line with point 2 of *Verification* (above), agent a_1 has enough confirmed witnesses ($K_{a_1} = 2 \geq t \times w_2 = 0.5 \times 2$) and stays in the tree, while agent a_2 does not have enough confirmed witnesses ($K_{a_2} = 0 < 0.5 \times 2$), and so a_2 is removed from the tree. However, since the total number of confirmed witnesses at level 2 is $M_2 = 2 \geq t \times n_2 = 0.5 \times 4$, T-PoP does not stop for g (*Verification*, point 3), and we move to level 1. At level 1, a_2 has been removed but a_1 confirms that g is its neighbour. As per points 2 and 3 of *Verification*, the final output of T-PoP is that g is *truthful* about their position. As can be seen in the example above, t is critical in determining the output of T-PoP. For instance, if $t = 1$, then $M_2 = 2 < t \times n_2 = 1 \times 4 = 4$, causing T-PoP to stop at point 3 of *Verification*, and returning an output of *untruthful* for g.

A. Possible Adversarial Behaviours

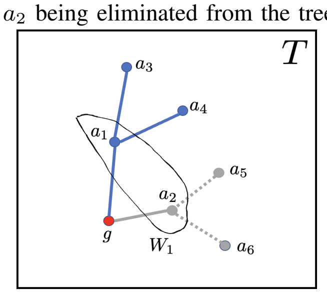
In order to analyse the properties of T-PoP, we introduce two qualities that each agent, $a_i \in A$, will exhibit:

Definition 1 (Honest and Dishonest agents). *Every $a_i \in A$ is either* honest *or* dishonest*. The set of honest agents is denoted by $H \subseteq A$, and the set of dishonest agents is denoted by $\overline{H}$. A dishonest agent will always commit a position $\hat{s}_i \neq s_i$. A honest agent on the other hand will always commit a position $\hat{s}_i = s_i$.*

Definition 2 (Coerced and Non-Coerced Agents). *Every $a_i \in A$ is either* coerced *or* non-coerced*. The set of coerced agents*



(a) We start by evaluating the outer level of the tree and we evaluate the witnesses in W_2. Agents a_5 and a_6 do not confirm that they see agent a_2, even though a_2 is an honest agent. This leads to agent a_2 being eliminated from the tree.



(b) We go down one level, and now evaluate the witnesses in W_1. a_2 has been eliminated by the tree (shown in grey) and so only agent a_1 is left.

Fig. 3: Example of T-PoP algorithm with $d = 2, w_1 = 2, w_2 = 2$.

is denoted by $C \subseteq A$, and the set of non-coerced agents by $\overline{C}$. A coerced agent will claim to see agents that are not actually in its vicinity, if the latter are dishonest.

a_i will interact with its neighbours in different ways—as defined next—depending on which of the four possible states it falls into with respect to the two 2-state qualities above.

Definition 3 (Neighbour-adding logic). *Every agent, $a_i \in A$, adds neighbours, a_j, according to the following logic:*

- *If $a_i \in \overline{H}$, it can add a_j as a neighbour if a_j's position, is within the range of sight r_i, of a_i's fake position, $\hat{s}_i \neq s_i$. This implies that a_i checks who is in the r_i-neighbourhood of the fake position that they committed.*
- *If $a_i \in H$, it can add a_j as a neighbour if a_j's committed position is within the range of sight, r_i, of a_i's true position, s_i.*
- *If $a_i \in \overline{C}$, it can only add a_j's true position, s_j, if this is within a_i's range of sight, r_i.*
- *If $a_i \in C$, it can add a_j's true position, s_j, if a_j is honest, and its fake position, $\hat{s}_j$, if a_j is dishonest.*

III. Theoretical Analysis

The stochastic nature of T-PoP is modelled via the graphical probability model in Figure 4, for the case where $d = 2, w_1 = 2, w_2 = 2$. We assume that the Honesty and Coercion states of each agent are independently and identically distributed (iid) Bernoulli trials, $\mathcal{B}(\cdot)$. Formally, for each agent, we define two independent random variables, $h \sim \mathcal{B}(p_h)$ and $c \sim \mathcal{B}(p_c)$, where $p_h \in [0, 1]$ and $p_c \in [0, 1]$

Algorithm 1 Tree Construction

Require: Prover a_i, Depth d, Number of witnesses $w_1, ..., w_l$

Initialise a_i as the root of the tree g and as a witness of level 0
for $l = 0, 1, ..., d-1$ **do**
  for Each witness a at level l **do**
    a names w_{l+1} witnesses among its neighbours
    All the named neighbours are added as nodes of G at level $l+1$, with a as parent node
  end for
end for
return G

Algorithm 2 Verification

Require: Tree G, Threshold t

Initialise $M_0, M_1, ..., M_{l-1}$ to 0
for $l = d-1, d-2, ..., 0$ **do**
  for Each witness a at level l **do**
    Set $C = 0$
    for Each b that has been named by a **do**
      if b confirms a and b unique in G **then**
        $C \longleftarrow C + 1$
        $M_l \longleftarrow M_l + 1$
      end if
    end for
    if $C < tw_{l+1}$ **then**
      Remove b from G
    end if
  end for
  if $M_l < \#\{\text{witnesses at level } l+1\}t$ **then**
    return False
  end if
end for
return True

are the probabilities of any agent being honest and coerced, respectively (and it follows that $1 - p_h$ and $1 - p_c$ are the probabilities of an agent being respectively dishonest and non-coerced). Depending on the outcome of these trials for a witness at level l, it will then deterministically confirm that the witness at level $l-1$, which named them, is its neighbour or not (note that agents might be lying about whether another agent is their *true* neighbour or not). The outcome of this interaction has been described in definition 3, and is summarized in the truth table (Table I). If agent, a_i, verifies agent a_j's position, the outcome is 1, and 0 otherwise.

In this model, we assume that the density of agents in T is very high. This means that provers, which are constructing their tree following Algorithm 1, are always able to find w_l witnesses at each level and that each witness is always unique. This assumption may not be satisfied in practice, since agents may be isolated and therefore not have enough witnesses around them. Nevertheless, studying the behaviour

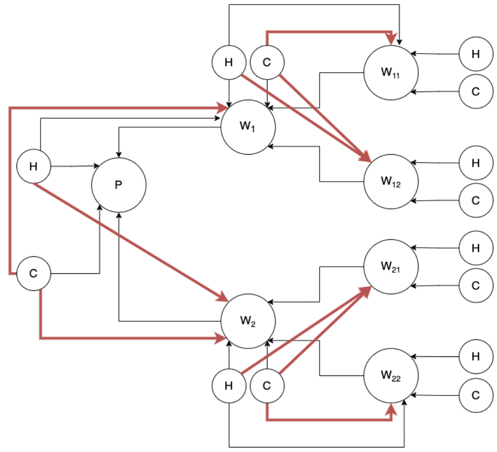


Fig. 4: Graphical Probability Model of T-PoP with parameters $d = 2, w_1 = 2, w_2 = 2$. The red lines indicate that those variables influence the output of a specific node.

a_i \ a_j	h and $\overline{c}$	h and c	$\overline{h}$ and c	$\overline{h}$ and $\overline{c}$
h and c	1	1	1	1
h and $\overline{c}$	1	1	0	0
$\overline{h}$ and c	1	0	1	0
$\overline{h}$ and $\overline{c}$	1	0	0	1

TABLE I: A truth table showing confirmation (1) or rejection (0) of a parent's (a_i) position by a witness (a_j), depending on the honesty (h) and coercion (c) states of each agent. Notice that the relationship between a_i and a_j is symmetrical.

of the model in this high-density scenario provides insight into the properties of T-PoP. Indeed, we argue that if an agent is honest but does not have sufficient witnesses, it is fair to consider them less trustworthy. Once the tree has been created, the *Verification* step can be used to provide the outcome of the algorithm, which can be either 0 (if the algorithm deems the prover dishonest) or 1 (if the algorithm deems the prover honest). Given a prover, g (the root of the tree), we define a random variable, $C(g) \in \{0, 1\}$, whose outcome depends on the ensemble of iid random variables, h, c, in its constructed tree, and on T-PoP parameters, $\theta \equiv \{t, d, w_1, ..., w_d\}$. In order to analyse T-PoP's performance, we consider two metrics: reliability and security.

Definition 4. Security, S, *is a conditional probability quantifying the ability of the algorithm to detect malicious agents. Specifically, it is the true-negative conditional probability, which, under stationarity assumptions, is independent of* $i \in \{1, \ldots, |A|\}$*:*

$$S \equiv \Pr[C(g) = 0 | a_i \in \overline{H}]$$

Definition 5. Reliability, R, *is a conditional probability quantifying the ability for the algorithm to detect honest agents. Specifically, it is the true-positive conditional probability. Once again, under stationarity assumptions:*

$$R \equiv \Pr[C(g) = 1 | a_i \in H]$$

In Figure 5, we display empirically evaluated R and S for two sets of parameters, respectively: $\theta_1 = \{t = 1, d = 1, w_1 = 6\}$ and $\theta_2 = \{t = 1, d = 2, w_1 = 2, w_2 = 2\}$, in each case varying p_h and p_c in their ranges, $[0, 1]$, with steps of 0.02. To emphasize the functional dependence of these probabilistic performance metrics on the honesty and coercion probabilities of the iid agents, we denote these metrics by $R(p_h, p_c)$ and $S(p_h, p_c)$, respectively. These are evaluated empirically via extensive Monte Carlo simulations of the graphical model. Specifically, we simulated 5000 trees for each of the two parameter settings above.

IV. Simulations

In this section, we present an agent-based simulator, coded in Python, to replicate a more realistic scenario for T-PoP and to validate the graphical theoretical model that we presented in the previous Section. Each agent has a number of varying attributes such as their range of sight, position, velocity, unique identifier and whether they are honest or dishonest, and coerced or not. Depending on the latter variables, each agent will commit to their true position or a fake one, and will add agents to their set of neighbours as outlined in definition 3. We then create an environment with a fixed density of agents in it, and place these randomly and uniformly across the environment. We allow them to move according to their velocity vector, within the bounds of the environment. Each time the agents move, all agents construct a new set of neighbours and discard the previous one. Next, each agent wishing to claim their position runs T-PoP; namely, they run the *Tree Construction* and the *Verification* algorithms. Our simulator can be found in this GitHub Repository.

Other key variables are the threshold, depth and number of witnesses used. A greater threshold increases security, but also reduces reliability. Increasing the number of witnesses increased both security and reliability. However, this may not be a suitable measure for sparser scenarios, or cases where agents are moving at high speed, potentially incurring a communication overhead. We advocate for the users to select the appropriate threshold, depth and number of witnesses based on the individual needs of their own application. Lowering the threshold can lower security, but provides more flexibility in the system. The user can then select an appropriate number of witnesses based on the expected density of their network, and use the depth parameter to find an appropriate trade-off between security and reliability, and communication overhead and flexibility.

A. Preliminary results

Our objective in this section is twofold. On the one-hand, we want to show some preliminary results on the performance of T-PoP for a given choice of operating conditions. On the other hand, we are interested in validating the results from the graphical probability model (Figure 4), with a view to creating an analytical framework for analysis of the T-PoP

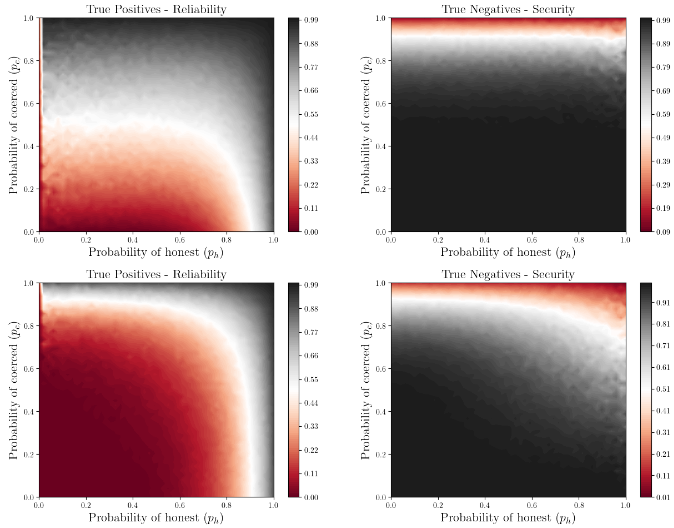


Fig. 5: T-PoP performance for the **graphical probability model** (Figure 4). The panels in the left column show reliability, R, while the panels in the right column show security, S. The first row is associated with model parameters, θ_1, while the second row is associated with model parameters, θ_2.

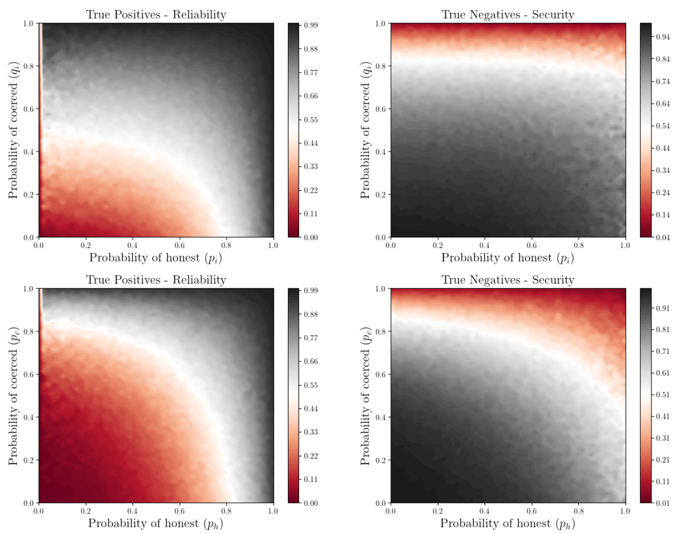


Fig. 6: T-PoP performance for the **agent-based model**. The panels on the left show reliability R, while the panels on the right show security, S. The first row is associated with model parameters θ_1, while the second row is associated with model parameters θ_2. Notice the close similarity to Figure 5.

class of algorithms. This gives us confidence that the results obtained for simple model parameter settings (e.g. d small) still hold in more realistic scenarios.

The simulations have been set up as follows: we considered each possible combination of p_h and p_c in the ranges $[0, 1]$, with steps of 0.02. For each combination we ran 50 Monte Carlo simulations and we computed empirical estimates of the values of $R(p_h, p_c)$ and $S(p_h, p_c)$. Simulations are set up in such a way that on average each agent has 50 neighbours in their range of sight r_i. While this number might appear very high, we wanted to make sure that the results obtained were comparable to the ones obtained with the graphical probability model. Moreover, real-life situations with high density of pedestrians (e.g., the underground during peak hours) would map well into this scenario. We ran these simulations for each set of parameters, θ_1 and θ_2. The results are shown in Figure 6.

T-PoP with θ_1 yields better performance overall (as both $R(p_h, p_c)$ and $S(p_h, p_c)$ are higher for each choice of p_h and p_c). The simulations with θ_2 show that decreasing the number of witnesses by a third and increasing the depth level by 1 allows us to achieve similar results. This is useful because—while the total number of nodes in each prover's tree is the same for both scenarios—a tree of depth 2 with 2 witnesses per parent places a smaller communication overhead on the prover, because it only needs to name 2 witnesses, as opposed to 6. Hence, the load is shared among the prover and the witnesses.

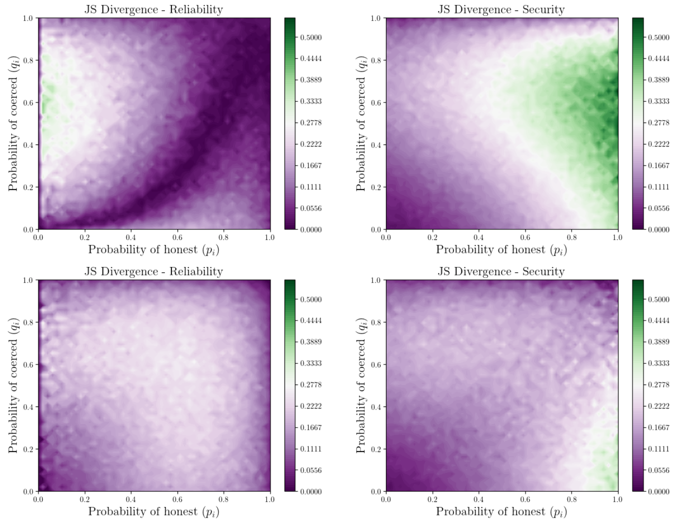


Fig. 7: Jensen-Shannon divergence (JSD) between R_s and R_m (left column) and between S_s and S_m (right column) for θ_1 (top row) and θ_2 (bottom row).

Overall, in high density scenarios, the results of both simulations show that—if $p_h > 0.9$ and $p_c < 0.2$—T-PoP is capable of achieving $S > 0.85\%$ and $R > 0.9\%$ for θ_1, and $S > 0.7\%$ and $R > 0.9\%$ for θ_2.

For lower proportions of honest agents and higher proportions of coerced agents (i.e. in the presence of many colluding, dishonest and coerced agents), the performance of T-PoP degrades. This is to be expected in a decentralised system such as T-PoP, since it is virtually impossible to distinguish between a group of honest agents verifying each other and a group of dishonest and coerced agents collaborating to verify each other in a fraudulent manner. Accordingly, we can observe across all figures that—even when the percentage of honest agents is low—the security remains high at the expense of reliability. We observe that—whilst, indeed, T-PoP can detect true negatives (i.e. be secure) in highly (and perhaps even unrealistically highly) adversarial environments—the drawback is that it penalises honest agents too harshly (i.e. is unreliable). This is a consequence of the collaborative nature of the algorithm. When the number of honest agents in the system is low (i.e. $p_h \downarrow 0$), they will—with high probability (w.h.p.)—be misclassified as dishonest because they will select dishonest witnesses w.h.p.

The density of agents in the environment vastly affected the performance of T-PoP. This was especially noticeable when the average number of agents per range of sight in the environment was lower than or equal to the total number of nodes of the tree being constructed, which greatly increased the number of False Negatives, thus making T-PoP unsuitable for low density environments. This is because—when the density is lower—the probability of selecting the same agent twice, or of not having sufficient neighbours to construct the tree, greatly increases.

B. Validation of the graphical model (Figure 4)

For validation of the graphical probability model, we make use of the Jensen-Shannon Divergence (JSD) [18] to quantify how close the probability distributions obtained through the agent-based model (i.e. the T-PoP implementation) and the graphical model are. In what follows, we refer to the values of R and S obtained from the simulated agent-based model as R_s and S_s, and the ones obtained from the graphical model as R_m and S_m. We compute two JSD-based metrics: (i) the (p_h, p_c)-indexed (i.e. pointwise) JSD map between R_m and R_s, and between S_m and S_s, respectively; and (ii) the global JSD between the normalized R_m and R_s maps, and the normalized S_m and S_s maps, respectively. By "normalized", we mean that each of these positive maps is divided by its element sum, yielding a probability mass function (pmf). In case (ii), we can therefore condense into a single number the relative performances displayed in the figures (R and S, respectively) for the simulated T-PoP system (s) and its graphical model (m) (Figure 4).

The results for the point-wise evaluation ((i) above) of the JSD are shown in Figure 7, while the global evaluation ((ii) above) is summarised in table II. Note that $0 \leq \text{JSD} \leq 1$, with lower values achieved when probabilities are close in value (i.e. in cases of good agreement between the behaviour of the simulated system and the graphical model). It is clear that—at least for high density scenarios—the behaviour of the graphical model closely mirrors that of the implemented T-PoP system. Nevertheless, the pointwise JSD results reveal significant discrepancies in the security (S) metric when $p_h \uparrow 1$ (i.e. for high proportions of honest users).

V. Comparison of Graphical Probability Model and Simulated Environment

In the graphical model, we consider only the agent's coercion and honesty attributes when it comes to witness approval, whereas in the simulated environment the approval

Parameters	$JSD(R_m, R_s)$	$JSD(S_m, S_s)$
θ_1	0.139	0.095
θ_2	0.174	0.059

TABLE II: Jensen-Shannon divergence (JSD) between the normalized reliability (R) and security (S) maps for two sets of model parameters.

is dependent not only on the honesty and coercion variables of the agents, but also on their range of sights, positions and velocity. The attributes that influence the agents' behaviour are their honesty and coercion status, and thus, for the graphical model, we do not consider variables such as range of sight, positions and velocities. In the graphical model, agents do not move and they do not build the tree for T-PoP according to other agents that are within their range of sight. Rather, agents build their trees assuming that other agents are always present, and approvals are made following Table I only. Upon building a more realistic simulation scenario, where we introduce variables that add noise and uncertainty to the behaviour defined in I, we observe that, at least for a high density scenario, T-PoP behaves, both qualitatively and quantitatively, in a similar way to the one characterised by our graphical model. This validates the hypothesis that, indeed, the most important attributes for the performance of T-PoP are those of honesty and coercion, in high density scenarios. It also suggests that the graphical model should be expanded to take into account different density settings.

VI. Conclusion

We have presented a proof-of-position class of algorithms that are fully decentralised. They can be run by any agent participating in the network and they do not assume trust in a central authority, nor do they rely on physical infrastructure. We also considered a range of attack vectors by allowing agents not only to lie about their own position, but also about others' positions. Our algorithm can also be computed in a privacy-preserving manner, as there is no need for the true location of an agent to be revealed to the network. We developed a graphical probability model for this class of proof-of-position algorithms, and statistically validated the model via comparative analysis of their respective performances. In future work, we will use the theoretical model to predict the performance of T-PoP as a function of its operating conditions, θ. Specifically, we will be interested in characterising the effect of the depth (d), threshold (t) and number of witnesses (w_l) on the security and reliability of the T-PoP class of algorithms. Developing such a framework can allow users to select the optimal operating conditions of the algorithm to meet their needs, based on their expected density, fault tolerance and proportion of honest and non-coerced agents in their system. The theoretical model will also allow performance guarantees to be deduced for T-PoP. Finally, we intend to explore the suitability of T-PoP for specific use-cases in the presence of more complex adversarial scenarios.

Acknowledgments: The authors would like to thank the IOTA Foundation for funding this research.

References

[1] R. Cogill, O. Gallay, W. Griggs, C. Lee, Z. Nabi, R. Ordonez, M. Rufli, R. Shorten, T. Tchrakian, R. Verago, *et al.*, "Parked cars as a service delivery platform," in *2014 International Conference on Connected Vehicles and Expo (ICCVE)*, pp. 138–143, IEEE, 2014.

[2] C. Fiesler and B. Hallinan, "'we are the product' public reactions to online data sharing and privacy controversies in the media," in *Proceedings of the 2018 CHI conference on human factors in computing systems*, pp. 1–13, 2018.

[3] S. C. Desai, B. Xie, and B. K. Hayes, "Getting to the source of the illusion of consensus," *Cognition*, vol. 223, p. 105023, 2022.

[4] M. R. Nosouhi, S. Yu, M. Grobler, Y. Xiang, and Z. Zhu, "SPARSE: privacy-aware and collusion resistant location proof generation and verification," in *2018 IEEE Global Communications Conference (GLOBECOM)*, pp. 1–6, IEEE, 2018.

[5] H. Alamleh and A. A. S. al-Qahtani, "A cheat-proof system to validate GPS location data," in *2020 IEEE International Conference on Electro Information Technology (EIT)*, pp. 190–193, IEEE, 2020.

[6] C. Javali, G. Revadigar, K. B. Rasmussen, W. Hu, and S. Jha, "I am Alice, I was in wonderland: secure location proof generation and verification protocol," in *2016 IEEE 41st conference on local computer networks (LCN)*, pp. 477–485, IEEE, 2016.

[7] W. Wu, E. Liu, X. Gong, and R. Wang, "Blockchain based zero-knowledge proof of location in IOT," in *ICC 2020-2020 IEEE International Conference on Communications (ICC)*, pp. 1–7, IEEE, 2020.

[8] 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)*, pp. 146–153, IEEE, 2018.

[9] Z. Zhu and G. Cao, "Toward privacy preserving and collusion resistance in a location proof updating system," *IEEE Transactions on Mobile Computing*, vol. 12, no. 1, pp. 51–64, 2011.

[10] Z. Zhu and G. Cao, "APPLAUS: A privacy-preserving location proof updating system for location-based services," in *2011 Proceedings IEEE INFOCOM*, pp. 1889–1897, IEEE, 2011.

[11] Y. Zheng, M. Li, W. Lou, and Y. T. Hou, "SHARP: Private proximity test and secure handshake with cheat-proof location tags.," in *ESORICS*, pp. 361–378, Springer, 2012.

[12] F. Boeira, M. Asplund, and M. Barcellos, "Decentralized proof of location in vehicular ad hoc networks," *Computer Communications*, vol. 147, pp. 98–110, 2019.

[13] O. Goldreich *et al.*, "Foundations of cryptography–a primer," *Foundations and Trends® in Theoretical Computer Science*, vol. 1, no. 1, pp. 1–116, 2005.

[14] Zama, "Concrete: TFHE Compiler that converts Python programs into FHE equivalent," 2022. `https://github.com/zama-ai/concrete`.

[15] D. J. Bernstein, N. Duif, T. Lange, P. Schwabe, and B.-Y. Yang, "High-speed high-security signatures," in *Cryptographic Hardware and Embedded Systems–CHES 2011: 13th International Workshop, Nara, Japan, September 28–October 1, 2011. Proceedings 13*, pp. 124–142, Springer, 2011.

[16] D. Johnson, A. Menezes, and S. Vanstone, "The elliptic curve digital signature algorithm (ECDSA)," *International journal of information security*, vol. 1, pp. 36–63, 2001.

[17] M. Swanson, J. Hash, and P. Bowen, "Guide for developing security plans for federal information systems," tech. rep., National Institute of Standards and Technology, 2006.

[18] M. Menéndez, J. Pardo, L. Pardo, and M. Pardo, "The Jensen-Shannon divergence," *Journal of the Franklin Institute*, vol. 334, no. 2, pp. 307–318, 1997.