SPARSE: Privacy–Aware and Collusion Resistant Location Proof Generation and Verification

Mohammad Reza Nosouhi*†, Shui Yu‡§, Marthie Grobler†, Yong Xiang*, and Zuqing Zhu‖
*School of Information Technology, Deakin University, Melbourne, Australia
† CSIRO's Data61, Melbourne, Australia
‡ School of Computer Science, Guangzhou University, 510006, China
§ School of Software, University of Technology Sydney, Sydney, Australia
‖School of Information Science and Technology, University of Science and Technology of China, Hefei, China
Email: *{mnosouhi, yong.xiang}@deakin.edu.au; ‡shui.yu@uts.edu.au; †Marthie.Grobler@data61.csiro.au; ‖zqzhu@ieee.org

***Abstract*—Recently, there has been an increase in the number of location–based services and applications. It is common for these applications to provide facilities or rewards for users who visit specific venues frequently. This creates the incentive for dishonest users to lie about their location and submit fake check–ins by changing their GPS data. To solve this issue, different distributed location proof schemes have been proposed to generate location proofs for mobile users. However, these schemes have some drawbacks: (1) they are vulnerable to either Prover–Prover or Prover–Witness collusions, (2) the location proof generation process is slow when users adopt a long private key, and (3) their implementation requires some hardware changes on mobile devices. To address these issues, we propose the Secure, Privacy–Aware and collusion Resistant poSition vErification (SPARSE) scheme to generate private location proofs for mobile users. SPARSE has a distributed architecture designed for ad–hoc scenarios in which mobile users generate location proofs for each other. Since we do not integrate any distance bounding protocol into SPARSE, it becomes an easy–to–implement scheme in which the location proof generation process is independent of the length of the users' private key. We provide a comprehensive security analysis and simulation which show that SPARSE provides privacy protection as well as security properties for users including integrity, unforgeability and non–transferability of the location proofs. Moreover, it achieves a highly reliable performance against collusions.**

***Index Terms*—collusion attacks detection and prevention; location–based services; location privacy; location proof systems.**

I. Introduction

The recent advancements in smartphone technology and positioning systems have resulted in appearance of new location–based services and applications. These services use real-time position data of a mobile user to provide his/her requested information such as the nearest ATM, restaurant or retail store [1–3]. Many of these applications like Yelp and Foursquare provide a benefit or reward for users who check–in regularly at specific locations (refer to [4] for other examples of such applications). However, it is possible for dishonest users to submit fake check–ins by changing their GPS data to gain more benefit. For example, in a research study, Zhang *et al* [5] found that 75% of Foursquare check–ins are false and submitted by dishonest users to obtain more rewards.

To address this issue, different systems have been introduced so far to prevent users from submitting fake location claims. These systems are called location proof systems in the literature [4], [6–12]. Generally, there are two types of such systems depending on the system architecture: *centralized* and *distributed* location proof systems. In the centralized location proof systems [4], [6], [7], a trusted fixed wireless infrastructure (like a WiFi access point) is employed at each site to check the physical presence of mobile users and generate location proofs (LPs) for them. A user can then submit his/her location claim with the service provider using the received LP. However, it might be too expensive for the service provider to employ a large number of access points (APs) for different sites or there might be some sites in which no fixed wireless infrastructure is accessible. For these scenarios, distributed location proof systems [8–12] have been proposed in which mobile users collaborate with the system and generate LPs for each other. In the literature, the user who wants to make a location claim and another user who generates a LP for him/her are called *prover* and *witness*, respectively.

Location proof systems can face different security and privacy challenges that must be addressed. Some of these challenges are common between both centralized and distributed location proof systems. For example, a dishonest user might submit a LP request with a neighbor AP in the centralized scenario or a mobile witness in the distributed scenario on behalf of a remote malicious prover. This attack is called a *Terrorist Fraud* or a *Prover–Prover* collusion in the literature [13], [14]. On the other hand, there are attacks that target the distributed location proof systems only. The reason is that in these systems, LPs are generated by mobile users which are not always trusted. For example, a dishonest witness might collude with a remote malicious prover and generate a fake LP for him/her. This is known as a *Prover–Witness* collusion in the literature [8]. Moreover, to preserve location privacy of users, the system must keep provers and witnesses anonymous during LP generation phase.

To the best of our knowledge, none of the current distributed location proof schemes can address all of these challenges at the same time. For example, the most recently proposed

978-1-5386-4727-1/18/$31.00 ©2018 IEEE

systems in this field are STAMP [8] and PROPS [12]. In these systems, the Bussard–Bagga distance bounding protocol [13] is integrated into the system to prevent *Terrorist Frauds* although Bay *et al* [15] has already broken this protocol and illustrated its inadequency in preventing *Terrorist Fraud* [15–17]. Furthermore, employing a distance bounding (DB) protocol prevents these systems from generating LPs fast when the users adopt a long private key [6]. The reason is that in DB protocols, users must perform an n–stage *fast–bit–exchange* process where n is the number of bits that the user's private key has. Obviously, this process takes long for large values of n (refer to [13] and [18] for more information about DB protocols). In addition, in these systems, the *Prover–Witness* collusions are not detected and prevented using a reliable mechanism. For example, in PROPS these collusions are prevented by using the LP shares. This method works only when the number of colluding users is less than the number of shares needed. In STAMP, these collusions are detected based on the LP transaction history between users. It considers a high likelihood of collusion for a user who has obtained most of his/her LPs from a specific group of witnesses. Although they have reached to a 90% success rate for collusion detection, this is not regarded as a high level of reliability.

In this paper, we propose the Secure, Privacy–Aware and collusion Resistant poSition vErification scheme (SPARSE) which provides secure and private LP generation and verification for mobile users. In the proposed scheme, we do not employ a DB protocol for protection against *Terrorist Fraud*. Instead we adopt a time–limited approach to make SPARSE resistant to these attacks. This introduces two advantages. Firstly, the speed of LP generation becomes independent of the length of the users' private key. Secondly, the costs of the system implementation is reduced since implementing a DB protocol requires some hardware changes on mobile devices [6]. Moreover, to address *Prover–Witness* collusions, we do not allow provers to choose their witnesses. Instead, the system performs a witness selection mechanism by which some witnesses are chosen and qualified to generate LPs for a specific prover. We show that by using this method, if the service provider creates necessary incentives for users to collaborate with the system and generate LP for each other, the success probability of these collusions is negligible.

The following are our contributions:

- We propose SPARSE, a secure and privacy–aware distributed location proof scheme for mobile users. Security analysis shows that SPARSE achieves the necessary properties of a secure and private location proof system including integrity, unforgeability and non–transferability of the LPs.
- We introduce a new witness selection mechanism and integrate it into the system to make SPARSE reliable when witnesses collude with a malicious prover to generate LPs for him/her.
- The reliability of the proposed scheme against *Prover–Witness* collusions is assessed through simulation which shows that the system achieves protection success rates better than 98%.

The rest of our paper is organized as follows. After a short review of the related work in section II, we introduce the proposed SPARSE scheme in section III. A comprehensive security analysis is provided in Section IV. Finally, after presenting the simulation results in Section V, we conclude the paper in Section VI.

II. Related Work

Previously proposed location proof systems can be classified into two categories depending on the system architecture: *centralized* and *distributed* systems. Since our proposed system has a distributed architecture, we focus on reviewing the previous research studies that are related to the distributed location proof systems.

One of the earliest distributed location proof systems is APPLAUS which has been proposed by Zhu *et al* [9]. In the proposed system, nearby mobile devices exchange information via their short–range Bluetooth interface to generate LPs for each other. Each mobile device registers a set of M public/private key pairs with a trusted Certificate Authority (CA) where the M public keys are the pseudonyms of a user. To protect the privacy of users, the mobile devices change their pseudonyms periodically. However, this approach imposes high communication and operation overhead because of the necessity of periodically changing pseudonyms and generating dummy LPs.

Another location proof system with a distributed architecture has been proposed by Davis *et al* [10] in which a privacy–preserving alibi (location proof) system is introduced. In the proposed mechanism, users do not reveal their identity during alibi generation. A user's identity is only disclosed when he/she decides to submit his/her alibi to a judge. However, they have not considered collusions and attacks in their designed system.

LINK, introduced by Talasila *et al* [11] is another distributed location proof protocol in which users collaborate with the system to verify the location of each other. In LINK, neighbour users are contacted by the prover via short–range wireless interface such as Bluetooth. Then, they send their verification messages to a trusted and centralized Location Certification Authority (LCA). The LCA determines the validity of the claim using three parameters, i.e., the spatiotemporal correlation between the users, each user's trust scores, and history of the trust scores. However, privacy issues have not been considered in the protocol design since a prover must broadcast his/her ID to the neighbor verifiers.

One of the main weaknesses of these kinds of location proof systems is that they are vulnerable against *Prover–Witness* collusions as witness nodes are not always trusted. A witness can generate a LP for a user while one or both are not at the claimed location. PROPS introduced by Gambs *et al* [12] is an example of distributed location proof protocols which has not proposed a reliable solution for *Prover–Witness* collusions. It

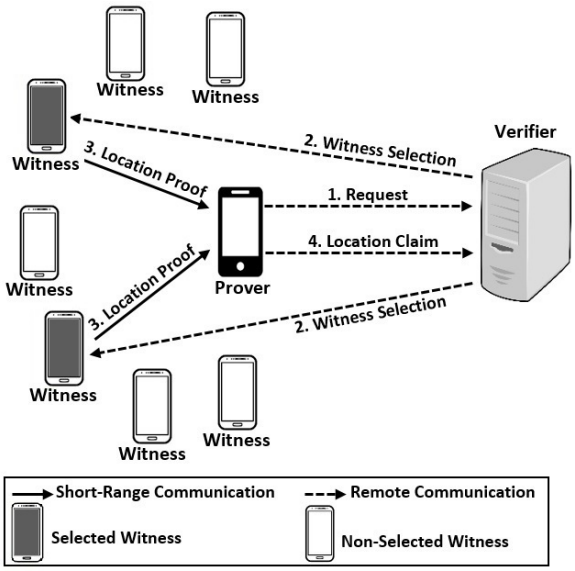

Fig. 1: The system architecture of SPARSE Scheme

allows users to act as witnesses and generate LPs for other users in a private way.

To the best of our knowledge, no good solution has been proposed yet to address *Prover–Witness* collusions although some significant efforts have been made so far. For example, in STAMP (Wang *et al* [8]) which is the most recent and significant work in this area, an entropy–based trust model is presented to solve this issue. However, their method is not able to detect or prevent *Prover–Witness* collusions with a very high level of certainty. Moreover, to protect STAMP against *Terrorist Frauds*, the authors have utilized the Bussard–Bagga protocol [13] while it has been shown that the Bussard–Bagga protocol can not provide the necessary protection against *Terrorist Frauds* [15–17] (this problem is faced by PROPS as well [12]). Furthermore, the computation time required by STAMP to generate LPs becomes long for large keys [8]. Although different novel methods have been introduced so far, each of these have their own constraints, i.e., privacy issues [4], [6], [11], vulnerability against collusions [8–12], high level of computation and communication cost [9], and expensive implementation [6], [7].

III. The SPARSE Scheme

A. System Model

Fig. 1 presents the proposed system architecture. As you see, SPARSE has a distributed architecture and consists of three types of entities:

Prover: A mobile user who wants to prove his/her location to a verifier.

Verifier: The entity that is authorized to assess and verify the provers' location proofs.

Witness: A mobile user who has accepted to generate a LP for his/her neighbor provers.

In the following we present some assumptions regarding our threat and trust model:

TABLE I: List of Notations

Notation	Description
$\|$	Concatenation symbol
$\mathcal{S}_u(m)$	Signature of user u on message m
$E_{ent}(m)$	Encryption of message m using public key of entity ent
Loc	GPS coordinates related to the prover's location
$Loc^{'}$	The witness's location
$time$	The current time
ID_P	The prover's ID
ID_W	The witness's ID

– Users (provers and witnesses) send their messages to each other via their short–range communication interfaces such as WiFi or Bluetooth.

– To obtain a fake LP, dishonest provers might provide the witnesses with fake information about their location or change the contents of a LP generated for him/her or another user. They might also collude with other users (provers or witnesses) to achieve their goal.

– Users never share their private key with each other [6–8].

– Witnesses are assumed to be untrusted. Thus, they may collude with a remote dishonest prover and issue a fake LP for him/her. Moreover, both provers and witnesses are untrusted from a privacy point of view.

– The verifier is supposed to be a trusted entity which does not publish users' identity and their data.

In the next subsection, we introduce our proposed location proof scheme, SPARSE.

B. SPARSE

The proposed scheme is executed in two separate phases: *Location Proof Generation*, and *Location Claim & Verification* (see fig. 2). Refer to table I for a short description about the cryptographic notations that are used in this paper. *1) Location Proof Generation*:

a. Prover: The prover starts the protocol by sending the following message m_1 to the verifier to inform it that he/she wants to submit a location claim.

$$m_1 = E_{Verifier}(Req \| \mathcal{S}_{Prover}(Req)),$$

where $Req = ID_P \| Loc$ is the prover's request.

b. Verifier: Upon receiving m_1, the verifier randomly selects K witnesses among those who are present at Loc. Then, it generates a unique ID for this location proof (ID_{LP}) and sends it to the prover and selected witnesses.

c. Witness: After receiving ID_{LP}, a selected witness generates a random sequence number rs and broadcasts the following message m_3 through its predefined short–range communication interface (Bluetooth or WiFi) for a period T (e.g., 100 ms).

$$m_3 = ID_{LP} \| rs$$

After this time, another rs is generated and broadcasted in a similar way. This process is repeated until the witness receives a response from the prover.

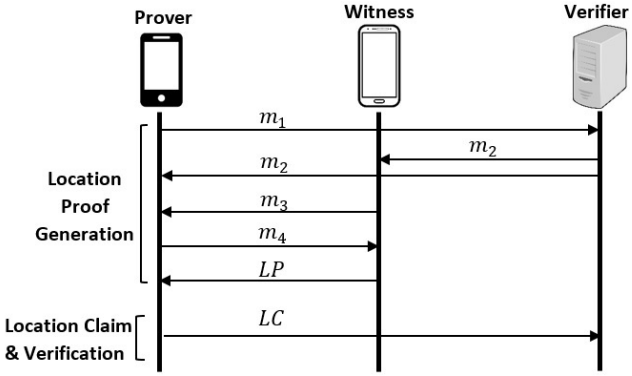

Fig. 2: Message exchange diagram for the proposed scheme.

d. Prover: When the prover device receives m_3, it first ensures that the ID_{LP} is the same as the one already received from the verifier. Otherwise, it just discards m_3 and continues to listen to the channel. If they are same, the prover must immediately compute message m_4 and send it to the witness:

$$m_4 = ID_{LP} \| rs \| E_{Verifier}(rs \| \mathcal{S}_{Prover}(rs))$$

e. Witness: Upon receiving m_4, provided that the ID_{LP} in m_4 is the same as the current location proof ID, the witness checks to see whether this rs is the last sequence number that had been broadcasted by itself. If it is, the witness generates the following location proof LP and sends it to the prover.

$$LP = E_{Verifier}(m_5 \| \mathcal{S}_{Witness}(m_5)),$$

where $m_5 = m_4 \| Loc^{'} \| time \| ID_W$. Otherwise, the following $null$ LP is sent to the prover:

$$LP = E_{Verifier}(m_5 \| \mathcal{S}_{Witness}(m_5)),$$

where $m_5 = null \| ID_W$.

2) Location Claim & Verification:

a. Prover: The prover generates the following location claim LC using the K received LPs from the K selected witnesses and submits it with the verifier:

$$LC = E_{Verifier}(m_6 \| \mathcal{S}_{Prover}(m_6)),$$

where $m_6 = LP_1 \| LP_2 \| \ldots \| LP_K \| ID_P$.

b. Verifier: After the verifier receives the LC, it checks the following items:

- Are the two ID_Ps received through messages m_1 and LC the same?
- Is the prover's signature on m_6 correct regarding the claimed ID_P?
- For each $LP_i, (i = 1, \ldots, K)$:
  - Is the witness with identity ID_W among the selected witnesses?
  - Is the witness signature on m_5 correct regarding the ID_W?
  - Are $time$ and Loc in an acceptable range of the current time and $Loc^{'}$ respectively?
  - Are the two random sequences rs in m_4 same?
- Is the number of non–$null$ LPs greater than a predefined threshold K_T?

If all the above checks are passed successfully, the verifier accepts this LC. Otherwise, the prover's claim is rejected.

IV. Security and Privacy Analysis

In this section, a comprehensive security and privacy analysis is presented to show that SPARSE achieves the fundamental security and privacy properties of a secure and privacy–aware location proof system described in [7], [10] and [12].

Resistance to Distance Frauds: In a distance fraud, a malicious prover tries to convince an honest witness (or a verifier) that his physical distance to the witness (or verifier) is less than what it really is. In SPARSE, the prover must sign and encrypt the random sequence rs in a limited time T over a short–range communication interface. If a malicious prover is not located in the communication range of the witness, he/she can not proceed with the attack. Thus, for him/her the only way to get the fake location verified is to collude with another user. In this section, we analyze the SPARSE performance against collusions separately.

Unforgeability: We consider several scenarios: If a dishonest prover wants to generate a LP by himself (without proving his proximity to a selected witness), the verifier will detect this. Note that the verifier checks the received ID_W with the signature on m_5. Since users do not share their private key, this prover can not compute the witness signature on m_5 even if he knows the identity of each selected witness. Moreover, if a malicious user wants to forge another user's LP, again the prover will detect it. The reason is that he must sign messages m_1 and m_6 using the victim's private key which is not accessible for him. Furthermore, if a malicious witness who is not among the selected witnesses wants to generate a LP for a prover, the issued fake LP is detected by the verifier.

Non–Transferability: If a malicious user wants to submit a LP which has been generated for another prover P, the verifier will find it because P's signature is on rs which does not match with the attacker's ID. Note that the attacker can not see and manipulate the LP's contents since it has been encrypted using the verifier's public key. Even if he knows the P's ID and wants to impersonate P by submitting his request using a new m_1, he must forge the P's signature which is unlikely without having the P's private key. Moreover, the presence of $time$ in m_5 makes it impossible for him to use this LP later.

Resistance to Mafia Frauds: In this attack, an adversary tries to convince an honest witness that an honest prover is in the vicinity of the witness while he/she is not really (readers can refer to [13] and [15] for detailed information about *Mafia Frauds*). We assume an adversary A is going to perform a *Mafia Fraud* on a remote prover P and witness W who are both honest. We model A with a witness $\bar{W}$ and a prover $\bar{P}$. The time–limited process performed in stages 1.d and 1.e prevents $\bar{P}$ from sending rs to $\bar{W}$ for obtaining

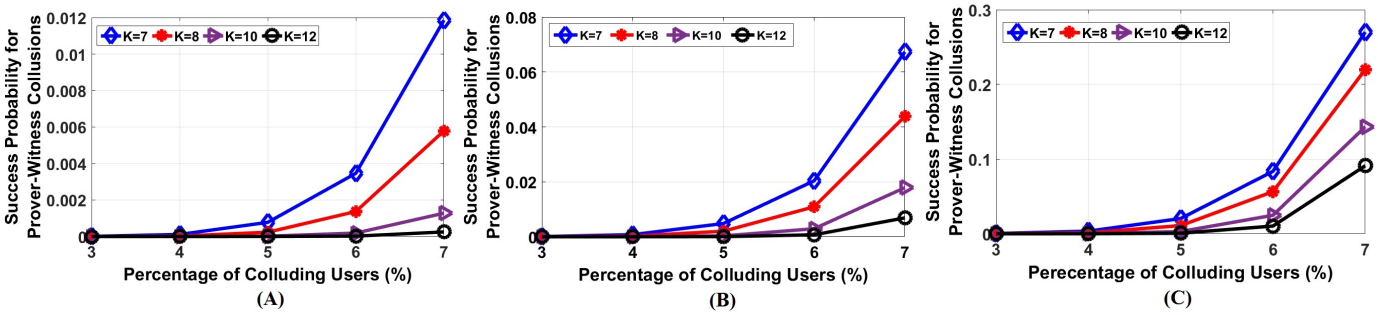

Fig. 3: The success probability of Prover–Witness collusions. (A) $\beta = 40\%$ (B) $\beta = 60\%$ and (C) $\beta = 80\%$

the P's signature on it because there is not much time to do so. If this process takes longer than T, the witness will send another rs which invalidates the previous rs. Thus, the attack is defeated.

Prover & Witness Location Privacy: Since all the messages that contain the prover's and witness' ID are encrypted with the verifier's public key, they can only be seen by the verifier. Moreover, we have employed the *sign–then–encrypt* model to generate SPARSE messages. This makes it infeasible for a curious entity or an eavesdropper to check the prover's or witness signature with the public key of all the users and find their identity.

Resistance to Terrorist Frauds (Prover–Prover collusions): In this attack, a remote malicious prover colludes with an adversary who is close to an honest witness to convince the witness that he/she is in its vicinity. Now, imagine an adversary A that is close to an honest witness W wants to collude with a remote dishonest prover P and answers to W's challenges on behalf of P. In this case, A must send rs to P to sign and encrypt it and then sends it back to A for submission with W. However, there is not enough time for them to do so because the validity of this rs is only for a short period T. After this time, the witness will broadcast a new rs and reject all the messages m_4 which have the previous rs. Thus, SPARSE is resistant to this type of attacks.

Resistance to Prover–Witness Collusions: Upon receiving a specific prover's request, it is the verifier that selects some witnesses to generate LPs for him/her. Later, in the Location Claim & Verification phase, the verifier rejects any LPs generated by witnesses not selected by the verifier. Thus, the prover is not permitted to collect a LP from any witness he/she likes. This makes it very difficult for a malicious prover to set up a successful *Prover–Witness* collusion. In this case, he has to increase the size of his collusion group to improve his chances of winning. In other words, he must collect at least K_T non–null LP to become successful. More precisely, if there are at least K_T colluding witnesses among the K selected witnesses, the attack will succeed. However, we show that this happens with a negligible probability. For this reason, suppose the malicious prover is colluding with K_C dishonest witnesses which are located at his desired location L. We assume N is the number of all witnesses present at L (including the dishonest witnesses) and x is the number of the colluding witnesses who are selected by the verifier to generate LP. Obviously, for $K_C < K_T$ we have $P_s = 0$, where P_s is the attack success probability. For $K_T \leq K_C < K$ we have:

$$\begin{aligned} P_s &= P(x \geq K_T) \\ &= P(x = K_T) + P(x = K_T + 1) + \ldots + P(x = K_C) \\ &= \sum_{j=K_T}^{K_C} P(x = j) = \frac{\sum_{j=T}^{K_C} \binom{K_C}{j}\binom{N-K_C}{K-j}}{\binom{N}{K}} \end{aligned}$$

Note that the malicious prover must collude with the witnesses who are physically present at L. This makes it too difficult to have a large K_C, specifically, for the applications where performing a large size collusion is too expensive. However, we assume he can select $K_C > K$. In this case, if $K_T = K$ is selected by the system, we have:

$$P_s = P(x = K_T) = \frac{\binom{K_C}{K}}{\binom{N}{K}} = \frac{K_C!(N-K)!}{N!(K_C-K)!}$$

Simulation results show that P_s is negligible if system parameters are carefully chosen (see section V for more details). Specifically, for large values of N, the attack is defeated with a high probability. Note that if the service provider creates enough incentives for the witnesses to collaborate with the system, we will have a large N. Therefore, SPARSE can significantly reduce the success probability of these collusions.

V. Performance Evaluation

In this section we evaluate the performance of SPARSE against *Prover–Witness* collusions. We adopt the same configuration with which STAMP [8] performance has been evaluated. Total number of users is set to 1000 and we suppose an average of 5% of these users are present at each location. Moreover, the threshold $K_T = K$ is adopted which means no null LP is accepted by the verifier. It is also assumed that the malicious prover sets up a collusion group of size K_C which is varied from 1% to 7% of all the users. An aggregation rate β is allocated to each collusion group which represents the percentage of colluding users who are located at the desired location during the time at which the attack is performed. Thus, $\beta = 0.7$ means that 70% of colluding users are present at the given location during the attack.

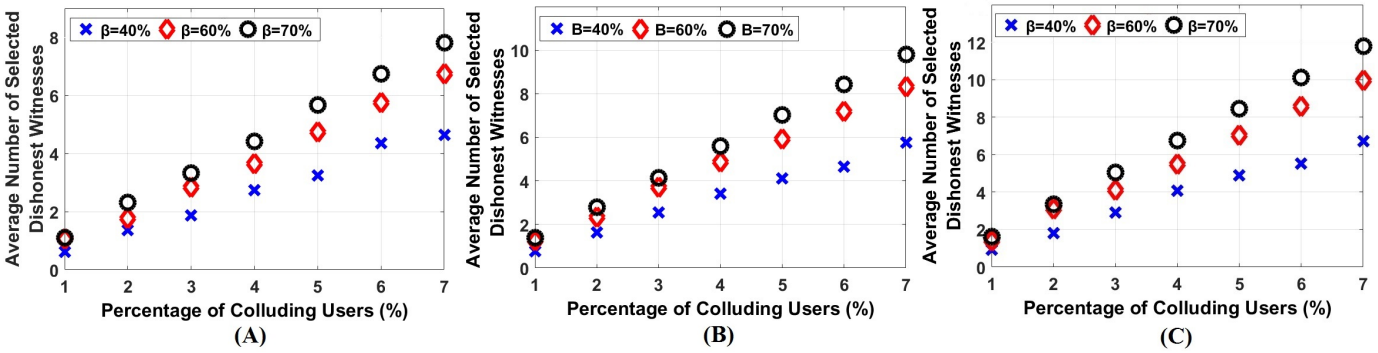

Fig. 4: The average number of colluding witnesses that are selected by the verifier for (A) $K = 8$ (B) $K = 10$ and (C) $K = 12$

Fig. 3 shows the success probability of *Prover–Witness* collusions for different values of β and K. As you see, the attack has a maximum success probability of 0.012 and 0.07 for $\beta = 40\%$ and 60% respectively when 7% of users collude with the malicious prover. This means that the system can prevent *Prover–Witness* collusions with the minimum success rates 0.988 and 0.993 for the mentioned situations. Even if 80% of colluding users are located at the intended location (i.e. $\beta = 80\%$) and the malicious prover colludes with 5% of users, the system prevents the attack with the success rates 0.98, 0.99, 0.997, and 0.999 for $K = 7$, 8, 10, and 12, respectively while in STAMP the maximum success rates 0.95, 0.92 and 0.65 have been achieved for different collusion tendencies when 5% of users collude. In fact, STAMP has a relatively poor performance against provers with a low collusion tendency. This is because they decide on the users' LP transaction history. Thus, for the malicious provers who have a diverse transaction history, STAMP does not offer a reliable performance (e.g. with a collusion tendency 0.2, STAMP achieves a collusion detection rate 0.65 when 5% of users collude with the malicious prover). However, SPARSE reaches the prevention success rates better than 0.98 regardless of the prover's past LP transactions. You can also see in Fig. 4 the average number of colluding witnesses who are selected by the verifier for different values of K and β. As you see, the number of selected dishonest witnesses are less than K which means that even for $\beta = 70\%$ and with 6% colluding users, the scheme is resistant to these collusions if K_T is chosen close to K.

VI. Conclusion

In this paper we have proposed SPARSE, a distributed location proof system for mobile users. The main distinguishing characteristic of the proposed system is that it provides a solution for the *Terrorist Fraud* and *Prover–Witness* collusions, the two issues from which the current distributed location proof systems suffer. Moreover, we have not employed the traditional distance bounding protocols. This not only results in fast location proof generation by the witnesses (because they become independent of the length of users' private key), but also provides an easy–to–implement system architecture. The performed security analysis and simulations show that SPARSE provides the fundamental security and privacy properties and prevents the *Prover–Witness* Collusion with success rates better than 0.98.

References

[1] M. L. Damiani, "Location privacy models in mobile applications: conceptual view and research directions", *GeoInformatica*, vol. 18, no. 4, pp. 819–842, Oct. 2014.

[2] H. Liu, X. Li, H. Li, J. Ma and X. Ma, "Spatiotemporal Correlation–Aware Dummy-Based Privacy Protection Scheme for Location–Based Services", in *IEEE INFOCOM*, 2017.

[3] M. R. Nosouhi, V. H. Pham, S. Yu, Y. Xiang, and M. Warren, "A Hybrid Location Privacy Protection Scheme in Big Data Environment", *IEEE GLOBECOM*, 2017.

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

[5] Z. Zhang, L. Zhou, X. Zhao, G. Wang, Y. Su, M. Metzger, H. Zheng, and B. Y. Zhao, "On the Validity of Geosocial Mobility Traces", *ACM Workshop on Hot Topics in Networks (HotNets)*, 2013.

[6] C. Javali, G. Revadigar, K. B. Rasmussen, W. Hu, S. Jha, "I Am Alice, I Was in Wonderland: Secure Location Proof Generation and Verification Protocol", *IEEE 41st Local Computer Networks Conference*, 2016.

[7] W. Luo and U. Hengartner, "VeriPlace: A privacy–aware location proof architecture", In *Proc. ACM GIS*, pp. 23–32, 2010.

[8] X. Wang, A. Pande, J. Zhu, P. Mohapatra, "STAMP: Enabling Privacy–Preserving Location Proofs for Mobile Users", *IEEE/ACM Transactions on Networking*, vol. 24, no. 6, pp. 3276–3289, Dec 2016.

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

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

[11] M. Talasila, R. Curtmola, and C. Borcea, "Link: Location verification through immediate neighbors knowledge", *Springer, LNICST 73*, 2012.

[12] S. Gambs, M. O. Killijian, M. Roy, and M. Traore, "PROPS: A Privacy–preserving location Proof System", *IEEE 33rd International Symposium on Reliable Distributed Systems*, 2014.

[13] L. Bussard and W. Bagga, "Distance–bounding proof of knowledge to avoid real–time attacks", In *Security and Privacy in the Age of Ubiquitous Computing*, New York, NY, USA: Springer, 2005.

[14] G. Avoine, C. Lauradoux, and B. Martin, "How Secret–sharing can Defeat Terrorist Fraud", In *Proceedings of the 4th ACM Conference on Wireless Network Security*, WiSec'11, Hamburg, Germany, June 2011.

[15] A. Bay, I. Boureanu, A. Mitrokotsa, I. Spulber, S. Vaudenay, "The Bussard–Bagga and Other Distance–Bounding Protocols under Attacks", *Inscrypt*, 2012.

[16] I. Boureanu, S. Vaudenay, "Challenges in distance bounding", *IEEE Security & Privacy*, 13 (1), pp. 41–48, 2015.

[17] I. Boureanu, A. Mitrokotsa, S. Vaudenay, "Practical and provably secure distance bounding", *Journal of Computer Security*, pp. 229–257, 2015.

[18] K. B. Rasmussen and S. C. Apkun, "Realization of RF Distance Bounding", in *Proceedings of the USENIX Conference on Security*, 2010.