Decentralised location proof system

February 29, 2016

Abstract

1 Introduction

2 Previous work

Location proof systems are expected to be accurate and tamper-proof. For this reason, existing solutions have chosen to use a central authority to issue proofs [1, 2, 3].

Hardware techniques [1] operate by supplementing existing WiFi access points with *femtocells* (small cellular antennae that connect to the mobile carrier via the Internet). Location verification over the internet is made possible by determining which femtocell a mobile node is connected to as it transfers data via Wi-Fi. This solution requires investment in additional hardware to supplement existing WiFi access points. It also requires access to mobile providers user database to identify users locations.

Other proof systems [2] deploy software on Wi-Fi access points. Access points taking part in the location proof network become part of a 'group' with a shared group signature. Mobile nodes can request a location proof from an access point, who signs it with the group signature. Access points provide location proofs to nodes without ever learning the node's identity, thus protecting user's privacy. However, due to the systems reliance on the group signature structure, access points become a target for attack. Compromising an access point would allow an attacker to create false location proofs.

[2] considers a number of threats in their architecture:

- **Dishonest users.** A dishonest user tries to obtain location proofs that certify her presence at some place at a particular time even if she was not there. Dishonest users may achieve this goal by colluding with malicious intruders.
- **Malicious intruders.** A malicious intruder is not interested in obtaining location proofs for her own use but offers to help other users to get location proofs on their behalf in exchange for other benefits like money.
- **Curious APs and applications.** A curious AP tries to learn a users identity while the user is acquiring a location proof from the AP. Similarly, a curious application tries to learn more location information from a location proof than it really needs.
- **Malicious applications.** A malicious application obtains location proofs from its users and then tries to take advantage of these proofs to get unauthorised access to other applications.
- **Active and passive eavesdroppers.** An eavesdropper records and maybe modifies communication between users, proof- ssuers, or applications.

Distributed P2P location proof systems also exist [3]. A central authority is used and acts as a certificate authority, issuing a unique ID to each person using the system. Unique identity is proved by users by providing SSN, drivers licence, or passport documents. This removes the threat of a Sybil attack [5], but makes the central authority a clear target for attack. The system uses three parties to provide location proofs; a mobile user who requests a proof, a fixed location authority, and a mobile user who acts as a witness. The system becomes vulnerable in the case where all three users collude, but is otherwise quite an interesting approach.

[3] also outlines a number of possible attacks on their system:

- **False presence:** A malicious user can create a fake location proof on his own, without being physically present at the location. The fake proof is supposed to resemble an actual proof, which the user could have actually collected from a valid location authority.

- **False timestamping (backdating, future dating):** In a backdating attack, the user and the location authority colludes to create a proof for a past time. Conversely, in future dating, the location authority and a user colludes to generate a proof with a future timestamp.
- **Implication:** A location authority and/or a witnesses can falsely accuse a user of his presence at a certain location. In this case, the malicious location authority and witness colludes to generate a false proof of presence for the user.
- **False assertion:** A user can collude with a witness, and generate a falsely asserted location proof. The truth value in such a fake proof is reinstated with the assertion received from the other user.
- **Denial of presence:** A user can visit a location and at a later time, deny his presence at that location. In such a case, the user actually denies the validity of a certain location proof that has been been generated upon his presence at that particular location.
- **Proof switching:** The user is expected to have full access to all storage facilities on his mobile device. Hence, the user utilizes the legitimate proof and manipulates the information to create a false proof for a different location.
- **Relay attack:** A user can use a proxy to relay the requests and collect a location proof. Alternatively, a location authority can maliciously relay assertion requests with the witness not being present at the site.
- **Sybil attack:** A Sybil attack occurs when a single user generates multiple presence and identities [29]. A user can launch a Sybil attack by generating multiple identities representing a user and a witness and provide false endorsements for location proofs.
- **Denial of witnesss presence:** At the time of proof verification, the user can claim the absence of witnesses at the site or falsely claim an assertion to be counterfeit. The user and the location authority may also collude and claim the non-availability of witnesses.
- **Privacy violation:** An attacker may capture an asserted location proof generated for a user, and discover the identity of the user and/or the witness.

'OTIT' [4], a model for designing secure location provenance, can be used to compare existing location proof systems. This model defines the following requirements necessary for designing any secure location provenance scheme: Chronological, Order Preserving, Vereifiable, Tamper Evident, Privacy Preserved, Selective In-Sequence Privacy, Privacy Protected Chronology, and Convenience & Derivability.

3 Decentralised location proof system

This location proof system is comprised of a network of mobile nodes, capable of anonymously providing and receiving an alibi to/from other nodes on the network.

When two nodes are in close physical proximity, they initiate a transaction with each other over a short range, ad-hoc network such as bluetooth. To create this transaction, they must agree on their current GPS coordinates and the current time. Once both nodes reach agreement on these parameters, they each create an encrypted transaction that claims their current time and location. This transaction must also contain references to both the alibi's identity, and the index of the nodes most recent transaction. Both nodes then publish their own transaction, indexed using their identity, onto a public append-only bulletin board/blockchain.

A node can give verifiers permission to verify its location. The node will give the verifier access to the keys used to encrypt each of the nodes n most recent transactions. After finding and decrypting the transactions, the 3rd party will decide based whether or not to accept the node's claimed location as truth.

3.1 Identities

In order to preserve user's privacy, a user will only ever use an identity for one transaction, before generating a new one. This prevents malicious users from watching the public blockchain for a known identity and tracking it. However, to prevent identity theft, it is important for the verifier to be able to prove that a node was the original creator of each identity. This is achieved using a public/private key pair for each node.

When a node is created, it generates (or is provided with?) a certificate (or key pair). To maintain anonymoty, the node will use the public key to

encrypt some nonces to create identities, and use these identify itself in a transaction. During the verification stage, the node will provide the verifier with its public key, along with the list of the nonces used to generate its n most recent transactions. The verifier calculates the node's identities using the public key and nonces, and retrieves the relevant transactions from the public blockchain. The verifier will also ensure that the node owns the private key associated with the provided public key, to prevent identity theft.

3.2 Transactions

During a transaction between two nodes, a number of different items must be shared and calculated. The first step in a transaction is for both nodes to agree upon a GPS location and current time. Nodes likely won't have the exact same GPS coordinates or time, but once the values don;t differ by more than some constant ϵ, the nodes will continue with the transaction.

After nodes agree upon the parameters of the transaction, they exchange their key packets with each other. A key packet is a list of the nodes m most recent identities, along with the keys needed to decrypt them. This allows the verifier to check the credibility of each alibi the node seeking verification has collected.

The following transaction data will be published onto the blockchain by node A:

$T_{An} = ID_{An}|K_{A1n}(ID_{An-1})|K_{A2n}(ts, loc, ID_{An}, ID_{An-1}, KP_{Bm})$

$\mathbf{ID_{An}}$: The current ID of node A.

$\mathbf{K_{A1n}(ID_{An-1})}$: Node A's previous identity. This creates a backward link between A's transactions, used to preserve the "privacy protected chronology" property of OTIT [4]. This is encrypted with a symmetric key K_{A1n} to stop curious users monitoring the blockchain to track other users.

$\mathbf{K_{A2n}}$: Node A's second current symmetric key at transaction n, used to encrypt the body of the transaction. This is separate from K_{A1n} to preserve "privacy protected chronology" and "selective in-sequence privacy" [4].

ts: The agreed timestamp.

loc: The agreed GPS coordinates.

$\mathbf{ID_{An}}$and$\mathbf{ID_(An-1)}$: Node A's current identity and most recent identity respectively, encrypted with K_{An}. These are used to prevent a possible identity theft attack whereby a malicious man-in-the-middle may try to alter the ID attached to the transaction.

$\mathbf{KP_{Bm}}$: Node B's key packet at transaction m (relative to B). This contains B's n most recent transaction keys, so any verifier that node A requests will be able to validate the credibility of alibi B by decrypting his transactions.

The key packet is more than simply a list of keys that B has used to encrypt his transactions; in order to preserve the "privacy preserved" property of OTIT [4], a user must be able to choose transactions that he doesn't want others to be able to decrypt. For this reason, the transaction backwards-chaining may need to be broken. For example, if node B doesn't want to reveal transaction T_{Bm-3}, he would reveal the following key packet:

$$KP_{Bm} = (\{ID_{Bm}, K_{Bm}\}, K_{Bm-1}, K_{Bm-2}, \{ID_{Bm-4}, K_{Bm-4}\}, K_{Bm-5}, \ldots, K_{Bm-n})$$

In this case, ID_{Bm-4} is needed along with K_{Bm-4} so the verifier knows at which ID the privacy-protected chain will become unprotected, and can be decrypted.

3.3 Verification

The verification stage is the most algorithmically complex part of the system. It is performed by the verifier. Given the keys for a node's n most recent transactions, the verifier will walk chronologically backwards along the public location proof chain. At each transaction encountered, the search will fork along the proof history path of both parties involved in that transaction, and continue until all available transactions have been exhausted.

The goal of the verification stage is to either accept or reject the location that the user is claiming. There are a huge number of factors involved in reaching a conclusion from the data on the blockchain; some of these factors will likely be unique to certain verifiers, and kept secret to improve reliability. Some simple example factors may include:

- **Alibi credibility:** if each of the nodes alibis have little or no alibis themselves, then the node is likely attempting a poorly-constructed Sybil attack, and the verifier will reject his verification request.
- **Alibi reuse:** if the verifier can prove that alibis are being reused frequently, then it will reject the verification request.
  - Due to transaction anonymity, the verifier may not always be able to tell if two alibi's are the same person or not. e.g. node A could use node B as an alibi, then wait for B to complete n more transactions, then use node B again.
  - This is not a viable attack on the verification system. Depending on the size of n, enough time may pass between both uses of node B as an alibi that the transactions are no longer relevant.

3.4 Sybil attack

This system is vulnerable to a Sybil attack, where a single physical node may create multiple *pseudoidentities* [5]. A suitably powerful node, or motivated attacker, could create enough pseudonyms to create a subnetwork of nodes, each creating malicious transactions with each other. This would allow an attacker to falsify a believable location proof.

It has been proven that a central certificate authority is the only method of preventing a Sybil attack [5]. Other measures, such as web of trust, increase the difficulty involved in performing an attack, but the system remains vulnerable to a motivated attacker.

4 Results

5 Conclusions

References

[1] J. Brassil, P.K. Manadhata, "Verifying the Location of a Mobile Device User", Proc. of MobiSec 2012, June 2012.

[2] Luo, W., Hengartner, U., "Proving your Location without giving up your Privacy", Proceedings of the Eleventh Workshop on Mobile Computing Systems & Applications, HotMobile 2010, Annapolis, Maryland, February 22 - 23, pp. 712. ACM, New York (2010)

[3] R. Khan, S. Zawoad, M. M. Haque, and R. Hasan, "'Who, When, and Where?' Location Proof Assertion for Mobile Devices", Proceedings of the 28th Annual IFIP WG 11.3 Working Conference on Data and Applications Security and Privacy, ser. DBSec. IFIP, July 2014.

[4] Khan, R., Zawoad, S., Haque, M., Hasan, R. "OTIT: Towards secure provenance modeling for location proofs", Proc. of ASIACCS. ACM (2014)

[5] Douceur, J.R., "The sybil attack", Druschel, P., Kaashoek, M.F., Rowstron, A. (eds.) IPTPS 2002. LNCS, vol. 2429, pp. 251260, Springer, Heidelberg (2002)