STAMP: Enabling Privacy-Preserving Location Proofs for Mobile Users

Abstract:

Location-based services are quickly becoming immensely popular. In addition to services based on users' current location, many potential services rely on users' location history, or their spatial-temporal provenance. Malicious users may lie about their spatial-temporal provenance without a carefully designed security system for users to prove their past locations. In this paper, we present the Spatial-Temporal provenance Assurance with Mutual Proofs (STAMP) scheme. STAMP is designed for ad-hoc mobile users generating location proofs for each other in a distributed setting. However, it can easily accommodate trusted mobile users and wireless access points. STAMP ensures the integrity and non-transferability of the location proofs and protects users' privacy. A semi-trusted Certification Authority is used to distribute cryptographic keys as well as guard users against collusion by a light-weight entropy-based trust evaluation approach. Our prototype implementation on the Android platform shows that STAMP is low-cost in terms of computational and storage resources. Extensive simulation experiments show that our entropy-based trust model is able to achieve high collusion detection accuracy..
Existing system:

Existing schemes which require multiple trusted or semi-trusted third parties, STAMP requires only a single semi-trusted third party which can be embedded in a Certificate Authority (CA). We design our system with an objective of protecting users’ anonymity and location privacy. No parties other than verifiers could see both a user's identity and STP information (verifiers need both identity and STP information in order to perform verification and provide services). Users are given the flexibility to choose the location granularity level that is revealed to the verifier. We examine two types of collusion attacks: (1) A user who is at an intended location masquerades as another colluding user and obtains STP proofs for it. This attack has never been addressed in any existing STP proof schemes. (2) Colluding users mutually generate fake STP proofs for each other. There have been efforts to address this type of collusion. However, existing solutions suffer from high computational cost and low scalability. Particularly, the latter collusion scenario is in fact the challenging Terrorist Fraud attack, which is a critical issue for our targeted system, but none of the existing systems has addressed it. We integrate the Bussard-Bagga distance bounding protocol into STAMP to protect our scheme against this collusion attack. Collusion scenario (1) is hard to prevent without a trusted third party. To make our system resilient to this attack, we
propose an entropy-based trust model to detect the collusion scenario. We implemented STAMP on the Android platform and carried out extensive validation experiments. The experimental results show that STAMP requires low computational overhead.

Disadvantage:

✓ Solutions suffer from high computational cost and low scalability. Particularly, the latter collusion scenario is in fact the challenging Terrorist Fraud attack, which is a critical issue for our targeted system, but none of the existing systems has addressed it.
Proposed system:

In this paper, we propose an STP proof scheme named Spatial-Temporal provenance Assurance with Mutual Proofs (STAMP). STAMP aims at ensuring the integrity and non-transferability of the STP proofs, with the capability of
protecting users' privacy. Most of the existing STP proof schemes rely on wireless infrastructure (e.g., WiFi APs) to create proofs for mobile users. However, it may not be feasible for all types of applications, e.g., STP proofs for the green commuting and battlefield examples certainly cannot be obtained from wireless APs. To target a wider range of applications, STAMP is based on a distributed architecture. Co-located mobile devices mutually generate and endorse STP proofs for each other, while at the same time it does not eliminate the possibility of utilizing wireless infrastructures as more trusted proof generation sources. In addition, in contrast to most of the existing schemes which require multiple trusted or semi-trusted third parties, STAMP requires only a single semi-trusted third party which can be embedded in a Certificate Authority (CA). We design our system with an objective of protecting users' anonymity and location privacy. No parties other than verifiers could see both a user's identity and STP information (verifiers need both identity and STP information in order to perform verification and provide services). Users are given the flexibility to choose the location granularity level that is revealed to the verifier.

Advantages:

✓ Our security analysis shows that STAMP achieves the security and privacy objectives.
Our implementation on Android smartphones indicates that low computational and storage resources are required to execute STAMP.

Extensive simulation results show that our trust model is able to attain a high balanced accuracy with appropriate choices of system parameters.

Conclusion:

In this paper we have presented STAMP, which aims at providing security and privacy assurance to mobile users' proofs for their past location visits. STAMP relies on mobile devices in vicinity to mutually generate location proofs or uses wireless APs to generate location proofs. Integrity and non-transferability of location proofs and location privacy of users are the main design goals of STAMP. We have specifically dealt with two collusion scenarios: P-P collusion and P-W collusion. To protect against P-P collusions, we integrated the Bussard-Bagga distance bounding protocol into the design of STAMP. To detect P-W collusion, we proposed an entropy-based trust model to evaluate the trust level of claims of the past location visits.

Hardware Specification:

- System: Pentium IV 2.4 GHz.
- Hard Disk: 40 GB.
The Master of IEEE Projects

- Floppy Drive : 44 Mb.
- Monitor : 15 VGA Colour.
- Mouse : Logitech
- Ram : 512 Mb.
- MOBILE : ANDROID

Software Specification :

- Coding Language : Java 1.7
- Tool Kit : Android 2.3 ABOVE
- IDE : Android Studio

Reference :


