TrustPos Model: Trusting in Mobile Users' Location

Javier Martín de Valmaseda, Georgia Ionescu and Michel Deriaz

University of Geneva, Institute of Services Science, 1227 Carouge, Switzerland

{Javier.MartinDeValmaseda, Georgia.Ionescu,
Michel.Deriaz}@unige.ch

Abstract. While social games based on geo-location are gaining popularity, determining the authenticity of the players' geo-position becomes a challenge, since there are ways to counterfeit it, quite accessible to everyone. We propose a solution based on global spatial and temporal observation of the players' interactions. In this paper we present TrustPos, a trust engine model that associates a trustworthiness factor to each player based on the context of the interactions with both the game and other players. The novelty of TrustPos is the fact that our model is based on an internal network of players linked through their interactions, as opposed to previous approaches that are strongly specialized to concrete domains as peer-to-peer networks and social recommenders, not adaptable to location trust concerns.

Keywords. Location, mobile, trust, social games.

1 Introduction

Nowadays, the interactions in social networks are increasingly linked to the user's location. Different games, applications and communities based on location capabilities are growing at a rapid pace. Let us consider as an example Foursquare [1], a well-known location-based social network that allows users to post their location at a venue ("check-in") and connect with friends. By January 2013, Foursquare acquired over 30 million users worldwide, with over 3 billion check-ins, and millions more every day. This shows how much success the location-based capabilities can bring to social applications. Games are becoming more and more social, and location features can only contribute to their success.

In order to bring innovations and to connect players to the real world, games are using location-based features. For instance Ingress [2], the new location-based massively multiplayer game developed by Google's Niantic Labs. In this game, location check-ins and interactions are necessary to unlock clues about what is going on, gather objects, work together and much more.

As more games integrate the geo-location capabilities of mobile devices, game providers need to trust the user's position, they need to be sure that users are really where they claim to be, in order to avoid cheaters and to keep the game interesting and competitive.

However, cheating over the current location is easy since it is the mobile device that gets the geo-position and sends it to a server. When searching for Ingress on the web, it is easy to find tutorials [3] about how to fake your location in the game.

Encryption is not a solution either, as it is easy to find where the position is obtained (GPS for instance), even in obfuscated code, and then a malevolent user can simply change the data before the encryption process. Nevertheless, a global spatial and temporal observation of the full system would allow us to find cheaters. For instance, a *user A*, interacting with a *user B* in London should not be able to interact with a *user C* in New York one hour later. At least one is cheating about his position. By looking further into the history of these three users, the chance of finding out who is cheating becomes higher.

This paper presents a system called TrustPos, which is a new trust engine model based on the idea of spatial messaging (defined by [4-5]). This idea is combined with a trust model where users, transparently, are able to create trust links between one another and with the system.

Position information given by a user is also rated, for instance it has more weight if it is confirmed by a trustworthy user or if the interaction has been done with a trusty source, like a NFC reader in a shop. The trust engine is able to determinate how trustworthy the user is and gives a rate to the interaction after evaluating it.

In section 2 we start by presenting different approaches made before in the location and trust field, and then we explain our chosen path. Section 3 presents an overview of our solution. In Section 4 the trust model and the trust engine is explained. Finally we give our conclusions and future work in section 5.

2 Related Work

There are several approaches in the field of certifying and proving users' location in mobile applications. For instance, Lenders et al. [6] propose a secure geo-tagging service. Their purpose is to be able to trust in the content produced by a mobile user. Towards this goal the content is tagged with a Data Location Time certificate. Later on, a user consumer can verify the original location and time of the information. The content can be verified by checking the signature of the certificate using the public key of the location/certificate authority. The problem with this method is that they trust, or assume that the way of obtaining the location is trustworthy. They propose several methods for ensuring the way of obtaining the location, however these methods are not enough. Cheating over the current location before the certification process is easy since it is the mobile device that gets its position and sends it to the server. Our engine takes into account this factor when analysing a user's position and should be able to detect if the user is potentially trying to cheat according to previous interactions.

Saroiu and Wolman [7] have an interesting approach for trusting the location, using what they called location proofs. A location proof is a piece of data that certifies a receiver to a geographical location. Basically they rely on Wi-Fi access points or cell towers to generate these location proofs. The geographical location of the access point

is embedded in each location proof, which is then transmitted to the mobile devices. Following this approach, Luo and Hengartner [8] propose what seems to be a better solution using a similar technique. Besides their own location proof architecture based on access points, they take care of privacy protection of the users and they provide a mechanism in order to deal with the cheating users.

Unfortunately, relying on external infrastructure is unaffordable in the world wide social games that we are targeting. This technique is limited to areas with this infrastructure already deployed.

There are several proposals to trusting in the field of sensor information. For instance [9-11] which present similar solutions based on a hardware device implanted in the mobile phone. These solutions use a dedicated hardware (called Trusted Platform Module hardware, TPM [12]). The main difference is that in Gilbert et al. [9] and Zhidong et al. [10] the target is a trustworthy mobile sensing platform integrated in the mobile phone while in Dua et al. [11] rely on signing the raw readings from the sensors. However these solutions are beyond the scope of our target application, a social game that will be played in several different devices that will not incorporate this specific hardware.

Several approaches have been made in the field of trust computations. These solutions are mostly oriented towards trust relations and calculations in social networks, recommender systems and peer-to-peer networks. From our point of view there are two ways of handling trust information: centralised and decentralized. The eBay site [13] is a good representation of a centralized system. In this case, each user has a global reputation that is calculated by the system, using the different rates given by other users. After each transaction, the buyer and the seller rate each other and the global reputation is automatically updated by the system.

On the other hand, the decentralised trust systems are a bit more complex. For instance, in a peer-to-peer network, peers rate each other and the reputation, of a peer is the sum of all the other peers' rate. As there is no global system or global values for each user, it is necessary to ask all the users about the reputation of a given one. These networks algorithms are widely used for recommendations in social networks, like movie recommenders.

Different surveys on Trust Computations [17-16] show that the majority of current algorithms have been proposed for special computing environment such as wireless networks, peer-to-peer systems and social recommenders. It will be complicated to use one of these approaches in a rapidly changing environment due to its complexity and specialisation.

A case of highly decentralized ad hoc networks is the Wireless Sensor Networks (WSNs). Over the past decade, several techniques like S. Capkun et al [14] and Joengmin Hwang et al [15] have been proposed for solving the positioning problem in wireless networks. J. G. Alfaro et al [16] proposes three different algorithms that enable WSNs to determine their location in presence of neighbour sensors that lie about their position. Additionally, these techniques aim at isolating the set of liars. All three of them are based on the radio signals submitted to neighbours, which allow them to compute the distance between different sets of nodes. As interesting and efficient this approach was proven to be, it cannot be used for our purposes. The most important

reason is the fact that in our case, the devices do not interact with one another. Their positions are shared through a server that holds the database with all the interactions.

Another problem when thinking about applying one of these algorithms to our case is that in most of the trust models there is no awareness of the context or the time.

However for our target, time and context are important, the users interact along the time, frequently, and the context is different. For instance, having a player conquering territories physically at the same time at Tokyo and Paris is not possible. The circumstances of the event are important to our trust model; therefore, we believe that a trust model similar to the idea of FoxyTag [4] could be relevant for our objective.

FoxyTag is a speed camera warning system based on special geo-located tags (speed cameras posted by users while driving) and a trust engine to self-maintain the tags database. FoxyTag allows drivers to easily signal a speed camera or to signal that a former one has been removed. Other users driving in areas with tags receive the alert about the speed camera. The main advantage of FoxyTag is that it does not require human checks to decide the trustworthiness of the posted tags because the system uses a computational trust engine to automatically make this decision.

Our idea is to develop a trust engine based on this model in which each user action is recorded and observed. With each action trust links are created between users. We can have a global view of one specific player asking the community about the trustworthiness of this specific user. Similar to human community, when we want to know if someone is trusty, we ask other people whom we trust, or people who already have interact with this user.

3 Solution Overview

As shown in the introduction, location features are going to be part of the games, at least a distinctive feature. Location capabilities bring opportunities to innovate and to gather players in the real world. For instance, the games are going to allow the players to create their own location-based territories, and to fight against others in the same location. For this purpose, we need to trust the position given by the users to keep the game interesting and competitive.

Our proposed solution is to develop a trust engine in order to manage the trustworthiness of the different users' actions when posting the position in the game. Based on these interactions, we make a global spatial and temporal observation of the users.

Therefore, users can create transparent trust links with the system and other users and rate the different interactions. With these different values, it is possible to achieve a global view of the different users and interactions. Furthermore it is possible to make the proper decisions upon the trustworthiness of each user.

For better understanding of our purpose, we further describe the target game and two main usage scenarios.

3.1 Target Application

The main target game that we are pursuing is a location-based game. In this game, the players have the possibility of play and interact using the location features of theirs mobile phones. For instance, players will create their own location-based territories, and they will fight against others in the same location. For this purpose, and for keeping the game interesting and competitive, we need to trust the position given by the users.

When two users are playing together at the same location, they are mutually agreeing of being there. Therefore if the trust engine detects that the user is cheating, the trust that the system and the other user has in this user will be decreased.

We are working closely with EverdreamSoft a game company that produces a successful worldwide card game – Moonga [20]. Moonga is a multi player mobile game running on both Android and iOS. Based on 5 cards the players can build creative strategies to dominate their opponents' cards. In the next main release EverdreamSoft is planning to integrate our engine in order to trust the positions given by the players. Their goal is to make the game more social by merging it with real world interactions. Players will meet to play in real world for conquering physical territories, for exchanging cards and for card fights.

3.2 Main Usage Scenarios

In order to keep things simple, we have defined two basic scenarios in our target game. The most basic one is when one user is trying to create a territory in the game using their geo-position. From the point of view of our trust engine, this is seen as a user-system interaction.

The second possible scenario is when one user plays against another user in the same physical location. This is seen as an interaction between two users. Bellow, we give a more detailed explanation for each scenario.

Scenario1. Alice is a user who wants to create a new territory in London. In the game options, she selects the corresponding create territory option at this position. She is now transparently interacting with the trust engine. The client application (mobile phone) sends a *post message* to the system (server) with the pertinent information about this interaction.

With the information contained in the message, the history of actions of Alice (position, time and other parameters) is updated on the server.

The server returns the trust value for Alice or whether it is possible to be where she claims to be (it is not possible to be in London and within 30 min in Paris). These calculations/observations can be made by taking into account the current context, the previous ones and the trust values that other users and the system have for this user.

Scenario2. Alice wants to play/fight with another user in the same location/area; after selecting the appropriate option in the game, Alice will get a list of the nearest

users to interact with. When Alice selects another user within the list (Bob), and Bob accepts the request, both users send a *review request* to the server. The *review request* means that both users' history is updated with the context of this last interaction (time, location, opponent, etc.).

Due to this interaction each user is transparently rating the other user and updating their local trust value (their opinion) for this user. For instance, if Alice is fighting with Bob in the same physical location, both users should have the same position, and the positions should be physically possible compared to their last one. If every parameter of the context is as expected, each user will transparently update the trust on their opponent, if something is wrong, the trust on the opponent will be decreased.

With this review process we keep track of the last interactions for each user and a local trust value for all the users.

It is important to notice that the whole process is completely transparent to the player. Users cannot see the different trust value or the rates given by the trust engine. However, when accepting to play against other user posting the location, the users are claiming to be at the same place. Typically the community of trustworthy users will not play against not trustworthy users. Therefore, we expect to have an isolated group of cheaters and a big community of fair players, as we can see nowadays in our target game Moonga.

3.3 System Definitions

User Interaction. When one or two users interact with the game, the context of the interaction (position and time) is sent to our trust engine. The trust engine calculates the trust value of the user, rates the interaction and updates the tag (the user information and history). Figure 1 shows two users interacting with our system.

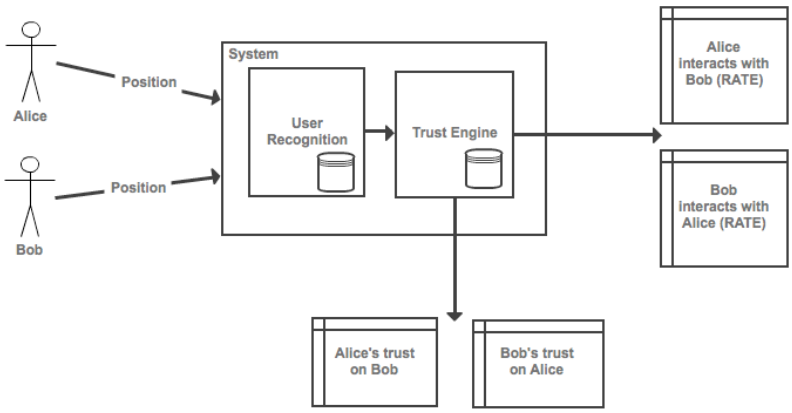

Fig. 1. Users interacting with the system

User. We keep track of the context (time and location) of the last interactions for each user to be able to evaluate his trustworthiness.

A user in the system has his own trust table. Each user keeps a local trust value (opinion) based on former interactions with other users. We consider the system as a special user who also has its own local trust table for all the interactions that the users do. This mechanism is similar to a human community, meaning that each individual keeps an opinion of other individuals based on the previous interactions.

Trust Value. In our model we use two important parameters:

- Local trust: Direct trust that one user has in another user, based only on former direct interactions between them.
- Global trust: The average of the local trust value of all the users in the system when asking for a given user.

MinTrustValue. The minimum trust a user can have. It should be big enough, bigger than the *maxTrustValue* (in absolute). It is difficult to become trusted after a few positive actions but easier to become untrusted after the same amount of negative actions. Initially the *minTrustValue* has been set to -70.

MaxTrustValue. The maximum trust a user can have. It should not be too high in order to avoid trusty users with a high trust value become malevolent. Initially its value is set to 5.

Rate. The rate of the interaction is a value between 0 and *maxRate* given by the trust engine to each user action. This value is helpful to evaluate the interaction (*user-system* or *userA-userB*) giving a rate or trust value to it. In this value the trust engine takes into account the different parameters involved (interactions made from a trusty source, same position as other user, etc). For instance, the rate is bigger if both users have the same position or if the previous interactions were positive, etc.

History. The history keeps track of the last *N* interactions with the system or other users sorted in reverse chronological order (recent events are in the top, the old ones are at the bottom). This field stores information related to the interaction, the different users, the positions and the rate given by the trust engine among others.

4 Trust Engine Model

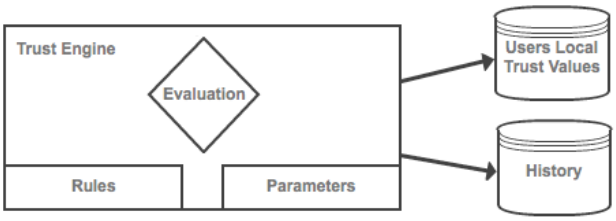

Fig. 2. Trust Engine Architecture

4.1 Trust Engine Architecture

The architecture of the trust engine's model has been kept as simple as possible. As shown in Figure 2 the Trust Engine consists in a set of parameters and rules and two databases with the entire local trust values and the history of the different users.

The rules and parameters are based on the different context and actions. For instance, if the position is possible when comparing to the history of positions, the trust calculation will be positive. The parameters reflect the values necessaries to make the decisions, such as the limit when a user becomes untrustworthy.

The evaluation component is in charge of handling the interaction, using the rules, parameters and trust model to make the proper decision and update the data and outputs.

4.2 Trust Model

Our model seeks to be close to the human concept of trust. In human communities, when an individual wants to know if another individual is trustworthy, the usual way of proceeding is to ask his friends or other individuals that have already interacted with this particular individual. We adopted a similar strategy; for each user, we store a trust value of other users, based on former interactions. We call this value *local trust value*, which in essence is the opinion that a user has on another user based on previous interactions.

So, when we want to know about the trustworthiness of a user, we only need to ask all the other users with whom he has interacted, including the system. We call this value *global trust value*.

We want a model similar to the human way of thinking in which recent events are in mind. It is difficult to become truly trustworthy. It is necessary to have several positive consecutive actions to achieve it. However once someone is trustworthy after a few disappointments it is easy to be untrustworthy.

In our model, trust increases linearly towards a limit and decreases exponentially. This means that several consecutive positive actions are needed in order to increase the trust value and become trustworthy. On the other hand, as the decrement is expo-

nential, little mistakes at the beginning are forgivable, but if the consequences of the actions are negative, the value decreases exponentially and it is very difficult to become trustworthy again. In our point of view this behaviours, reflect the human way of thinking in managing trust. Perhaps misunderstanding how the game works or small cheats at the beginning are forgivable but if the behaviour persists, the trust on this individual will decrease quickly and he will never be trustworthy again.

As an example, consider a situation where to co-workers go to buy coffee every day. It is always the same person who pays but he is reimbursed by his colleague when they return to the office.

The one who pays, trusts that his partner will pay him the coffee later. If one day his partner does not pay, as he trusts his partner, he will think that is a mistake or misunderstanding. But if the behaviour of the partner continues like that, at the end, he will stop paying his colleague's coffee because he will not trust him anymore.

It is important to take into account the fact that we need to fix a maximum and a minimum trust value that a user can achieve. Typically, the maximum value should not be too high in order to avoid that a user which has been acting properly for a long time, suddenly becomes malevolent and tries to subvert the system. However for the minimum value, we should set a big negative value so a user that has been cheating during all the interactions should not be easily trustworthy again.

As we previously commented, the trust engine rates each interaction, taking into account the context and the different parameters of the interaction. For instance it has more weight if the interaction is with a trustworthy user or if the interaction has been done with a trusty source, like a NFC reader in a shop or if the last interactions were also positive.

4.3 Trust Engine Model

In Figure 3 we can see the basic behaviour of the trust engine. The process starts with an interaction, that could be one user alone posting his position or two users who want to play against each other. All the important contextual information is transmitted to the trust engine; after evaluating the different inputs and the previous history the trust engine updates the corresponding trust values and rates the interaction following specific rules and decisions of this model.

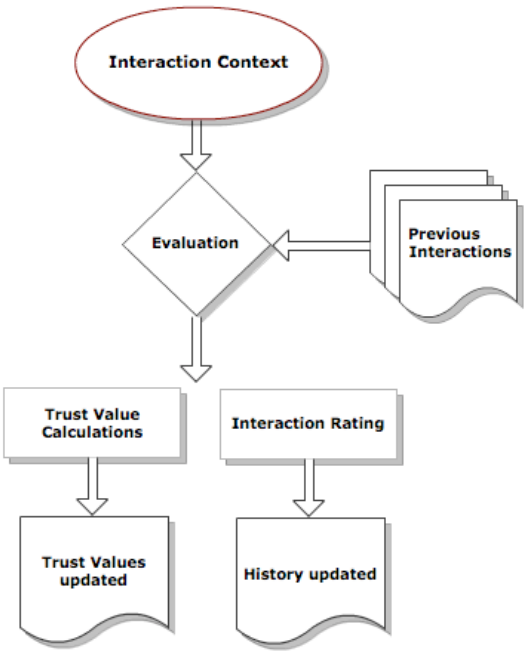

Fig. 3. Trust Engine Model

5 Conclusions and Future Work

Mobile technologies and smartphones with location capabilities are widely popular nowadays as well as applications and games with location-based features. For social location-based applications, like games with physical features based on users' mobile location it is important to trust the position sent by the users. All the similar solutions that we have found are basically focused on using external infrastructures, which is unconceivable for a social game spread all over the world.

In this paper, we have presented our approach to solve the problem of trusting the position given by a user's mobile phone in a game environment. We have explained our model of trust engine, which is able to generate and manage transparent trust links between the users in order to determinate their trustworthiness. This trust model uses the user history, and interactions to rate each interaction in the game and calculate a local trust value for each user and a rate for each interaction. The trust engine, combined with the appropriate security measures when posting the position in the game, is a powerful tool for avoiding cheaters.

TrusPos is a very good first filter, which will be combined with the appropriate security methods to obtain the position. Additionally, a management tool will be provided, so a human operator can see the last suspicious interactions, trust updates, and make the decision whether to ban or put in quarantine suspicious users and actions.

After a review of the current related approaches, we can conclude that there is not yet a popular solution to this problem and that our model could be an interesting solution. We have presented an overview of our model, our trust model, the behaviour and architecture.

Currently, this model is in deployment and testing phase. We have started the development of the trust engine and we are constantly working on and improving it. In the future we will finish all the functionalities presented. Simulations will be made to test the proper behaviour of our trust engine. A simulator is under development in order to help us test different situations in a location-based game.

Acknowledgments. The work of this paper was supported by the Commission for Technology and Innovation - CTI under contract no 14249.1.

6 References

1. Foursquare, https://foursquare.com/about Accessed on May 2013
2. Ingress. The game. NianticLabs@Google. http://www.ingress.com Accessed on May 2013
3. Ingress mock your location, http://decodeingress.me/2012/11/23/ingress-chea-mock-your-location Accessed on May 2013
4. Deriaz, M. and Seigneur, J-M.: Trust and Security in Spatial Messaging: FoxyTag, the Speed Camera Case Study. In Proceedings of the 3rd International Conference on Privacy, Security and Trust, ACM, 2006.
5. Deriaz, M.: Trust without Truth In IFIP International Federation for Information Processing, Volume 238, Trust Management, eds. Etalle, s., Marsch, S., (Boston: Springer), pp. 31-45, Canada, 2007
6. Lenders, V., Koukoumidis, E., Zhang P., and Martonosi, M.: Location-based Trust for Mobile User-generated Content: Applications, Challenges and Implementations. In Proc. HotMobile 2008
7. Saroiu, S., and Wolman, A.: Enabling new mobile applications with location proofs. In Proc. HotMobile 2009
8. Luo, W., and Hengartner, U.: VeriPlace: A Privacy-Aware Location Proof Architecture. In Proc. Of ACM SIGSPATIAL GIS, 2010
9. Gilbert P., Cox L. P., Jung J., and Wetherall D.: Toward Trustworthy Mobile Sensing. In Proc. HotMobile 2010
10. Zhidong Shen; Xiaoping Wu; Jing Zhan,: Trust management for mobile agent system based on trusted computing platforms. In Wireless Communications, Networking and Information Security (WCNIS), 2010 IEEE International.
11. Dua, A., Bulusu, N., Feng, W.-c., and Hu, W.: Towards trustworthy participatory sensing. In HotSec '09: Proc. of the Usenix Workshop on Hot Topics in Security, 2009
12. Trusted computing group - trusted platform module specifications. http://www.trustedcomputinggroup.org/developers/trused_platform_module/specifications Accessed on May 2013
13. eBay http://www.ebay.com Accessed on May 2013
14. Capkun, S.; Hubaux, J-P, "Secure positioning in wireless networks," Selected Areas in Communications, IEEE Journal on, vol.24, no.2, pp.221,232, Feb. 2006

15. Joengmin Hwang; Tian He; Yongdae Kim, "Detecting Phantom Nodes in Wireless Sensor Networks," INFOCOM 2007. 26th IEEE International Conference on Computer Communications. IEEE , vol., no., pp.2391,2395, 6-12 May 2007
16. Alfaro, J.G.; Barbeau, M.; Kranakis, E., "Secure Localization of Nodes in Wireless Sensor Networks with Limited Number of Truth Tellers," Communication Networks and Services Research Conference, 2009. CNSR '09. Seventh Annual, vol., no., pp.86,93, 11-13 May 2009
17. Govindan, K., Mohapatra, P.: Trust Computations and Trust Dynamics in Mobile Adhoc Networks: A Survey. In Communications Surveys & Tutorials, IEEE, vol.14, no.2, pp.279,298, 2012
18. Viriyasitavat, W.; Martin, A.: A Survey of Trust in Workflows and Relevant Contexts, Communications Surveys & Tutorials, IEEE, vol.14, no.3, pp.911,940, Third Quarter 2012
19. Ping Zhang; Durresi, A.; Barolli, L., Survey of Trust Management on Various Networks, Complex, Intelligent and Software Intensive Systems (CISIS), 2011 International Conference on, vol., no., pp.219,226, June 30 2011-July 2 2011
20. Moonga the Game. EverdreamSoft, `http://www.moonga.com` Accessed on May 2013