Location Proof via Passive RFID Tags*

Harry Gao, Robert Michael Lewis, and Qun Li

The College of William and Mary

Abstract. With the surge in location-aware applications and prevalence of RFID tags comes a demand for providing location proof service with minimal cost. We introduce two protocols that provide secure and accurate location proof service using passive RFID tags. Both protocols are lightweight, adaptive and cost-effective. The first protocol assumes the connection of a user to the remote server. The second protocol does not require real time interactions with the server. Instead, it uses the self-reported time of local RFID reader (such as a cell phone), which may be biased. The user can upload the information to the server later to obtain the location proof. The paper presents a solution to derive users' actual time of presence in the absence of a reliable clock, assuming an arbitrarily large number of falsified data points from malicious users.

1 Introduction

Location proof service, which seeks to provide a means for clients to show that they are present at a particular place at a particular time, has been generated lots of interest from both companies and academia [9]. Social applications such as Foursquare, Yelp, Gowalla and Google Latitude are just some of the recent burst of location-aware services.

These applications seek to take advantage of users' location information to provide unique and personalized resources and services. Other classic examples of location proof service include location-based access control, bad player identification, creating alibi etc. It is standard to assume some preexisting certification authority (CA) [18], which would be referred to as the "server" in this paper.

In this paper we present a location proof protocol that is designed to be adaptable to different situations with respect to real-life needs. Benefits of the protocol we discuss include the possibility of replacing some of the more complicated and expensive systems in place today, providing location-based security measures, and deploying location proof services to applications previously economically prohibitive.

Passive RFID tags are distributed to where we wish to provide location proof service, so that the users can automatically scan these tags when they are present. Successful access to the tags indicates that the user is at the location. The key challenge is that accurate timekeeping is a necessity for location proof

* This project was supported in part by grants CNS-1117412, CAREER Award CNS-0747108, and CSUMS grant DMS 0703532.

X. Wang et al. (Eds.): WASA 2012, LNCS 7405, pp. 500–511, 2012.

service; however, a smart tag, unlike a more advanced computing devices, does not have the energy required to keep its own reliable clock. If we overcome this obstacle by assuming the proximity of a server or other powerful computational devices, such systems would not be qualitatively simpler or cheaper than existing technologies.

We present two protocols that utilize passive RFID tags to provide location proof service. The first one requires real-time server interaction, which can provide precise time of presence, and only require one remote central server. The second protocol does not require real-time server interaction, but would lose some precision due to the lack of a reliable clock. The latter is academically and practically interesting because real-time server interaction may be impossible for certain applications. For instance, if we wish to provide location proof service to subway systems in metropolitan areas, it is often the case that cell phones have little or no signal when underground. While cell phones can communicate with local RFID tags, interacting with servers become impossible in this situation. The offline server protocol broadens the field of potential applications while reduces the cost.

Both protocols require little or no user interaction, and can be fully automated. The design is a good fit for most situations location proof service may be required, such as established public locations, including shopping centers, public transportation, parking garages, offices, restaurants, hospitals etc. Our design can provide the service at a much lower cost than contemporary measures, and can help extend location proof service into sectors previously prohibited due to economic and technical concerns.

The paper first introduces two straightforward protocols for providing location proof service with and without the need for real-time communication with the server. Then we introduce an algorithm to support the offline protocol and to increase the quality of service. Due to the page limit, we have omitted the more advanced algorithms and evaluation.

2 Related Work

Considerable work has been done in the field of location proof service, primarily based on the use of sensors, computers, or other similar advanced hardware [9,11,12]. While these approaches provide a cheaper and easier means of location proof than conventional camera-based methods, the cost of deployment and maintenance is still substantial. The design we present uses RFID technology, which is significantly cheaper than other electronic devices, to achieve the same end. Unlike GPS-based or mote-based solutions, there is not a need to replace batteries, and the costs of initial deployment and hardware are significantly lower.

As noted previously, the low computational and storage capacities of RFID tags compared with sensors and computers pose a challenge. Previous work such as [3,4,8] provide important insight into designing systems and protocols that increase the reliability of the location proof service and reduce a malicious party's

means to obtain location proof falsely. However, previous work tends to bypass the issue of the absence of a reliable clock by assuming the sole malicious user could not collude with others in a multi-pronged attack, or by real-time interaction with more sophisticated hardware. Without an intelligent design that derives the actual time from local information, the protocols must rely on external trusted sources such as an online server to provide the certified time. Other related work includes [14,10,13,15,7,16,5,17,1,6].

3 Problem Formulation

The overall goal is to provide location proof service using passive RFID tags. We assume the passive RFID tags can only perform simple hash functions and arithmetic calculations, and is unable to perform public encryption/decryption. We make the following assumptions about the situation in which we wish to deploy a location proof service:

1. The service is used frequently and by many users.
2. In the absence of truly malicious users, minor inexactness in the time of visit is acceptable.
3. There exists the possibility of truly malicious users who attempt to fool or disrupt the system. Assumptions concerning malicious users are discussed in Section 3.1.

3.1 Adversary Model

For the online protocol, we have an accurate clock and the malicious parties are unable to manipulate the clock to their advantage.

For the offline protocol, in the absence of malicious users, we can still determine times of presence correctly within the fluctuation range of users' clocks. At the same time, we must also be able to deal with the wickedness of malicious parties.

We assume that a malicious party has two primary goals: falsely obtaining location proof, and/or disrupting the service by making the solution wrongly record the time of presence for other users. To achieve the first goal, the malicious party may either falsely report the time at which he is present at the particular location, or report a time t at which he is not present. For example, if the malefactor Mallory attempts to demonstrate falsely that he is present at the opening ceremony of the Olympics, he can try to either show up in the Olympic stadium before or after the ceremony and obtain a location proof with a false timestamp, or simply not show up at the Olympic stadium while attempting to appear to have been present.

The malicious party is expected to have the following capabilities:

1. They can insert an arbitrarily large number of timestamps in order to contaminate the data stream, either to invalidate legitimate users' claims or corroborate malicious users' claims.

2. They can perfectly coordinate all malicious users' actions towards a common goal.
3. They can eavesdrop on legitimate users' communications.

Of the three, the first is by far most disruptive. It means that it is within the malicious party's power to submit as many timestamps as is physically feasible. Consequently, in dealing with a malicious party we must assume that in given pool of timestamps, a majority of them could be potentially malicious. While it is reasonable to argue that in certain situations the malicious party would not have enough access to dominate the data set, making an assumption on the upper bound of malicious data is difficult and unreasonable for other situations.

We have also identified two foremost weaknesses in the adversary:

1. Because of the computational intensity required to crack even short secret values via brute force and the secret values built into the RFID tags are known to the central server only, it is reasonable to assume that the malicious party cannot compromise these values. Thus, we will assume the security of timestamps reported by RFID tags to the server (as well as the security of the server itself).
2. In light of the preceding observation, the delivery of the falsified data requires the physical presence of malefactors in the proximity of an RFID tag. Marshalling a large number of malicious users in one physical location to deliver systematically falsified data would require a significant level of conspiracy and organization. For this reason it is reasonable to assume that the majority of users who report data are honest, and that there is some *a priori* upper bound on the number of malicious users present in a given span of time.

In the offline protocol, the countermeasures against a malicious party is based on a consensus of unique users, not of timestamps. This approach is based on the observation that there is no reasonable bound on the number of false time stamps, while there is a reasonable (or, at least, parameterizable) bound on the number of malicious users.

4 The Protocol

There are two protocols proposed for two different situations. The first one ultilizes a server (one can communicate with it using cellular communication, for instance), while the other requires no real time interaction from any devices other than the passive RFID tags. The latter might be necessary for specific applications where cellular signal is unaccessible or undesirable.

4.1 Symbols and Notations

Both the online and offline protocols will use the following symbols.

Reader	R
Server	S
time, adjusted time	t, $\hat{t}$
random number challenge	r
Tag's unique name	T_{id}
Secret values known to T_{id} and S	S_{id}
X encrypted by *name*'s private key	$\{X\}_{name}$

4.2 Online Protocol

The section describes a protocol that assumes the availability of a remote server to provide real time information that aid the local tags in providing the location proof service. The following protocol is designed to serve location proof, with one remote online server.

R to S	: Location Proof Service Request
S	: creates a new session for R; generates r; records r,t
S to R	: r, t
R to T_{id}	: r
T_{id}	: computes h(r, S_{id})
T_{id} to R	: h(r, S_{id}), T_{id}
R to S	: h(r, S_{id}), T_{id}, $\{T_{id}\}_R$
S	: verifies h(r,S_{id}); create the location proof: $\{T_{id}, \{T_{id}\}_R, t\}_{Server}$; end session
S to R	: $\{T_{id}, \{T_{id}\}_R, t\}_{Server}$

Pre-distribution

Arbitrarily many RFID tags can be distributed for the purpose of providing location proof. Each should have a unique ID and secret value, and all such ID-value pairs should be known to the server and no one else.

Communication from Reader to Server

A hello message indicating the reader's desire to obtain a location proof. The server will create a new session for this particular user, create a random challenge r, and records the time t at which the request was received.

Communication from Server to Reader

Server sends the time of presence t and the random challenge r to the reader. The notification of time t is entirely for the reader's convenience and has no impact on the integrity of the protocol.

Communication from Reader to Tag

The reader can then forward the challenge r to the tag. The tag uses the value r in computing a value v = $h(r, S_{id})$, where h is a one way hash function. The server can later compute the value in the same way to verify that the reader is present at the tag's location.

Communication from Tag and Reader
The tag sends the computed hash value to the reader. The tag also sends its own identification.

Communication from Reader to Server
The reader forwards both pieces of information (v = $h(r, S_{id})$ and T_{id}) to the server. The reader also signs and sends T_{id}, which allows the server to verify the user's identification. The reader does not need to send the time t because his session should still be active, so the server is aware of the value of t.
The server verifies the correctness of the hash value by carrying out the same calculation v = $h(r, S_{id})$, and if the result matches the value uploaded, the server can provide location proof to the reader.

Communication from Server to Reader
The server calculates the location proof $\{\{T_{id}\}_R, t\}_{Server}$, which encompasses four essential elements: the location, the person, the time of presence, the authenticator, in that order. The server can then terminate the session.

Note that all sessions should have a timeout threshold, and the location proof request is automatically denied if the protocol is not executed in full within the allocated time. This timeout threshold should not be too large, because an honest user with a reasonable connection would not require much time at all to complete the entire protocol. The protocol only proves that the user is present at the location some time while the location proof session is live, thus a large timeout threshold lowers the resolution of the protocol. For example, a user could take advantage of a large time out threshold by initializing the protocol prior to arrival at the desired location, then carry out the rest of the protocol later when he is actually present at the location to obtain a location proof with misleading time of presence.

4.3 Offline Protocol

The section describes a protocol that does not rely on real-time server interaction with the user. The server can be contacted at some time after users' provide timestamps to the RFID tag to provide the users to with their location proof.

The protocol contains two major components. The first component is exclusively between the reader and the tags, carried out locally in real time. The second component can be carried out at a later time, and is exclusively between the reader and the server. In the first part the reader sends the time to the tag and the tag stamps the time submitted, and provides the reader with all the information it needs upload to the server later. In the second part, the reader uploads all relevant information to the server for verification, and the server processes the information by comparing it with all the other location proof service requests. The server will consolidate the information from all the users to determine the accuracy of the readers' claims, finally infer the actual time of presence from the data available. The location proof is then sent back to the reader.
The steps of the protocol are as follows:

Pre-distribution

Arbitrarily many RFID tags can be distributed for the purpose of providing location proof. Each should have a unique ID and secret value, and all such ID-value pairs should be known to the server and no one else.

Communication from Reader to Tag

Because the tag has no other way of obtaining the time, the hello message from R to T_{id} should contain the current time t. The tag computes the encrypted value v using the time t, counter n (i.e., the sequence number of the timestamp), and secret value S_{id}:

$$v = h(t, n, S_{id}). \tag{1}$$

The tag then increments its internal counter.

Communication from Tag to Reader

The tag then sends v, n, and its ID back to the reader.

The reader-tag interaction is summarized below:

R to T_{id}	: t
T_{id} computes	: v=h(t,n,S_{id}), n++
T_{id} to R	: v,n,T_{id}

The user can hold on to the information provided by the tag (v, n, T_{id}) without any time-sensitive need to upload any information to the server. However, when the user is ready to obtain the location proof, he can carry out the following:

1. initiate a challenge-response nonce N to verify his ID.
2. Upload N, v, n, T_{id}, t, and the public key to the server.

Upon receiving the location proof service request and all the aforementioned material, the server verifies the following information before providing the location proof to the user:

1. *Verify that no user is using multiple pairs of keys.* A user database is queried to make sure the user is not using multiple public-private key pairs. Multiple submissions from the same user should be grouped together as such when calculating $\hat{t}$.
2. *Verify the tag's identication.* Carry out the same operation as the tag did in Eq. (1), and verify that the result matches the value v provided.
3. *Determine the time.* After checking the validity of the data point, the server should add it to the time-verification algorithm (described in detail in the following section) to calculate $\hat{t}$, the adjusted time.

Once the server obtains all three pieces of information, it can provide the location proof by

$$\texttt{location proof} = \{\{(T_{id}), \hat{t}\}_R\}\}_S \tag{2}$$

which summarily embeds the who, where and when.

Lastly, the server sends to the user the above location proof, $\hat{t}$ and the T_{id}, where the latter two serve as an readable description of the content of the location proof.

R and S	: exchange a challenge-response
	: nonce N to verify R's ID
R to S	: v, n, T_{id}, PK_R, t
S verifies	: $v == h(t, n, S_{id})$
S calculates	: $\hat{t}$
S to R	: $\{(T_{id}), \hat{t}\}_R\}_S$

4.4 Discussion

The protocols defend against many malicious attacks, and ensures that no personal or crucial information is passed back and forth when the reader communicates with the tags; therefore, an eavesdropper cannot hope to capture the radio signal and harvest users' private information. There is also little set-up, and clients can be added and removed from the system seamlessly. A malicious party cannot hope to obtain a location proof without being physically present at the location.

The online protocol needs to address denial of service attacks where a malicious user repeatedly request for a new session to be opened, hence exhausting system resources and hinder the server's capacity to aid good users. The effectiveness of this attack is limited by the timeout threshold of the session.

The offline protocol has one key difference from the online protocol: the tags receive the unreliable time of presence t from the reader, which cannot be fully trusted. Therefore the offline protocol relies heavily on the time-verification scheme from central processing and consolidation of data points.

A malicious user can report falsely the time of presence, hoping to obtain a location proof with a timestamp different from when he is actually there. He may also report a large number of data points to support his false claims and distort others time of presence. This translates to server's responsibility to sort out a large number of data points. In other words, malicious users can deliberately submit a false time t to achieve their goals outlined in the adversary model section, but behave otherwise the same as a good user to avoid detection. If a user is allowed to use multiple pairs of keys without the server's knowledge, a malicious user could create an unbounded number of phantom users and skew the $\hat{t}$ calculation, which is based on the consensus of users.

The goal of checking the uniqueness of the users is to recognize all data from one user belongs to exactly one user, but it would not allow the server to pinpoint the identity. In other words, the server is interested only in finding out how many

users it is servicing, not who. This protects the users' privacy while allowing the server to correctly root out malicious inputs.

To protect users' privacy, we can segregate the tag-dispensing service from the server consolidate service. This way, the server may be aware, for example, that a particular, anonymous user travelled from point A to point B before turning to point A, it has no way of knowing exactly where points A and B are. Furthermore, all data are collected anonymously, because the server only verifies the users' uniqueness by checking the validity of the signature against a database, but the server is unable to obtain the identities or any sensitive personal information of the users. The privacy of the users is at risk only if the multiple service providers (database, consolidation service and tag-dispensing service) collude together.

5 Dealing with False Timestamps

We next turn to the problem of dealing with false timestamps, particularly those introduced into the data stream by a malicious party. Because the protocol does not make any assumptions on the delay between time of presence and the submission time, attacks such as delayed attack would have no effect on the protocol's accuracy. In general, we wish to derive from our collected data a monotonically non-decreasing series of adjusted timestamps $\hat{t}_1 \leq \hat{t}_2 \leq \cdots \leq \hat{t}_n$ that best describes the behavior of the data. From the index numbers the local tags embed in each data point we know the chronological order the data points should be in. However, there is no guarantee that the time reported at those data points are correct, or even if they are monotonically non-decreasing. If there are timestamps that are out-of-order, the central server must resolve the conflict. This means we must derive a series of adjusted timestamps, subject to the monotonicity constraint, that best resemble the data points reported in individual location proof service requests. In the following, we present our solution to this problem.

Our first step is to consider the problem of finding a monotonically non-decreasing series that minimizes an ℓ^p-norm mismatch with the observed timestamps. This solution treats all timestamps equally and attempts to find a monotonic time series that best resembles the given timestamps. This monotonicity constraint makes this approach an instance of *isotonic regression*.

The three measures of mismatch that we consider are the ℓ^1, ℓ^2, and ℓ^∞ norms, since these leads to easily solved optimization problems. The ℓ^1 fit is the least sensitive to outliers, while the ℓ^∞ fit is the most sensitive, and ℓ^2 is in-between. Thus, sporadic malicious timestamps with incorrect values affect the solution of the ℓ^1 and ℓ^2 fits less than the ℓ^∞ fit.

The ℓ^1 fit is the problem of finding a solution $t = (\hat{t}_1, \ldots, \hat{t}_n)$ of the following convex program:

$$\begin{aligned} &\underset{t}{\text{minimize}} \quad \sum_{i=1}^{n} \left| \hat{t}_i - t_i \right| \\ &\text{subject to } \hat{t}_1 \leq \hat{t}_2 \leq \cdots \leq \hat{t}_n. \end{aligned} \tag{3}$$

This problem can be transformed into the following linear program:

$$\begin{aligned} \underset{s,t}{\text{minimize}} \quad & \sum_{i=1}^{n} s_i \\ \text{subject to } & t_1 \le t_2 \le \cdots \le t_n, \\ & -s_i \le t_i - d_i \le s_i,\ i = 1, \ldots, n. \end{aligned} \tag{4}$$

The advantage of the linear programming formulation (4) is that linear programs can be solved very efficiently. A drawback of (4) is that there may be nonunique solutions.

The ℓ^2 fit seeks a solution of the following convex quadratic program:

$$\begin{aligned} \underset{t}{\text{minimize}} \quad & \sum_{i=1}^{n} \left| \hat{t}_i - t_i \right|^2 \\ \text{subject to } & \hat{t}_1 \le \hat{t}_2 \le \cdots \le \hat{t}_n. \end{aligned} \tag{5}$$

While this problem can be solved by standard quadratic techniques, there also exist specialized $O(n)$ algorithms for its solution (see [2] for an overview). Solutions of this problem are unique.

Finally, ℓ^∞ fit entails the solution of

$$\begin{aligned} \underset{t}{\text{minimize}} \quad & \max_{i=1,\ldots,n} \left| \hat{t}_i - t_i \right| \\ \text{subject to } & \hat{t}_1 \le \hat{t}_2 \le \cdots \le \hat{t}_n. \end{aligned} \tag{6}$$

Like the ℓ^1 fit, this problem can be transformed into a linear program:

$$\begin{aligned} \underset{s,t}{\text{minimize}} \quad & s \\ \text{subject to } & t_1 \le t_2 \le \cdots \le t_n, \\ & -s \le t_i - d_i \le s,\ \ i = 1, \ldots, n. \end{aligned} \tag{7}$$

As with the ℓ^1 fit, that there may be nonunique solutions to (7).

Unfortunately, straightforward isotonic regression is easily thwarted by a malicious party. We do not assume a bound on how many bad timestamps the malicious party can inject into the data pool, and there is no restriction on how far from the actual time the malicious data can be. As a consequence, it is not hard to see that the malicious party can take advantage of the situation by flooding the data stream with false timestamps with systematic errors (i.e., the false timestamps are all in the future or all in the past relative to the truth). With a large number of false timestamps, the malicious party can manipulate the results of any of the isotonic regression fits just described, and even a small proportion of false timestamps can cause mischief.

This point is illustrated in Fig. 1. The vertical axis denotes normalized time in the range 0 to 1. In these simulations, malicious (but self-consistent) timestamps have been introduced that are out of sync with legitimate timestamps. The results of isotonic regression for the ℓ^1, ℓ^2, and ℓ^∞ norms are indicated by the black line. We have designed more advanced approaches to resolve the problem, but they are omitted in this paper due to the page limit.

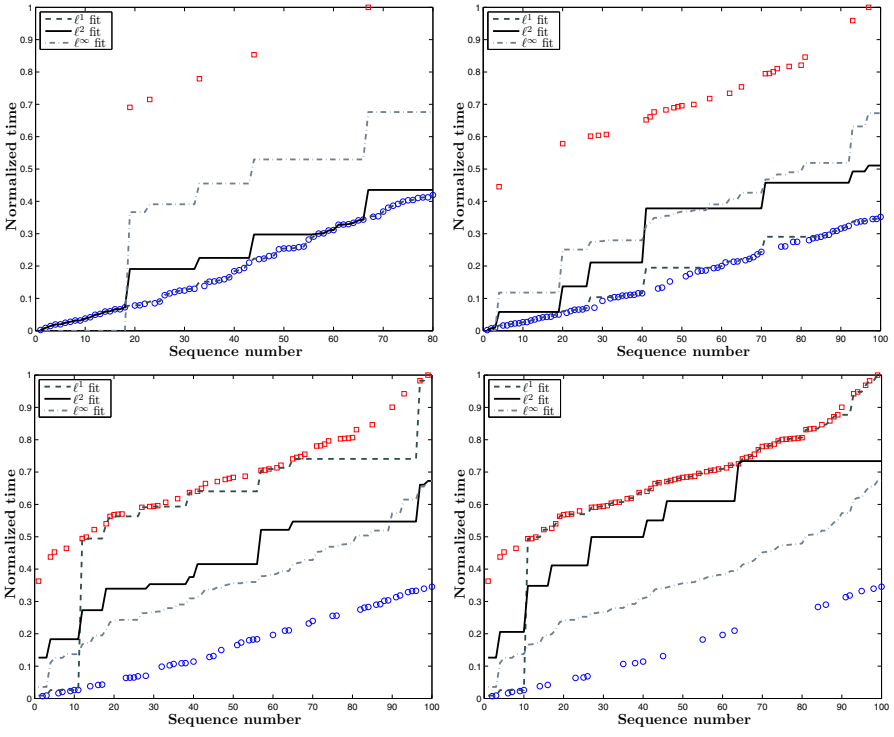

Fig. 1. The red squares are false timestamps, the blue circles are true timestamps, and the lines indicate the result of isotonic regression in the ℓ^1, ℓ^2, and ℓ^∞ norms. The four figures are for the following four cases separately: up-left (75 true, 5 false), up-right (75 true, 25 false), bottom-left (50 true, 50 false), bottom-right (25 true, 75 false).

6 Conclusions

Through two protocols and an approach to correct for incorrect user inputs, we have shown the feasibility of providing location proof service using RFID tags. While economical, the simplicity of the hardware imposes serious challenges to the central processing unit which is responsible for consolidating and verifying data from local tags.

References

1. Bai, X., Zhang, C., Xuan, D., Teng, J., Jia, W.: Low-connectivity and full-coverage three dimensional wireless sensor networks. In: Mobihoc, pp. 145–154 (2009)
2. Best, M.J., Chakravarti, N.: Active set algorithms for isotonic regression: A unifying framework. Mathematical Programming 47(1–3), 425–439 (1990)
3. Bussard, L., Bagga, W.: Distance-bounding proof of knowledge to avoid real-time attacks. In: SEC, pp. 223–238 (2005)

4. Chiang, J.T., Haas, J.J., Hu, Y.-C.: Secure and precise location verification using distance bounding and simultaneous multilateration. In: Proceedings of the Second ACM Conference on Wireless Network Security, pp. 181–192. ACM, New York (2009)
5. Ding, M., Liu, F., Thaeler, A., Chen, D., Cheng, X.: Fault-tolerant target localization in sensor networks. EURASIP Journal on Wireless Communications and Networking 2007(1), 19–19 (2007)
6. Gu, W., Bai, X., Chellappan, S., Xuan, D.: Network decoupling for secure communications in wireless sensor networks. In: 14th IEEE International Workshop on Quality of Service, IWQoS 2006, pp. 189–198. IEEE (2006)
7. Han, H., Sheng, B., Tan, C.C., Li, Q., Mao, W., Lu, S.: Counting RFID tags efficiently and anonymously. In: IEEE Infocom, San Diego, CA (March 2010)
8. Hancke, G.P., Kuhn, M.G.: An RFID distance bounding protocol. In: SECURECOMM 2005, pp. 67–73. IEEE Computer Society, Washington, DC (2005)
9. Luo, W., Hengartner, U.: Veriplace: a privacy-aware location proof architecture. In: GIS, pp. 23–32 (2010)
10. Ren, S., Li, Q., Wang, H., Chen, X., Zhang, X.: Analyzing Object Detection Quality Under Probabilistic Coverage in Sensor Networks. In: de Meer, H., Bhatti, N. (eds.) IWQoS 2005. LNCS, vol. 3552, pp. 107–122. Springer, Heidelberg (2005)
11. Saroiu, S., Wolman, A.: Enabling new mobile applications with location proofs. In: Proceedings of the 10th workshop on Mobile Computing Systems and Applications, HotMobile 2009, pp. 3:1–3:6. ACM, New York (2009)
12. Stojmenovic, I., Liu, D., Jia, X.: A scalable quorum based location service in ad hoc and sensor networks. Int. J. Commun. Netw. Distrib. Syst. 1, 71–94 (2008)
13. Tan, C.C., Sheng, B., Li, Q.: How to monitor for missing RFID tags. In: IEEE ICDCS, Beijing, China, pp. 295–302 (June 2008)
14. Wang, H., Sheng, B., Tan, C.C., Li, Q.: WM-ECC: an Elliptic Curve Cryptography Suite on Sensor Motes. Technical Report WM-CS-2007-11, College of William and Mary, Computer Science, Williamsburg, VA (2007)
15. Wang, H., Tan, C.C., Li, Q.: Snoogle: A search engine for physical world. In: IEEE Infocom, Phoenix, AZ, pp. 1382–1390 (April 2008)
16. Xie, L., Sheng, B., Tan, C.C., Li, Q., Chen, D.: Efficient tag identification in mobile RFID systems. In: IEEE Infocom, San Diego, CA (March 2010)
17. Xing, K., Ding, M., Cheng, X., Rotenstreich, S.: Safety warning based on highway sensor networks. In: Wireless Communications and Networking Conference, vol. 4, pp. 2355–2361. IEEE (2005)
18. Zhu, Z., Cao, G.: Applaus: A privacy-preserving location proof updating system for location-based services. In: INFOCOM, pp. 1889–1897 (2011)