LEXP: Preserving User Privacy and Certifying the Location Information

Ken Nakanishi[1], Jin Nakazawa[2], and Hideyuki Tokuda[13]

[1] Graduate School of Media and Governance, Keio University
[2] Keio Research Institute at SFC
[3] Faculty of Environment Information, Keio University
{ken, jin, hxt}@ht.sfc.keio.ac.jp

Abstract. We propose Location information EXchange Protocol (LEXP) as a protocol for location-aware applications using a tracking system. This protocol is designed for preserving user privacy, and certifying users location information. In LEXP, object-detection entities are separated from location-aware applications, and users can disclose their location information based on their intention. LEXP guarantees users to keep anonymity, and guarantees applications that a user cannot forge his location information. LEXP realizes these requirements by applying '*chain of confidence*' model and extending one-time password architecture.

1 Introduction

Many kinds of location sensing technologies already exist, and more are under development. Some, tracking systems such as radio frequency IDs (RFIDs) and the Active Badge[15] system, detect the objects' presence in a certain region. Other, positioning systems such as the Global Positioning System and Cricket[12], allow an object to compute its location by itself.

Under the tracking systems, an identifier is attached to each object, and readers detect the identifiers inside their sensing area. In ubiquitous computing environments, identifiers can be attached to our PDAs and cellular phones, and many readers will pervasively exist in public settings such as shopping malls, and museums[13]. Location-aware services in such environments recommend our favorite apparel shop nearby in a shopping mall, describe the artist that painted the painting we are looking at in a museum, and provide some other information depending on our location. However, we are afraid that these services collude against us to share the track of our movement behind our back since we cannot basically trust these services and have no control over them. Assuming that, applications using a tracking system create significant privacy risks. Therefore, protecting personal privacy is going to be a prime concern for the deployment of location-aware applications[9, 6, 1].

While we do not trust location-aware applications, these applications might not trust us either. That is, they doubt that our location information describes our actual location. Several location-aware applications decide users' privileges based on their location[11]. For instance, a supermarket publishes discount coupons only to customers inside the store, and a family permits a visitor to operate networked home appliances while he is inside the house. Therefore, these applications need to certify users' location

information to protect themselves from being damaged by malicious users' false location information. While many works address the protection of user privacy, comparatively few consider the validity of location information since they basically treat users not to be suspected but to be protected on their assumption.

To cope with both of these problems, we have designed a protocol for location-aware applications using a tracking system, called Location information EXchange Protocol (LEXP). LEXP is designed to preserve user privacy, and certify users' location information by applying '*chain of confidence*' model and extending one-time password architecture. The rest of this paper is configured as follows. In section 2, we discuss design considerations. We explain the chain of confidence model and LEXP architecture in section 3. We present a security analysis of the protocol and social roles and responsibilities of the constituents in section 4. We describe related works and compare those with LEXP in section 5, and conclude in section 6.

2 Design Considerations

In this paper, we mainly address the RFID system as a prototype of tracking systems, and assume an environment in which many RFID readers pervasively exist in public settings and users move around with their mobile device such as PDA and cell phone, to which an RFID is attached.

In LEXP, entities which detect the objects inside a certain area are separated from location-aware applications. They send location information to the users inside their sensing area. The users disclose the information to an application based on their intention.

The following scenario illustrates example services using LEXP.

> *When a salesperson, Dom, visits his client's office to perform a sales presentation with his laptop to which an RFID is attached, the laptop obtains location information from a detection entity in the office. When he uses the networked devices there, such as printers and projectors, he is required to send his location information to the devices for clarifying that he is in the office.*
>
> *When he comes back to his office, he wears his neck strap to which an RFID is attached and starts an application on his desktop computer. The application obtains location information from detection entities, and spontaneously sends the information to an employee tracking service. The service can be aware of Dom's movement while the application is running.*
>
> *After work, he transfers his past location information from his laptop and desktop to his PDA which an RFID is attached to. He takes a side trip with his PDA, which obtains location information from detection entities in public settings. A diary service on his PDA can save the record of today's movement.*

Based on this scenario, let us discuss design requirements of LEXP.

Anonymity of Location Information If location information contains an identifier associated with users, malicious services can create histories of users' movement by gathering their location information. To prevent this situation, location information in LEXP should not contain any identifier associated with users.

Validity of Location Information If users can forge their location information, they might conspire to obtain some privileges to invoke them maliciously. In our scenario, Dom can cause serious damages to his client's office by using printers

maliciously from outside and interrupting office work (he can keep anonymity even if he sends fake location information). LEXP, thus, should guarantee applications that users cannot forge their location information.

Reusability of Location Information Users might send the same location information to different services, which confirm the validity of the information. In our scenario, Dom sent his location information to the employee tracking service. Then, he also sent the information to the diary service that refers to his past location information. LEXP, thus, should allow users to reuse their location information.

Independency of Location Information Frequently, a user owns several devices, which different RFIDs are attached to, and picks up the suitable device to carry among them. In this case, users might want to gather their location information distributed to these devices. In our scenario, Dom gathered his location information from his laptop and desktop into his PDA. LEXP, thus, should offer a format, which allows users to transfer their location information among their devices.

3 Architecture

Figure 1 shows the constituents of LEXP. A *detector* is a detection entity, connected to an RFID-reader. A *client*, corresponding to an RFID, is a user-side computing device. A *service provider* runs a location-aware application. The *resolver* is the entity which manages a mapping table between clients' RFID and IP address.

Clients send their address to the resolver every time the address has changed (*address notification*). When detectors detect an RFID inside their sensing area, they request the resolver to resolve the client's address that corresponds to the RFID (*address resolution*), and send a notification to the address that a *ticket* is available. Then, the client can obtain the ticket, which is a presence evidence at the detector's sensing area (*ticket publication*). When clients are requested a ticket by a service provider, they decide whether they consume the ticket based on user's intention or a formulated policy. After service providers obtain a ticket, they request the detector, which published the ticket, to verify it (*ticket verification*).

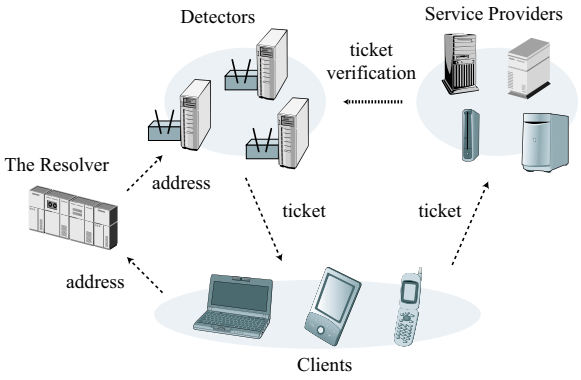

Fig. 1. Constituents of LEXP

3.1 Chain of Confidence

Clients might receive tickets from unknown detectors, and service providers might need to verify tickets generated by unknown detectors. Although clients and service providers want to confirm that the detectors are credible, it is a burden for them to sort out the credible detectors various amount in advance. To settle this problem, LEXP establishes a chain of confidence.

On the assumption, every client, detector, and service provider place confidence in the resolver. The resolver sorts out socially credible detectors (*detector registration*). At registration, the resolver registers a secret key for communication with a detector, and generates the certificate of the detector's public key.

At address resolution, detectors obtain clients' address. Clients allow the detectors to obtain their address since they indirectly trust the detectors, certified by the resolver. At ticket verification, LEXP applies public key cryptosystem. Service providers receive the validity of a ticket, which is encrypted by a detector's private key. Service providers can trust the validity since the detector's public key is certified by the resolver.

3.2 LEXP

This section describes the behavior of LEXP. We assume that every detector and service provider have a static IPv4/v6 address, and every client has an IPv4/v6 address, which might be private, and might change frequently[2]. In the rest of this paper, 'address' means IP address, and 'socket address' means IP address and port number. We set it forth as a premise that every client, detector, and service provider have acquired the resolver's RSA public key in a certified way, such as public key distribution with certificate authority[16] or software-preinstalled distribution. Each client, detector, service provider, and ticket have its own unique identifier CID, DID, SID, and TID. The description $h(x)$, $h^2(x)$, and $\{x\}_{key}$ denote the MD5 hash value of x, $h(h(x))$, and data x encrypted key, respectively.

Detector Registration As a configuration, the resolver needs to register a credible detector. At first, the resolver registers a secret key with the detector securely by publishing a password for the detector and using the resolver's public key. In LEXP, DES, DESede, and AES[4] are supported as a secret key algorithm, and 192 bits is the max size of a secret key. Next, it generates a certificate of the detector's public key.

Address Notification At address notification, clients send their $h(RFID)$ and socket address to the resolver every time the address has changed so that detectors can obtain the address by $h(RFID)$. LEXP needs to keep confidentiality of these information, and to prevent malicious users from registering fake address, in order not to provide tickets of the rightful users to them.

To notify its own socket address, a client sends a request to the resolver and gets a random number r. Next, it generates a random number ra, and acquires self-global socket address through the STUN protocol[7]. Then, the client start waiting UDP-datagram packets at the socket address, and sends to the resolver

$header\ \{h(RFID)\ h(CID)\ r\ ra\ socketaddress\}_{resolver\_publickey}$.

The resolver, holding the record sets of $h(RFID)$, $h(CID)$, socket address, and ra of clients, decrypts the received data by its private key. Then, the resolver checks r to prevent a replay attack, and checks if $h(CID)$ stored with $h(RFID)$ is equal to the

received one. If the condition is met, the resolver updates new socket address and ra of the client.

Address Resolution When detectors detect an RFID inside their sensing area, they need to obtain the socket address of a client that corresponds to the RFID in order to notify the client that a ticket is available.

To resolve the address, a detector generates a random number r and sends the resolver

$header\ h(DID)\{h(RFID)\ r\}_{secretkey}$.

The resolver, holding the record sets of $h(RFID)$, socket address, and ra of clients, decrypts the received data by registered secret key and searches the socket address and ra from record sets by $h(RFID)$, and sends back

$header\ \{r\ ra\ socketaddress\}_{secretkey}$.

The detector decrypts the received data, checks r to prevent a replay attack, and appends a record set of the socket address, ra, and $h(RFID)$.

Ticket Publication If clients receive a notification that a ticket is available, they can obtain the ticket as their location information. In order to meet the reusability of location information, tickets should be able to be consumed several times. Tickets should not contain any identity associated with clients in order to meet the anonymity of location information.

After address resolution, a detector sends a UDP datagram packet to the socket address of a client to notify that a ticket is available.

Receiving it, the client sends a request to the detector (the detector's socket address is contained in the UDP packet). The client and detector establish a secure session under the Diffie-Hellman key exchange algorithm[5]. Then, the client sends to the detector

$\{h(RFID) \oplus ra\}_{sessionkey}$.

The detector, holding the record sets of socket address, ra, and $h(RFID)$ of clients, searches $h(RFID)$ and ra from the record sets by the address of its source host. Then, the detector decrypts the received data, exclusive-ors decrypted data by ra, and checks if the calculated data is equal to $h(RFID)$. If the condition is satisfied, the detector generates a random number R and a ticket.

$ticket = \{TID\ timestamp\}_{detector\_privatekey}$

The detector sends to the client

$\{DID\ TID\ R\ ticket\}_{sessionkey}$.

The client decrypts the received data, and stores a set of DID, detector's address, TID, rt(=at this point's ra), R, and the ticket in XML format in order to meet the independency of location information.

After publishing, the detector calculates $h^N(R)$, and appends a new record set of TID, rt(=at this point's ra), $h^N(R)$, and n (at first $n = N$). n is the number of times the ticket is consumable. Neither the stored information nor tickets contain anything about clients.

Ticket Verification Service providers may request a ticket of a certain detector, or all the ticket a client has acquired, or the latest ticket the client has acquired. Users or client applications should decide whether they consume the ticket or not. Preferably,

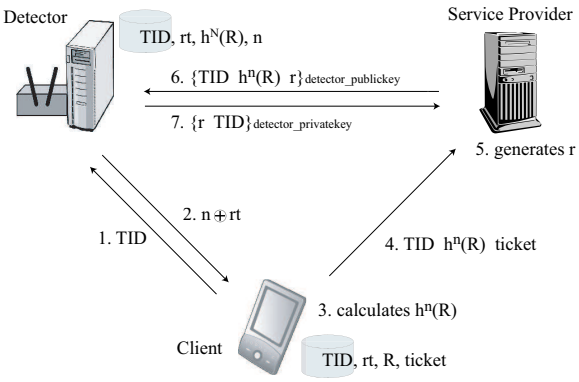

Fig. 2. Ticket Verification

client applications and service providers negotiate for tickets and client applications autonomously handle their tickets.

Figure 2 shows the brief transactions at ticket validating. A client sends a ticket's TID to the detector as a request to consume the ticket. The detector sends back $n \oplus rt$. The client obtains n by exclusive-oring received data by rt, generates the $h^n(R)$ from stored R, and sends TID, $h^n(R)$, and the ticket to a service provider (it also sends the detector's socket address if the service provider does not know the detector). Receiving the data, the service provider generates a random number r. If it does not have the detector's public key, it obtains the key, certificated by the resolver, from the detector. The service provider requests the detector to verify the ticket by sending

$header\ \{TID\ h^n(R)\ r\}_{detector\_publickey}$.

The detector decrypts the received data, searches the record set of the ticket from TID, and checks if $h^{N-n}(h^n(R))$ is equal to stored $h^N(R)$. If all conditions are satisfied, it decrements the ticket's n, and sends back

$header\ \{r\ TID\}_{detector\_privatekey}$.

The service provider decrypts the received data by the detector's public key, checks r to prevent a replay attack, and checks if TID can be obtained.

4 Analysis

This section explores the set of attacks that remain possible in each phase, and shows social roles and responsibilities of the constituents of LEXP.

4.1 Security Analysis

We explain how to adapt LEXP to thwart the following attacks. We also argue that some of the remaining attacks are not likely to be handled within the context of our system.

Address Notification If attackers sniff data packets at address notification, they can never obtain the data contents since it is encrypted with the resolver's public key.

If attackers try to perform a replay attack after packet sniffing, their attempt is thwarted since the data is exclusive-ored by a random number every time.

If attackers obtain an RFID by deploying RFID-readers by themselves and try to register their address in the resolver maliciously, their attempt is not carried out unless they know the CID corresponding to the RFID. However, if the attack is done before the very first time a client registers its $h(CID)$ with the $h(RFID)$, the resolver will register the attackers' fake $h(CID)$ and address.

Clients obtain self-global IP address and port number through the STUN protocol. If the STUN server disguises the address of a client, the client cannot obtain a ticket and other users might obtain it. LEXP cannot cover this problem, but assumes that the STUN protocol works correctively.

Address Resolution If attackers sniff data packets at address resolution, they can never obtain the data contents since the resolver and a detector communicate in a secure channel with the secret key, which has been transferred securely under public key cryptosystem.

If attackers try to perform a replay attack after sniffing the encrypted data, their attempt is thwarted since the data is exclusive-ored by a random number every time clients' address is exchanged.

Ticket Publication At ticket publication, a detector and a client generate a session key every time, and establish a secure session. Even if attackers pretend to be a detector and establish a secure session with a client, they cannot obtain the secret data unless they know ra, which is known only to rightful detectors. Also, if attackers pretend to be a client, they cannot obtain a ticket unless they know ra and $h(RFID)$.

Ticket Verification At ticket verification, a client obtains $n \oplus rt$ from a detector. If attackers sniff the data packets, they cannot obtain n unless they know rt.

The client sends a ticket and $h^n(R)$ to a service provider. The data might be sent without encryption. Even if attackers obtain the data by packet sniffing, they cannot track the user's movement since the ticket does not contain any identifier associated with the user, and also they cannot re-consume the ticket unless they know R of the ticket. This model is an application of the one-time password architecture[8]. $h^n(R)$ is the one-time password. The password required next time is $h^{n-1}(R)$, which attackers cannot generate unless they know R ($h^{n-1}(R)$ can never be calculated from $h^n(R)$ although $h^{n+1}(R)$ can be easily calculated from $h^n(R)$).

The detector, which generated a ticket, verifies the ticket and sends back its validity. No attacker can generate a fake result since the result is encrypted by the detector's private key. If attackers try to perform a replay attack after packet sniffing, their attempt is thwarted since the result is exclusive-ored by a random number every time.

4.2 Social Role and Responsibility

In this section, we describe social roles and responsibilities of the resolver, detectors, and service providers. Table 1 shows the clients' information that is obtained by the resolver, detectors, and service providers respectively. Personal information means the users' information in the real world such as actual name, sex, and age.

Table 1. Clients' information obtained by the resolver, detectors, and service providers

	Resolver	Detectors	Providers
RFID	–	◯	–
$h(RFID)$	◯	◯	–
$h(CID)$	◯	–	–
Personal Information	–	–	△

Resolver The resolver bears social responsibilities toward every client in sorting out credible detectors since it discloses clients' address to them.

The resolver is located at the top hierarchy of the chain of certificates in LEXP. Therefore, it should be run by an organization that has gained public confidence such as a governmental organization or a nonbusiness organization.

Detectors Detectors are the only entity that obtains clients' RFIDs. They take upon responsibility that they never disclose the RFIDs and they never stack the detected RFIDs.

If malicious detectors or attackers that deploy RFID-readers by themselves stack detected RFIDs that passed into their sensing area, they can create histories of the RFIDs' movement by gathering their information. However, they cannot figure out actually who carries the RFIDs since personal information is never acquired from an RFID in LEXP.

In any case, detectors should handle detected RFIDs confidentially under social and legal pressures like P3P[3]. In an ideal solution, every RFID-reader by itself should behave as a detector which contains computing capability, network connectivity, and runs LEXP protocol stack. Then, no one can obtain RFIDs improperly.

Service Providers Some services might work completely anonymously. Other services might publish an identifier to each user and manage the user's personal information. Receiving a ticket from a client, the former services confirm that the owner of the ticket certainly was in the detector's sensing area, and the latter services might figure out who the owner is, and might track the client in certain degree if the client sends many tickets.

Users should decide whether they disclose their ticket to a service provider by considering several aspects of the provider's credit. They can also set their client to send their ticket spontaneously to credible providers. The same way of disclosing personal information has been commonly accepted in real world, we disclose our information in order to use some services. Of course, service providers should treat clients' information carefully.

5 Related Work

In this section, we discuss three location-aware systems that aim to preserve user privacy.

The research of Norwegian Computing Center[14] presents advanced concepts for specifying policies in the context of a mobile phone network. These concepts enable

access control based on criteria such as time of the request, location, speed, and identity of the located object. However, the authors conclude by expressing doubt that average users will specify such complex policies with the knowledge of all applications. In LEXP, users do not need to have the knowledge of applications in advance. They can decide to disclose their location information when utilizing a location-aware service.

Context Service[10] is a general middleware infrastructure for context collection and dissemination. It has a server, which stores location information for an each organization such as an enterprise or a family. As a configuration, the administrator of the organization defines some groups to classify users, and the administrator or the member of a group need to set a policy, which describe what information is permitted to read for each application. Context Service enforces users to set a server and have knowledge of all applications to set a privacy policy in advance, and costs for configuration is much higher than LEXP.

Cricket is a location-support system for in-building, mobile, and location-aware applications, using own location detectable devices *listeners* and *beacons*. A listener learns its physical location by analyzing information from beacons spread throughout the building, and sends its location information to an application if it wants to. Cricket is similar to LEXP in disclosing user's location information in response to user's demand. However, Cricket has no mechanism that an application can use to verify the validity of users' location information.

6 Conclusion

We proposed Location information EXchange Protocol (LEXP) as a protocol for location-aware applications using a tracking system. LEXP was designed for preserving user privacy and certifying users location information. In LEXP, object-detection entities are separated from location-aware applications, and users can disclose their location information based on their intention. LEXP guarantees users to keep anonymity, and guarantees applications that users cannot forge their location information. LEXP realizes these requirements by applying chain of confidence model and extending one-time password architecture. We will develop a context-aware application framework, which is aware of user context lies over LEXP and discloses a ticket based on user context automatically.

References

1. A. R. Beresford and F. Stajano. Location privacy in pervasive computing. IEEE Pervasive Computing, 2(1), 2003.
2. C. Bisdikian, I. Boamah, P. Castro, A. Misra, J. Rubas, N. Villoutreix, D. Yeh, V. Rasin, H. Huang, and C. Simonds. Intelligent pervasive middleware for context-based and localized telematics services. In *Proceedings of the second international workshop on Mobile commerce*, pages 15–24. ACM Press, 2002.
3. L. Cranor, M. Langheinrich, M. Marchiori, M. Presler-Marshall, and J. Reagle. The platform for privacy preferences 1.0 (p3p1.0) specification. W3C Recommendation, April 2002.
4. J. Daemen and V. Rijmen. *The Design of Rijndael*. Springer-Verlag New York, Inc., 2002.

5. W. Diffie and M. E. Hellman. New directions in cryptography. *IEEE Transactions on Information Theory*, IT-22(6):644–654, 1976.
6. A. F. Ginger Myles and N. Davies. Preserving privacy in environments with location-based applications. IEEE Pervasive Computing, 2(1), 2003.
7. C. H. J. Rosenberg, J. Weinberger and R. Mahy. Stun - simple traversal of user datagram protocol (udp) through network address translators (nats). The Internet Engineering Task Force, Request for Comments: 3489, 2003.
8. L. Lamport. Password authentication with insecure communication. *Communications of the ACM*, 24(11):770–772, 1981.
9. M. Langheinrich. A privacy awareness system for ubiquitous computing environments.
10. H. Lei, D. M. Sow, I. John S. Davis, G. Banavar, and M. R. Ebling. The design and applications of a context service. *ACM SIGMOBILE Mobile Computing and Communications Review*, 6(4):45–55, 2002.
11. D. W. Naveen Sastry, Umesh Shankar. Secure verification of location claims. ACM Workshop on Wireless Security (WiSe 2003) (to appear), September 2003.
12. N. B. Priyantha, A. Chakraborty, and H. Balakrishnan. The cricket location-support system. In *Proceedings of the sixth annual international conference on Mobile computing and networking*, pages 32–43. ACM Press, 2000.
13. S. Sarma, D. Brock, J. Foley, L. Putta, S. Ramachandran, and G. Nassar. The object name service: Version 0.5 (beta), technical report mit-autoid-tm-004. Technical report, Auto-ID Center, February 2002.
14. E. Snekkenes. Concepts for personal location privacy policies. In *Proceedings of the 3rd ACM conference on Electronic Commerce*, pages 48–57. ACM Press, 2001.
15. R. Want, A. Hopper, V. Falcao, and J. Gibbons. The active badge location system. *ACM Transactions on Information Systems (TOIS)*, 10(1):91–102, 1992.
16. L. Zhou, F. B. Schneider, and R. V. Renesse. Coca: A secure distributed online certification authority. *ACM Transactions on Computer Systems (TOCS)*, 20(4):329–368, 2002.