Splendor: A Secure, Private, and Location-aware Service Discovery Protocol Supporting Mobile Services*

Feng Zhu[1] Matt Mutka[1] Lionel Ni[1,2]
[1]*Department of Computer Science and Engineering, Michigan State University, East Lansing, Michigan, USA*
[2]*Department of Computer Science, Hong Kong University of Science and Technology, Kowloon, Hong Kong, China*
{zhufeng, mutka, ni}@cse.msu.edu

Abstract

In pervasive computing environments, powerful handheld devices with wireless connections create opportunities for many new nomadic applications. We propose a new service discovery model, called Splendor, supporting nomadic users and services in public environments. Splendor emphasizes security and supports privacy. Location awareness is integrated for location dependent services discovery and is used to lessen service discovery network infrastructure requirements. We analyze the Splendor system performance and provide our experimental results.

1. Introduction

We increase the usage of various computer devices and network services at our homes or in our offices to facilitate our daily tasks, but we also spend more effort to manage these devices and services. Service discovery protocols simplify the interactions among users, devices, and services. Many service discovery products and protocols are designed to solve this dilemma in home and enterprise environments.

Handheld and wearable computers are becoming more powerful and practical. As prices decrease, there are more mobile services and users. Meanwhile, these mobile devices increasingly support nomadic users by offering 2.5G/3G, wireless LAN, or Bluetooth capabilities. They may even support critical tasks, as in the scenario below.

David is a physician who volunteers to help patients such as at a shopping mall in case of emergencies. He has a handheld with a cell phone (3G), Bluetooth, and IEEE 802.11b built-in. Patrick (a patient) is over 70 and has heart disease. He also has a handheld similar to David's. In his handheld, all his vital signs and disease history are stored. Assume David and Patrick are at the same shopping mall on a Saturday, when Patrick has a heart attack. He pushes one button on his handheld. As a result, his handheld dials 911 to contact emergency rescue services. In addition, the handheld signals Mobile 911 (M911) – which is a request to find help for those in the immediate vicinity. Via the M911 signal, David is notified of this emergency and Patrick's position. David follows the directions on a map shown on his PDA, while listening to Patrick's medical history as he moves towards Patrick. David finds Patrick and offers some assistance before the ambulance arrives.

This scenario illustrates that a mobile client discovers a mobile service in an infrastructure environment. Let's analyze this scenario and discuss the challenges in this type of environment. First, how are people's mobility and services' mobility supported? For example in our scenario many people (physicians and patients) are at shopping malls and they come and go. Second, how are security and privacy provided? For instance, neither David nor Patrick is a user of the shopping mall computing system, but we need to authenticate David and Patrick. Patrick expects that his medical information is secure and not even revealed to the shopping mall's system. Moreover, David wants to offer his expertise while keeping his privacy. Third, how is location information, which enables David to easily find Patrick in case of emergencies, integrated to service discovery?

Based on this scenario, we identify that security, privacy, and location-awareness are important in service discovery for both nomadic users and services. Splendor considers environments, in which services may be discovered, but mobile users and services may not have accounts in the infrastructure systems. Therefore users, services, and network infrastructure systems are not trusted by each other. To our knowledge, most service discovery protocols are designed for home or enterprise environments, but not for these types of untrustworthy environments, which Splendor targets. We propose a new model to support mobility, security, and user privacy, while in the meantime we integrate location-awareness to service discovery. The security protocols in Splendor enables all parties to mutually authenticate each other, no matter if users have accounts in the network infrastructure systems or not. The security protocols also provide service authorization, confidentiality, integrity, and non-repudiation capabilities. Furthermore, Splendor supports user privacy, but so far we are not aware that any other service discovery protocol does. Moreover, Splendor

* This paper was supported in part by NSF Grants No. CCR-0098017, EIA-9911074, MSU IRGP Program and the Microsoft Research Foundation.

Proceedings of the First IEEE International Conference on Pervasive Computing and Communications (PerCom'03)
0-7695-1893-1/03 $17.00 © 2003 IEEE

integrates location-awareness, which we believe that the integration not only helps the service discovery, but may lessen the service discovery network infrastructure requirements as well. While providing these functionalities, Splendor still keeps the applications as easy to use as possible.

In Section 2, we discuss related work in service discovery and location-awareness. Next, in Section 3, we present our architecture and our ideas to solve all these challenges. In Section 4, we discuss the system performance and analyze the critical path of M911, as an application of the Splendor framework. Last, we list our future research directions in Section 5.

2. Related Work

Discovery of available services is a basic and critical task in pervasive computing environments [1]. Many service discovery protocols or products address service discovery mechanisms for different environments. Our work is largely influenced by these projects and is based upon them. In another paper [2], we categorize and analyze various service discovery protocols including Bluetooth Service Discovery Protocol [3], DEAPspace [4], Intentional Naming System (INS) [5] and INS/Twine [6], Jini [7], Salutation [8], Secure Service Discovery Service (SSDS) [9], Service Location Protocol (SLP) Version 2 [10], and Universal Plug and Play (UPnP) [11]. DEAPspace proposed a service discovery mechanism in single hop ad hoc environments. Bluetooth SDP enables nearby Bluetooth devices to discover each other's services. UPnP targets home environments. Jini and SLP focus on enterprise environments. Salutation works for both home and enterprise environments. Both UPnP and Salutation are device and appliance oriented. INS emphasizes name-to-service mapping. INS/Twine and SSDS address support for large numbers of services.

Only a few protocols have built-in security. SSDS, from UC Berkeley, implements more security features than other service discovery protocols [9]. In SSDS, services and clients trust directories, known as Service Discovery Service servers. Authentication between clients and services is based on certificates. Public key and symmetric key encryption are used for confidentiality and communication data privacy. Message Authentication Code (MAC) is used to ensure message integrity. Services manage their access control lists for their users and publish on servers, known as capacity servers. In short, SSDS provides security in distributed environments such as enterprise environments, but may not work in the environments that Splendor targets: directories may not be trusted servers; mobile services may not be able to handle service authorization; and there may not be centralized servers to store information. In addition, Splendor supports non-repudiation and user privacy.

Location-awareness is a key feature in pervasive computing [12]. Since AT&T OCL Active Badge [13], the pilot location-awareness project, many location-awareness research projects have been conducted [14]. Location sensing systems may be categorized as passive or active systems [15]. Active location sensing systems have sensing networks and track users' locations. On the contrary, passive location sensing systems do not track users and have distributed sensing devices, from which users read their location information.

Few projects integrate location information to service discovery protocols. Nevertheless, many services are location dependent. Coupled with network connections, location information may be very useful, such as in our scenario – M911. We not only use location information to better serve location dependent service discoveries, but use location sensing systems to provide more flexible service discoveries as well.

The Cooltown project, at Hewlett-Packard Laboratories, inspires our work [16]. One innovative idea is to tag things, places, and persons and to associate them with their "web presence". URLs are emitted from the tags and used in web browsers to obtain relative information from web servers. Thus, things, places, or people are vividly augmented. We tag places and people and integrate with our service discovery protocol to discover available services.

3. Architecture

In this section, first we describe how Splendor provides a new model to support mobile clients and services. Next, we illustrate our integration of location-awareness to our service discovery protocol. Then, we show the security support for all service discovery parties. Last, we discuss how Splendor supports privacy.

3.1. A New Service Discovery Model

For simple environments, such as home environments, *client-service models* are usually used. There are two types of components: clients and services. Clients look for services and services reply if they match required service attributes. Then clients select services to use. In more complex environments, *client-service-directory models* are deployed. Clients query directories; services register with directories and provide services to clients; directories cache service information and answer clients' queries. After receiving a matched service list from directories, clients select services, contact services, and then start to use services.

Providing security in enterprise environments, servers may be used to store and maintain user information. System users authenticate with the servers and are authorized to use services. For other environments, such as shopping malls, users do not have accounts on the shopping mall's computer systems. Third-party servers may be used for mutual authentication between each communicating pair, client-service, client-directory, and service-directory, as shown in Figure 1(a). These third-party servers may be trusted servers, as in SSDS, or untrusted servers. They may be inline, online, or offline third-parties [17]. We use this model to provide security for stable services. One advantage of this model is that the client-service-directory model may be used with little modification to support security for many situations. One disadvantage, however, is that many limited resource mobile services are burdened with various security checks and maintenance themselves.

Proceedings of the First IEEE International Conference on Pervasive Computing and Communications (PerCom'03)
0-7695-1893-1/03 $17.00 © 2003 IEEE

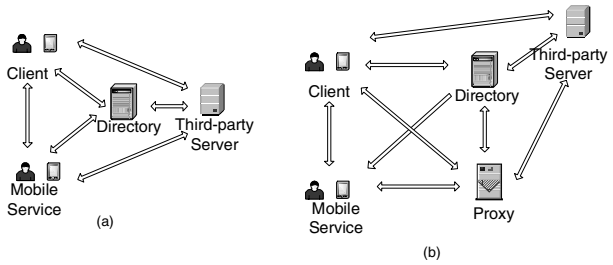

Figure 1. Two secure service discovery models with third-party servers. (a). Client-service-directory model. (b). Client-service-directory-proxy model.

In Splendor, we propose a new model that has four types of components: *clients*, *services*, *directories*, and *proxies*, shown in Figure 1(b). By including proxies, we may achieve privacy for service providers, offloading much mobile service's computational work to the proxies, and enabling mobile services to do authentication and authorization easily. We focus on this model for the rest of this paper. Non-mobile services work the same way as in the client-service-directory model, and we do not discuss in detail here.

3.2. Tag-based Location-aware Service Discovery

There are many technologies to do location sensing. We give further discussion of location dependent service discovery in a separate paper. Let's assume that we have a passive sensing system, which lets mobile clients and services read location information. The sensing system consists of two types of components: readers, which are attached to the mobile clients or services; beacons, which emit information either periodically or after reader's requests.

We use tags to label locations and people, for example, entrances of shopping malls and stores, or Patrick's and David's clothes. Tags, which label places, emit location information and optionally directory's addresses and the directory's certificates. (We use X.509 public key certificates [17] in Splendor, which are discussed in Section 3.4. For shorthand we use the term certificates.) The clients use the location information to search for the relevant services. Mobile services use the tags' information to determine that they have moved to new locations and notify their proxies. Tags attached to people's belongings other than the PDAs, such as clothes, may be used to verify that the PDAs are still in possession of their owners.

We show a snapshot that uses location tags for service discovery in Figure 2. In the upper half of the figure, we show a scenario in which a client may use a service through its 2.5G/3G connection. In the lower half, a client may use a wireless LAN for service discovery. Most service discovery protocols have an assumption that clients, services, and directories are using one underlying network connection for service discovery. We argue that location information not only provides better precision for location dependent service discovery, it also provides more flexible network infrastructure for service discovery.

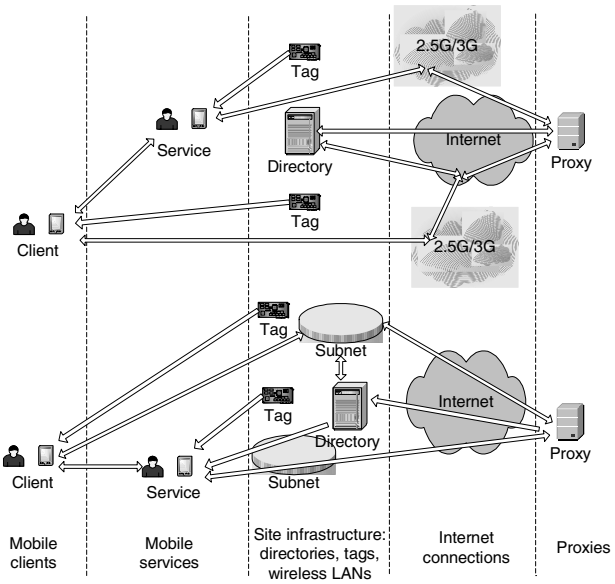

Figure 2. Location-aware service discovery architecture with clients, mobile services, directories, and proxies.

Service discovery may work with available wireless LANs, 2.5G/3G, other network connections, or combinations of these as long as the directory's addresses are known. Client queries, service registration, and client-service interactions do not need to be bound to any network connections.

3.3. Splendor Service Discovery Protocol

3.3.1. Bootstrapping. Multicast addresses are used for initial communications among clients, services, and directories. All communicating parties have a priori knowledge of these multicast addresses, so that no manual configuration is necessary for any party.

Mobile clients and services have three ways to tell that they move into new environments. First, they may find that they are attached to different networks, for example, that they acquired new IP addresses or were handed over to new wireless Access Points. Second, directories periodically announce their unicast network addresses, and send out solicitation messages asking services to register. After receiving these messages, mobile clients and services notice that the messages are from directories, which are in charge of other domains. Therefore, there are new directories around and they have moved to new places. Third, location-aware mobile clients and services may read location tags, hence they are able to tell whether they are in new places.

When mobile clients or services move to a new place, they may solicit directories for announcements. Using the directories' unicast addresses, clients may query for services and services know where to register.

3.3.2. Service Announcements and Lookups. After bootstrapping, the communications among clients, services, proxies, and directories are all unicast. In this way, only parties, which are necessary to be involved,

Proceedings of the First IEEE International Conference on Pervasive Computing and Communications (PerCom'03)
0-7695-1893-1/03 $17.00 © 2003 IEEE

handle messages. Mobile services authenticate with proxies and ask proxies to handle registration, authentication, authorization, and key management for them. (Mobile services may do all the work themselves without contact proxies.)

Proxies are trusted servers for services. They manage services, for example a medical organization offering licensed physicians for M911 services. When clients query the directories, they may receive some services represented by proxies. Therefore, clients need to contact proxies first and then services. Although an indirection is added into the interactions, removing overwhelming tasks such as service authorization from mobile devices is rewarding. Security policies may be easily modified at the proxy's side without worrying about deployment problems. Mobile service providers do not need to manage those security policies and settings.

Directories cache service information and answer client's queries. They also verify clients' and services' identities. Furthermore, directories may provide support for mobile clients, such as confirmation that the services' authenticities are checked and help to set up connections with services. While announcing their unicast addresses, directories may also announce their public key certificates. Clients and services use the certificates to verify and authenticate directories.

Most service discovery protocols store service states as soft state in directories [2]. Thus, services announce their lifespan. Before the services expire, they announce again. Handheld devices are less stable, since they may have poor wireless connection, or people may just turn them off. Compared to servers (directories and proxies), mobile services are transient. For that reason, Splendor caches soft state information of mobile services in proxies and deals with the instability simply and well. On the other hand, Splendor stores services represented by proxies as hard state in directories. This means that proxies explicitly register and unregister services with directories. To solve possibility of the inconsistent services' state problem, directories explicitly ask proxies about the services' status.

3.3.3. Mobility Support Using Aggregation and Filtering. One difficult problem in supporting mobile services is the extremely dynamic property of the services. Services may come and go, or even be shut down and come up again quite often. As a consequence, the directories keep refreshing the services' status – adding and removing entries. We address this problem by extensively using aggregation and filtering at the service registration stage and service lookup stage.

At the service registration stage, a service may not need to register and un-register itself repeatedly. In our scenario for instance, a physician's PDA will not contact the proxy as long as it is within the same shopping mall, although it moves between different stores. Proxies are also doing aggregation and filtering, whenever it is necessary. For instance, when a physician is in a shopping mall and another physician from the same hospital goes to the same shopping mall, instead of registering two services with the shopping mall's directory, the proxy may only register one. On the other hand, the mapping at the proxy's side is changed from one shopping center associated with one physician to one shopping center associated with two physicians. When another physician from the same hospital goes to the shopping mall, there may be still one record in the shopping mall's directory; or when a physician goes home, the record in the shopping mall's directory stays the same. At the service lookup stage, directories may match and/or filter queries first. When requests go to the proxies, based on the requests, the proxies may filter and match some services.

3.4. Security Issues

We consider mutual authentication, service authorization, confidentiality, integrity, and non-repudiation for Splendor [18]. Various public key and symmetric key technologies are used to achieve these goals. Because symmetric key encryption is much faster than public key encryption, we use public key techniques for signature and key management, while using symmetric key techniques for data encryption and date integrity. Each party has two sets of public keys: one for encryption and decryption use, another for signing and verification use [17].

Before communication, each communicating pair first sets up a session. In the session set up stage, a new session key is generated and securely transported to the other party using public key technologies. The session key is only known to the pair. It is used in the following session and discarded after the session finishes. On the contrary, those public keys used in session set up stage are used much longer. After communicating parties acquire session keys, communication data are encrypted using the session keys.

If any party wants to record messages for non-repudiation purposes, it may ask the other party to sign the messages using that party's signing private key before encrypted using their shared session keys. Since the signature uses the private key, which is slow, the messages are hashed first and then the hashes are signed. Using this mechanism, the messages are verifiable, even if the session keys are destroyed. The hashes are used for message integrity. The receiving party hashes the original messages and compares with the hashes.

Since clients, services, proxies, and directories may not belong to the same organization, Splendor is based upon Public Key Infrastructure (PKI) technologies, specifically the X.509 strong two way authentication protocol, which uses certificates [17] for communicating pairs to mutually authenticate each other and exchange keys. PKI enables strangers to exchange public keys securely and its passive infrastructure makes it very simple to use for end users [19]. X.509 two way authentication does key transport and to use public key certificate technology, thus no online trusted servers are needed to set up sessions and share session keys [17]. A public key certificate for each entity includes a serial number, the entity's name, its public key pair, a signature of a certification authority (CA) on the certificate, etc. (Detail certificate structure may be found in [19].) We

Proceedings of the First IEEE International Conference on Pervasive Computing and Communications (PerCom'03)
0-7695-1893-1/03 $17.00 © 2003 IEEE

Notation: C is a client; D is a directory; P is a proxy; S is a service. N_D, a directory's unicast address. $CertE_X$ is an encryption public key certificate of X. $CertV_X$ is a verification public key certificate of X. S_X is X's signature using its signing private key. T_X is the expiration time of the message, which is from X. r_X is a unique random number, which X generates in time period T_X. M is a message. K is a session key shared between the sending and the receiving party. A_X is X's multicast message. $P_X(K, Y)$ means Y generates a session key K and encrypts it with its identity using X's encryption public key. E_{KXY} is an encryption using symmetric key K shared between X and Y. t_X is a timestamp which X attaches. h(M) is a hash of a message, M. (Notation is similar to [17].)

Let $F = (T_D, D, N_D)$.

D→C: D→S:	$CertE_D$, $CertV_D$, F, $S_D(F)$	(1)

(a). A directory's announcement of its unicast address and certificates.

Let $R_P = (T_P, r_P, D, M, P_D(K1, P))$ and $R_D = (r_D, T_D, r_P, P, M, P_P(K2, D))$.

P→D:	$CertE_P$, $CertV_P$, R_P, $S_P(R_P)$	(1)
P←D:	R_D, $S_P(R_D)$	(2)

(b). Key transport between a proxy and a directory. T_X and r_X are used against replay attacks.

Let $R_C = (T_C, r_C, D, M, P_D(K1, C))$.

C→D:	$CertE_C$, $CertV_C$, R_C, $S_C(R_C)$	(1)

(c). Key transport between a client and a directory.

S→P:	$E_{KSP}(P, t_S, M, A_D)$	(1)
S←P:	$E_{KSP}(S, t_P, M)$	(2)

(d). A service forwards a directory's multicast message to a proxy. E_K here is a derived key from S's password. A_D is a directory's multicast message as shown in part (a).

C←P:	$E_{KPC}(P, C, S, K, t_S, M)$	(1)
S←P:	$E_{KPS}(P, S, C, K, t_S, M)$	(2)

(e). A session key generated at a proxy and transported to a client and a service.

X→Y:	$E_{KXY}(M)$	(1)

(f). A message encrypted using a session key shared between X and Y.

X→Y:	$E_{KXY}(M, h(M), S_X(h(M)))$	(1)

(g). A message is hashed and the hash is signed using X's signing private key before encryption using the session key shared between X and Y.

Figure 3. Security protocols among Splendor components.

assume that the public key infrastructures are available. Certificate revocation and trust models of public key infrastructures are important, but are out of the scope of this paper. Suppose there are CAs, which sign public keys for clients, directories, proxies, and services. These four parties acquire their certificates before interacting with others. When receiving a pair of new certificates, each party caches them locally. Before using the public keys in the certificates, the certificates are verified first: the signing CA is trusted, the signature is correct, the certificates are in valid time period and not revoked, and they are used for right purpose [19]. For example, a certificate, which a physician uses for his personal financial use, is not valid for his emergency services.

Service authorization is based on privilege levels. Therefore, mobile services only keep several levels. Proxies assign access levels to clients. Security policies are changed and applied at the proxy's side, thus no policy synchronization is necessary at the mobile service's side.

3.5. Security Protocols

In this subsection, we show Splendor's security protocols for authentication, confidentiality, integrity, and non-repudiation. As shown in Figure 1 (b), the pairs of components that communicate are: clients and directories, services and proxies, proxies and directories, clients and proxies, and clients and services. We show directories' announcements and the protocols among these pairs in Figure 3.

- Directory announcements

Using multicast addresses, directories periodically announce messages including their certificates, unicast network addresses, and signatures on the addresses, shown in Figure 3 (a). Clients verify the certificates before their service lookups. Services forward the messages to proxies and let proxies verify them.

- Proxy – Directory

Key transport between a proxy and a directory is a modification of X.509 strong two-way authentication [17] with some suggestions given by Menezes, et al [17]. (In (2), the certificates of the directory are not sent to the proxy, since the proxy has the certificates already.) The protocol authenticates both parties and exchanges session keys. A proxy represents a service and registers with a directory. First, the proxy verifies the directory's certificates. Then, it sends its certificates, a message, a session key encrypted using the directory's encryption public key and signs the message and the encrypted session key using its signing private key, as shown in Figure 3 (b). The directory verifies the proxy's certificates, signature, and the validity of the message. Then it replies with a message, another session key encrypted using the proxy's encryption public key, and signs the message and the encrypted session key. Next, the proxy does a similar check.

Proceedings of the First IEEE International Conference on Pervasive Computing and Communications (PerCom'03)
0-7695-1895/03 $17.00 © 2003 IEEE

- Client-Directory

A client checks the directory's certificates when it receives them. If a client inquiries a directory, it sends a query to the directory with its certificates, a secret session key encrypted using the directory's public key, as shown in Figure 3 (c). The directory verifies the client's certificates and then records the session key. This is half of the X.509 strong two-way authentication [17], which only the client shows its authenticity, due to the client has already verified the directory's certificates and it implicitly authenticates the directory in the following messages exchange between them.

- Service - Proxy

The key transport between a service and a proxy may be based on public-key encryption as the techniques we use for key transport between a proxy and a directory. In some situations, a service provider is a user of its proxy and the proxy is trustworthy. To offload tasks from mobile services such as services on PDAs, we provide an alternative solution, which uses symmetric encryption techniques. We use a symmetric key derived from the service provider's password.

When a service moves to a place and wants to register with a directory, it sends a timestamp, a message requesting to register with a directory, the directory's multicast message encrypted using the derived symmetric key shared with its proxy [17], as shown in Figure 3 (d). The proxy replies it with a message. We may optionally let the service provider type in a password to make sure that he is still in the possession of the PDA.

- Client-Proxy

After receiving a matched service list from a directory, a client selects and contacts a service or a proxy. If the client contacts a proxy, it verifies the certificates of the proxy that the directory sent along in the matched list. Next, the client and the proxy authenticate, generate, and transport a session key as proxies communicate with directories. Then the proxy checks the access permission and grants a privilege to the client. The proxy also generates a session key to be used between the client and the service as shown in Figure 3 (e). If the service does not want to use this session key, it may generate a session key itself, which is encrypted using the client's encryption public key.

Thus, we have shown that all the communicating pairs share session keys. After that, all the communication data are encrypted using session keys. For non-repudiation purpose, data are hashed and hashes are signed before encryption. We show these in Figure 3 (f) and (g).

3.6. Privacy Issues

Very few service discovery protocols have considered privacy issues [2]. In SSDS, communication data are encrypted to prevent information from being exposed to eavesdroppers. In Splendor, we also encrypt communication data. In the M911 scenario, the communications are confidential, which not only prevent eavesdropping, but also avoid exposure to the parties that do not need to know. For example, Patrick's medical history is not exposed to the shopping mall computer systems in a medical emergency situation, because the shopping mall system does not know the session key shared between Patrick and David.

Furthermore, we choose to use a passive location sensing system, so only users are aware of their location, and the location sensing system is not aware of its users. Thus, users' location information is kept private until users want to release their positions.

In the Splendor service discovery model, hiding the identity of a service provider is quite easy. We use the M911 scenario as an example. A physician may go to a shopping mall many times, but very few times there are emergency situations. Providing physicians' help in an emergency at the price of exposing their private information to the directories is not ideal. Splendor uses proxies to provide privacy for mobile services. Before registering with directories, proxies may generate names that only make sense to the proxies themselves for the services. In M911, a proxy registers a physician with a directory as "ABC hospital service provider 1001," instead of registering the physician's real ID. Only proxies keep the mappings from the registered service names to the services. Therefore, the directories are unable to know the actual service providers and they are only aware that a service provider is from a known proxy. The proxies, however, are responsible for the services that they represent and register with the directories. Directories may require proxies to sign their messages to assure proxies' responsibility. If proxies do hide the identities of the services, the directories do not reply to clients with matched services, but with proxies instead.

4. Performance Analysis and Evaluation

We analyze the overhead of adding proxies, providing privacy support, location-awareness integration, and security protocols in Splendor. First, the overhead of the indirection caused by adding proxies is small. The difference between the client-service-directory model and the client-service-directory-proxy model is that in the former model, all communications are within a few network hops, but for the latter model, a number of messages may be sent over the Internet. Nevertheless, the round trip delay over the Internet is not critical for most of the applications at the service lookup stage. Second, the addition of proxies is transparent to the directories. Clients, however, feel the indirection: they mutually authenticate with proxies, but then communicate with services. If mobile services do not generate session keys themselves, there are the same numbers of mutual authentications in the client-service-directory-proxy model and the client-service-directory model. Third, if there are more than one service matched to a client's request and the proxy does not select a matched service for the client, there is another round of message exchange, in which the proxy lets the client pick one service. Fourth, integrating location awareness has overhead for mobile clients and services, but reading location tag information does not have overhead on the critical path of the service discovery processes.

Our software running on PDAs are developed using Microsoft eMbedded Visual C++ 3.0. We measured

Proceedings of the First IEEE International Conference on Pervasive Computing and Communications (PerCom'03)
0-7695-1893-1/03 $17.00 © 2003 IEEE

Table 1. Public key and symmetric key operation overhead.

	PC	iPAQ
Key generation		
RSA encryption public key pair	253ms	395ms
RSA 512-bit signature public key pair	86ms	386ms
DES 64-bit session key	0.03ms	0.18ms
Encryption		
RSA public key encrypting a DES 64 bit session key	247ms	373ms
DES 64-bit symmetric key encrypting a 1K bytes message	0.07ms	0.89ms
Decryption		
RSA private key decrypting a DES 64 bit session key	7.55ms	15ms
DES 64-bit symmetric key decrypting a 1K bytes message	0.07ms	0.82ms
Signature		
Hashing a 1K bytes message and RSA 512-bit signature private key signing	1.42ms	15.5ms
Hashing a 1K bytes message and RSA 512-bit signature public key verification	0.13ms	1.47ms

average overhead of 1000 security operations for mobile clients and services on a ARM SA1110 206 MHz computer (Compaq iPAQ) with 64MB memory running Microsoft PocketPC 3.0. Other software is developed using Microsoft Visual Studio .NET 1.0. We measured average overhead of 1000 security operations for proxies and directories on an Intel Pentium III 866MHz computer with 192MB memory running Windows 2000 Professional. The cryptography software packages, which we use, come with the development tools.

As we see in Table 1, these security operations are fast. The longest operations are the public key operations, which take hundreds of milliseconds. Public key pairs are generated once and used for a long time. Session keys are generated and encrypted using encryption public keys in the session setup stage; the overhead is less than 400 ms. After sessions are setup, messages are exchanged many times, but only symmetric key encryptions are necessary, which take less than less 1ms for 1KB messages. For parties, which need to exchange messages with signatures for the purpose of non-repudiation, the overhead is less than 20ms for 1KB messages. (We choose 1024-bit RSA encryption keys on PC, but 512-bit on iPAQs, because it is limited by the software package that we use for iPAQ.)

We have discussed certificate validation in Section 3.4. Although we do not discuss PKI here, certificate validation may affect performance. Various trust models and certificate revocation mechanisms affect performance differently. Detailed discussion of PKI trust models may be found in [19] and certificate revocation may be found in [19, 20]. We assume that for different applications different trust models may be used. For example, Secure Electronic Transaction (SET) has a hierarchical model for which the certificate validation is efficient [21]. RSA

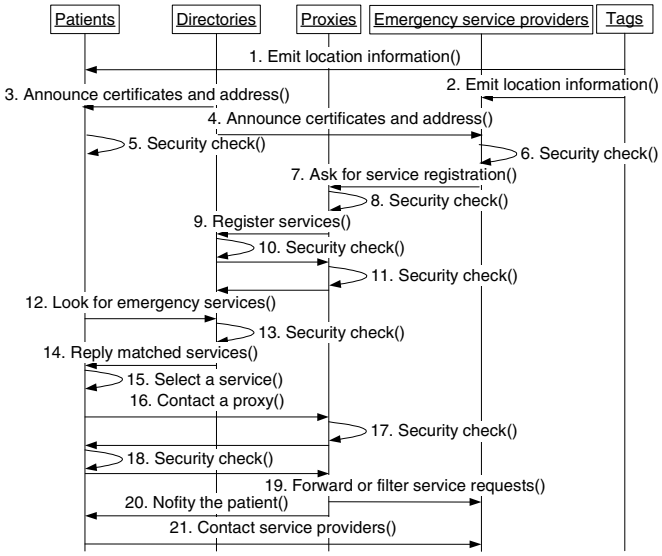

Figure 4. Interaction of the four parties in M911. (10,11 and 17,18 are mutual security checks.)

Research tested its CA product on eight million certificates, and the average time of online certificates status checking in the tests is less than 1 second [22]. This result gives us a good estimate of certificate status checking for such a large numbers of certificates.

4.1. The Critical Path of M911

The interaction among the four parties in M911 is shown in Figure 4. Let's consider the critical path of M911 -- from the time that a patient has a request to the time that the patient's PDA exchange information with the physician's PDA. It is from step 12 to step 21.

We further look into the critical path of M911. In Figure 5, we define the time of the major operations and then give the response time of the critical path. The operation of certificate validation and the operation of service matching at directories and proxies are parallel. Using the data discussed above, we estimate that the response time of the critical path in M911 should be in a reasonable time.

One possible improvement is that the directory sends a confirmation to the client that the services' authenticities are checked, so the client does not need to verify the certificates itself. This helps the patient's PDA set up connections with the services. Thus, step 18 (TV2) is removed and it will give us a better response time.

5. Conclusion and Future work

In this paper, we discussed a new service discovery model, Splendor, which supports nomadic users in public environments. Splendor offers mutual authentication among components; simplifies service authorization; provides communication confidentiality and message integrity; and supports non-repudiation. User privacy, data privacy, and user location privacy are achieved. The security protocols also are designed to protect against

Proceedings of the First IEEE International Conference on Pervasive Computing and Communications (PerCom'03)
0-7695-1893-1/03 $17.00 © 2003 IEEE

TQ1 - time interval that a client pushes a button and a directory receives the query message.
TQ2 - time interval that the client sends out a service request message and a proxy receives.
TR1 - time interval that the directory sends out the client's query result and the client receives.
TR2 - time interval that the proxy sends out a service's network address and the client receives.
TS1 - time interval that the directory finds a list of matched services or proxies then encrypted the reply message.
TS2 - time interval that the proxy finds a matched service then encrypted the reply message.
TV1 - time interval that the directory verifies the authenticity and status of the client's certificates.
TV2 - time interval that the client verifies the authenticity and status of the proxy's certificates.
TV3 - time interval that the proxy verifies the authenticity and status of the client's certificates.
TC - the response time of the critical path.
TC = TQ1 + max (TS1, TV1) + TR1+ TV2+ TQ2 + max (TS2, TV3) + TR2

Figure 5. The critical path response time of M911.

attacks such as eavesdroppers and replay attacks. Location-awareness is integrated to our service discovery protocol to support location dependent service better and reduce the requirements of the underlying network infrastructure. We are working on extending the capability of proxies to support service discovery in other pervasive computing environments. We will also design different service authorization strategies to support different types of users with different service requirements.

Reference

[1] T. Kindberg and A. Fox, "System Software for Ubiquitous Computing," IEEE Pervasive Computing, January-March, 2002, pp. 70-81.

[2] F. Zhu, M. Mutka, and L. Ni, Classification of Service Discovery in Pervasive Computing Environments, MSU-CSE-02-24, Michigan State University, East Lansing, 2002.

[3] "Specification of the Bluetooth System -- Core," Bluetooth SIG, Version 1.1, February 22, 2001, available at http://www.bluetooth.org/docs/Bluetooth_V11_Core_22Feb01.pdf.

[4] M. Nidd, "Service Discovery in DEAPspace," IEEE Personal Communications, August, 2001, pp. 39-45.

[5] W. Adjie-Winoto, E. Schwartz, H. Balakrishnan, and J. Lilley, "The design and implementation of an intentional naming system," 17th ACM Symposium on Operating Systems Principles (SOSP '99), Kiawah Island, SC, 1999.

[6] M. Balazinska, H. Balakrishnan, and D. Karger, "INS/Twine: A Scalable Peer-to-Peer Architecture for Intentional Resource Discovery," Pervasive 2002 - International Conference on Pervasive Computing, Springer-Verlag, Zurich, Switzerland, 2002.

[7] "Jini™ Architecture Specification," Sun Microsystems, Version 1.2, December, 2001, available at http://wwws.sun.com/software/jini/specs/.

[8] "Salutation Architecture Specification," Salutation Consortium, Version 2.0c, June 1, 1999, available at ftp://ftp.salutation.org/salute/sa20e1a21.ps.

[9] S. Czerwinski, B. Y. Zhao, T. Hodes, A. Joseph, and R. Katz, "An Architecture for a Secure Service Discovery Service," Fifth Annual International Conference on Mobile Computing and Networks (MobiCom '99), Seattle, WA, 1999, pp. 24-35.

[10] E. Guttman, C. Perkins, J. Veizades, and M. Day, "Service Location Protocol, Version 2," IETF, RFC2608, June 1999, available at http://www.ietf.org/rfc/rfc2608.txt.

[11] "Universal Plug and Play Device Architecture," Microsoft Corporation, Version 1.0, 08 June, 2000, available at http://www.upnp.org/download/UPnPDA10_20000613.htm.

[12] M. Weiser, "The Computer for the 21st Century," Scientific American, vol. 265, Issue 3, 1991, pp. 66-75.

[13] R. Want, A. Hopper, V. Falcão, and J. Gibbons, "The Active Badge Location System," ACM Transactions on Information Systems, vol. 10, 1, 1992, pp. 91-102.

[14] J. Hightower and G. Borriello, "Location Systems for Ubiquitous Computing," Computer, vol. 34, 8, 2001, pp. 57-66.

[15] A. Ward, Sensor-driven Computing, PhD thesis, Corpus Christi College, University of Cambridge, Cambridge, UK, 1998.

[16] "People, Places, Things: Web Presence for the Real World," T. Kindberg, J. Barton, J. Morgan, G. Becker, D. Caswell, P. Debaty, G. Gopal, M. Frid, V. Krishnan, H. Morris, J. Schettino, and B. Serra, available at http://cooltown.hp.com/dev/wpapers/index.asp.

[17] A. Menezes, P. v. Oorschot, and S. A. Vanstone, Handbook of Applied Cryptography, CRC Press, 1996.

[18] W. Stallings, Cryptography and Network Security: Principles and Practice, 2nd ed, Prentice Hall, 1998.

[19] S. Lloyd, C. Adams, and S. Kent, Understanding Public-Key Infrastructure: Concept, Standards, and deployment considerations, New Riders, 1999.

[20] D. Cooper, "A Model of Certificate Revocation," Fifteenth Annual Computer Security Applications Conference, 1999, pp. 256-264.

[21] "SET Secure Electronic Transaction Specification Book 2: Programmer's Guide," Visa and MasterCard, Version 1.0, May 31, 1997, available at http://www.setco.org/download/set_bk2.pdf.

[22] "Scalability Proof of Concept: RSA Keon Certificate Authority Eight Million Certificate Test," RSA Security Inc, 2002, available at http://www.rsasecurity.com/products/keon/whitepapers/kca/KCAS_WP_0702.pdf.

Proceedings of the First IEEE International Conference on Pervasive Computing and Communications (PerCom'03)
0-7695-1895/03 $17.00 © 2003 IEEE