Facilitating Secure Ad hoc Service Discovery in Public Environments *

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

Securely accessing unfamiliar services in public environments using ad hoc wireless networks is challenging. We present a proxy-based approach that uses other existing network channels to set up a secure and trust relationship between communication parties to facilitate ad hoc wireless communications. Based on a service discovery protocol, our models achieve secure, trusted, anonymous, efficient, and economical communications between unfamiliar parties. Our protocols are formally verified using BAN logic.

1. Introduction

Accessing unfamiliar services in public environments is becoming more realistic as we move towards ubiquitous computing environments. PDAs, cell phones, laptops are becoming commodities. Using these mobile devices to access public services enables computing everywhere. Let's look at the following scenario.

Bob is in an airport and he has an hour before his flight leaves. He turns on his PDA and finds that there is a wireless LAN available. However he does not subscribe to the service provider of the wireless LAN. Is there a simple and secure way for Bob to use the wireless LAN to surf the Web and read email, which is cheaper and faster than using a 3G connection? After Bob makes the connection, he receives an email that includes an attached document. Then, he uses his PDA to search for nearby printers to print the document, so he can read it during his flight.

There are two basic security problems when accessing unfamiliar services as in the above scenario. How is a trust relationship set up between two parties? How is a secure ad hoc wireless communication set up? One common vision for future computing is that everything is connected to the Internet. Public services such as wireless access points or printers are very likely to have Internet connections since the Internet connections enable these devices to be managed remotely. Meanwhile, many mobile devices may have more than one network channel, for example, 3G, IEEE802.11x, and/or Bluetooth. These channels not only enable devices to be connected to the Internet, but also enable them to communicate to other devices in the vicinity via ad hoc mode. Ad hoc mode is more efficient for many communications, such as what we have discussed in the above scenario. By using Internet channels, we may facilitate ad hoc communications in order to achieve inexpensive, fast, and secure communications. Unlike existing solution attempts, which seek to use pure ad hoc environments, we shift from pure secure ad hoc communication problems to secure ad hoc communications with assistance from other network connections. Our models are also designed to defend against many attacks, including attacks from malicious services. Moreover, based on service discovery protocols, our framework provides better usability.

In Section 2, we discuss work related to secure communications in pervasive environments, service discovery protocols, and proxy-based communications. Next in Section 3, we present our design of two secure and trusted models. In Section 4, we use BAN logic to verify our communication models formally. Last in Section 5, we conclude and discuss our future work.

2. Related Work

The Resurrecting Duckling security policy [1] provided a new way for authentication in ubiquitous computing environments. By mimicking the behavior of mother ducks and ducklings, the policy set up a master-slave relation between devices. The master-slave relation limited peer devices to talk to each other. Therefore, Stajano proposed additional research [2] to enable peer communications. The basic idea was that master devices might define policies, which allowed other devices to set up temporary master-slave relations to control the slave devices. The authors are also the first who proposed the idea of using physical contact to exchange a secret before two devices set up secure wireless communications.

Balfanz, et al. at Xerox Palo Alto Research Center, extended the Resurrecting Duckling security policy [3]. Their work tried to solve the authentication problem for securely using services without using a public key infrastructure and universal naming convention for printers via side channels. The authentication protocols

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

were based on public key cryptography or the Guy Fawkes protocol [4]. Our work solves similar problems, such as accessing public computing resources securely. Instead of using IrDA or physical contact as a side channel, we use other existing network connections for secure key exchanges. The Resurrecting Duckling security policy and Balfanz's work did not solve the problem of determining whether to trust an unfamiliar service within public environments? In other words, an unfamiliar service might be a malicious service. Our approach provides a means to protect against attacks from the services.

A proxy-based approach is one way to assist mobile applications. Our previous work proposed a proxy-based service discovery in infrastructure environments to offload tedious work from mobile services and provide privacy for them [5]. Burnside, et al. had another proxy-based solution for secure service discovery to enable low processing power devices in infrastructure environments to discover each other [6]. Langheinrich designed a proxy-based privacy-aware system [7]. This work also suggests proxy-based approaches to facilitate communication in ad hoc environments. Although the names of these approaches sound similar, to the best of our knowledge, none of the existing work solves the problems that we introduced in section 1.

UPnP was designed for unmanaged networking environments [8]. UPnP is a device oriented two-party (client-service) service discovery protocol. In UPnP, either clients learn from services' periodical announcements or clients actively query for services (announcement-query approach). Nidd studied and proposed another service discovery protocol for single-hop ad hoc environments, known as DEAPspace [9]. In contrast to other existing service discovery protocols that use the announcement-query based approach, DEAPspace uses a cache-broadcast approach. Each node caches service information, and then each node broadcasts its knowledge of other services and its own services in turn. The nodes learn from others. Service lookup is accomplished by searching the local cache. For other two-party service discovery protocols, such as Salutation [10], Service Location Protocol (SLP) Version 2 [11], and Bluetooth Service Discovery Protocol [12], the discovery mechanisms are similar to UPnP. A detailed comparison was provided in [13].

3. System Design

In this section, we first discuss many possible threats and attacks, which we take into consideration when we design our models. Then we show our service discovery protocol. Later, we illustrate our two communication models and the security protocols.

3.1. Threats and Attacks

Securely accessing unfamiliar services in public environments is more challenging than conventional service accesses. Public services may not have and maintain user information and users do not have accounts for service accesses. Unfamiliar services might be dishonest or even malicious. Furthermore, users might take "free rides" or even "break" the services.

We consider the threats of disclosure, integrity, and denial of service (DOS) [14]. It is easy for services to detect DOS. If someone jams the wireless channel, the attack may be reported through other network links. If a user abuses a resource, there is no need to do anything as long as there is a service charge for a service access. In order to protect against eavesdroppers, we use cryptographic technology to encrypt messages. However, it becomes trickier if there are fake services, which allure users and collect users' information, but do not actually provide services. Even normal services may record user information.

For active attacks, we consider the man-in-the-middle attack and the message replay attack. Furthermore, we consider situations when unfamiliar services or service providers may initiate attacks on users. Meanwhile we also consider cases when malicious users attack services. We describe more details of how we protect against threats and attacks when we discuss our models.

3.2. Secure Ad hoc Service Discovery

We only consider service discovery in single hop ad hoc networks in this paper. As discussed in Section 2, the announcement-query and cache-broadcast approaches represent two methods for service discovery within ad hoc environments. When there is more than one service provider in a public environment, it is more reasonable to use the announcement-query approach since there is no incentive for services to broadcast service information for other service providers. However, it is more efficient for services, which are from the same service provider, to broadcast in turn and share the load of broadcast. For services from the same service provider (sibling services), they cache each other's service information and take turns to broadcast their knowledge of available services. As long as a sibling service broadcasts a service announcement message, which contains correct information of its service, the service will not broadcast again itself. We recommend that the rate of the service announcement should be kept at a low frequency; otherwise services from different service providers might compete against each other and jam the wireless channel by sending out service announcements.

Instead of learning the available services by listening to service announcements, a client may send query messages. Comparing the attributes in the query with its own service attributes, only the matched services reply to the client. A client may also search for all the available services in the vicinity by sending a wildcard query. If more than one service from the same service provider matched the query, then the service, which last announced, replies with a message that contains the set of matched services. Then a client or user picks a service to use.

Each service's state is a soft state. In other words, each service has a life span and will be invalid after its lifespan. To continue providing its service, a service announces its new lifespan before the service expires.

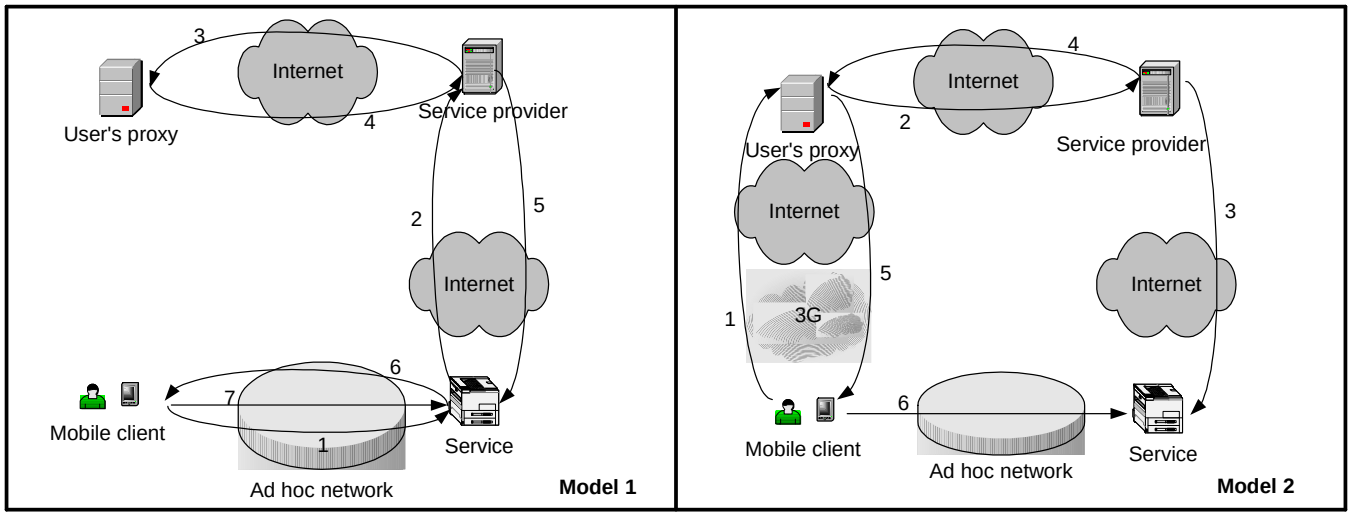

Figure 1. Two secure communication models to set up secure ad hoc communications between clients and services.

3.3. Communication models

While a mobile client wants to use an unfamiliar service, it is difficult to exchange a key securely in ad hoc environments via a wireless radio frequency channel. Eavesdroppers may learn keys. Likewise, it is difficult to prevent the man-in-the-middle attack. As we discussed in Section 2, physical contact or using location-limited channels are two approaches to exchange a key securely. However, the usability decreases, because users need to learn to use these approaches and different devices may have different interfaces. We suggest using a proxy-based approach, which not only facilitates authentication, but simplifies usage as well.

There are four parties in our communication models: mobile clients, user proxies, services, and service providers as shown in Figure 1. We assume that services have wireless ad hoc communication channels, via which mobile clients in the vicinity may access the services. They also have Internet connections, so service providers may manage the services remotely. A user's mobile device may have more than a wireless LAN capability, for example the device also has a 3G connection. To access an unfamiliar service, we have two models – one uses 3G channels and one does not. We show the two models in Figure 1 and discuss them in detail shortly.

Both models use user proxies to assist mobile users. The user proxy is a program running on a machine, which is connected to the Internet. The machine could be a home PC or a server from a service provider running thousands of proxies for users. The user proxy is designed to fulfill the following functions, which are difficult to achieve in pure ad hoc environments. First, there is a need to verify that a service is the service that it claims to be. Our approach is based on public key cryptography and Public Key Infrastructure (PKI). Detailed PKI information may be found in [15], which we do not discuss here. In our models, service providers have public key certificates, but services do not have them. The user's proxy checks whether the service provider's certificate is valid and used for the right purpose. In addition, the service providers assure the mobile clients and the users' proxies that their services are honest. By using certificates, we defend against the chosen protocol attack [16]. Second, the user's proxy is used as a safe guard. Every service request from a mobile client goes through its proxy. In case a mobile device is lost, the user may disable the mobile device to access any services via the proxy. Likewise, if a mobile device's encryption key, which is shared between the mobile client and the user's proxy, is compromised and used for services by a hacker, we may discover it by examining the log on the proxy. Thus, a key revocation is simple. Third, more complicated service negotiations such as TrustBuilder may be deployed [17].

A service provider manages services and handles service authorization. Since all the mobile clients use the services temporarily, service authorization is lease-based and only valid for a certain time period. The service provider sends a ticket to the mobile client and a copy to the service. Therefore, services only need to handle a few service access levels. As long as a client's ticket matches the service's copy, access is granted. The service providers imply its assurance of their services when issuing the tickets. In this way, users have more confidence to use unfamiliar services and the service providers have a simple way to stop tempered services and revoke compromised keys shared with the services.

To simplify the discussion of the two models, we suppose that a client has discovered a desired service to access. Next, we discuss the different procedures of the two models to set up secure ad hoc communication channels.

- Model 1: accessing an unfamiliar service without a 3G connection.

Notation: C is a mobile client; S is a service; P is a service provider; U is a user's proxy. t_X or $t_{X\#}$ is a timestamp, which X attaches. T_X is the expiration time of the service for a client to access, which X attaches. $CertE_X$ is an encryption public key certificate of X. $CertV_X$ is a verification public key certificate of X. K_{XY} is a symmetric encryption key shared between X and Y. $(\,)_{KXY}$ is an encryption using symmetric key K shared between X and Y. $(\,)_{KX}$ is an encryption using the public encryption key of X. $(\,)_{KX}^{-1}$ is X's signature using its signing private key. G_X is a granted privilege to X. A_{XY} is the access code for X to access Y. M is a message.

Step	From→To	Message
1	C→S:	$C, t_{C2}, U, (S, P, t_C)_{KCU}$
2	S→P:	$S, (C, U, t_S)_{KSP}, (S, P, t_C)_{KCU}$
3	P→U:	$P, CertE_P, t_P, (S, P, t_C)_{KCU}$
4	U→P:	$(t_P, K_{CS})_{KP}, (K_{CS}, t_C)_{KCU}, CertV_U, (C, S, P, t_P, K_{CS})_{KU}^{-1}$
5	P→S:	$(K_{CS}, t_C)_{KCU}, (C, G_C, T_P, t_S, K_{CS})_{KSP}$
6	S→C:	$t_{S2}, (K_{CS}, t_C)_{KCU}, (t_{C2}, G_C, T_P)_{KCS}$
7	C→S:	$C, (t_{S2}, M)_{KCS}$

(a). Model 1: accessing an unfamiliar service without a 3G connection.

Step	From→To	Message
1	C→U:	$C, CertE_P, (S, P, t_C)_{KCU}$
2	U→P:	$U, S, (t_U, K_{CS}, K_{UP})_{KP}, CertV_U, (U, S, P, t_U, K_{CS}, K_{UP})_{KU}^{-1}$
3	P→S:	$(A_{CS}, K_{CS}, G_C, t_P, T_P)_{KSP}$
4	P→U:	$P, (t_U, A_{CS}, T_P)_{KUP}$
5	U→C:	$(t_C, A_{CS}, K_{CS}, T_P)_{KCU}$
6	C→S:	$(A_{CS}, M)_{KCS}$

(b). Model 2: accessing an unfamiliar service with a 3G connection.

Figure 2. Security protocols of the two models.

In this model, a mobile client only has the wireless ad hoc communication channel. In order to communicate with its proxy, the mobile client has to use the Internet connection that the service has and sends a message to its proxy via a service and a service provider. We outline the interaction in Model 1 of Figure 1. First, a client sends a service request message to a service (step 1). Then the service generates its copy of the service request and sends to the service provider along with the message from the client (step 2). Next, the service provider forwards the message to the user's proxy. Along with the message, the service provider also sends its certificate (step 3). After verifying the service provider's certificate, the user's proxy generates a session key for the mobile client and the service. Afterwards, the proxy puts the session key in two copies: one copy for the mobile client, which also implies that the service provider is sound and the service provider has assured the service; another copy for the service provider within its request for accessing the service. The first copy is encrypted using the key shared between the mobile client and its proxy. The other copy is encrypted using the service provider's public key, which is in the service provider's certificate (step 4). When receiving messages from the user's proxy, the service provider decrypts the session key and re-encrypts it using the key shared between the service and the service provider. Then, it sends the session key and the access authorization to the service (step 5). Last, the service forwards the session key from the user's proxy to the mobile client (step 6). Now the mobile client and the service share a key for secure communication (step 7). We show this protocol in Figure 2 (a). (We use a notation similar to the BAN logic notation [18].)

Furthermore, when the mobile client sends a service request to its proxy, the message is encrypted using a key, which is shared between the mobile client and the user's proxy beforehand. This encrypted message protects the user from dishonest services, which might alter the service requests.

One possible attack is that a mobile client does not actually access any services but takes a "free ride" and sends packets to a machine on the Internet via step 1, 2, and 3 of Model 1 in Figure 1. To guard against this attack and protect services and service providers, service providers check the address of the destination proxy to prevent free rides.

The drawback of the model is that privacy information of the user and user proxy may be sacrificed. An eavesdropper or a service from a competitive service provider may learn information about users, their proxies, and other services from the interaction. Meanwhile, an extra load is placed on the services and the service providers to forward messages.

- Model 2: accessing an unfamiliar service with a 3G connection.

In this model, the client's mobile device also has a 3G connection. Instead of communicating through the service, the mobile client directly contacts its proxy via a 3G connection. This model provides a more efficient way of communication than Model 1, while incurring the price of the 3G-connection cost. However, the cost is very low: only two messages are required for each service access. Additionally, it is more difficult for eavesdroppers to listen on the 3G and wireless ad hoc channels at the same time (the 3G connection is encrypted [16]).

In the service discovery process, a mobile client learns a service provider's certificates. Along with its service request messages, the client also forwards the certificates to its proxy (step 1 in Model 2 of Figure 1). After verifying the certificates, the user's proxy contacts the service provider (step 2). If the access is granted, the service provider sends an authorization message to the service first (step 3) and then sends another message to the user's proxy (step 4). Last, the user's proxy forwards the session key, which is used between the mobile client and the service (step 5). Thus, the mobile client is ready to access the service (step 6). We show the interaction of Model 2 in Figure 2 (b).

Step	From→To	Message
1	C→S:	$\{CSP_C, t_C\}_{KCU}$
2	S→P:	$\{CSP_S, t_S\}_{KSP}$, $\{CSP_C, t_C\}_{KCU}$
3	P→U:	$\{ \underrightarrow{K} P, P\}_{KCA^{-1}}$, $\{CSP_C, t_C\}_{KCU}$
4	U→P:	$\{C \underleftrightarrow{K_{CS}} S, \#(C \underleftrightarrow{K_{CS}} S)\}_{KP}$, $\{C \underleftrightarrow{K_{CS}} S, \#(C \underleftrightarrow{K_{CS}} S)\}_{KCU}$, $\{CSP_U, \#(CSP_U), C \underleftrightarrow{K_{CS}} S\}_{KU^{-1}}$, $\{ \underrightarrow{K} U, U\}_{KCA^{-1}}$
5	P→S:	$\{C \underleftrightarrow{K_{CS}} S, \#(C \underleftrightarrow{K_{CS}} S)\}_{KSP}$, $\{C \underleftrightarrow{K_{CS}} S, \#(C \underleftrightarrow{K_{CS}} S)\}_{KCU}$,
6	S→C:	$\{C \underleftrightarrow{K_{CS}} S, \#(C \underleftrightarrow{K_{CS}} S)\}_{KCS}$ from S, $\{C \underleftrightarrow{K_{CS}} S, \#(C \underleftrightarrow{K_{CS}} S)\}_{KCU}$
7	C→S:	$\{C \underleftrightarrow{K_{CS}} S, \#(C \underleftrightarrow{K_{CS}} S)\}_{KCS}$ from C

(a). Idealize protocol of Model 1. (CSP_X is a service request message, which X generates.)

1	$C \models C \underleftrightarrow{K_{CU}} U$, $U \models C \underleftrightarrow{K_{CU}} U$, $U \models C \underleftrightarrow{K_{CS}} S$, $S \models S \underleftrightarrow{K_{SP}} P$, $P \models S \underleftrightarrow{K_{SP}} P$
2	$P \models \underrightarrow{K_P} P$, $U \models \underrightarrow{K_U} U$, $P \models \underrightarrow{K_{CA}} CA$, $U \models \underrightarrow{K_{CA}} CA$, $P \models CA \mapsto \underrightarrow{K_U} U$, $U \models CA \mapsto \underrightarrow{K_P} P$
3	$C \models (U \mapsto C \underleftrightarrow{K_{CS}} S)$, $P \models (U \mapsto C \underleftrightarrow{K_{CS}} S)$, $S \models (U \mapsto C \underleftrightarrow{K_{CS}} S)$, $S \models (P \mapsto U \models C \underleftrightarrow{K_{CS}} S)$, $U \models (C \mapsto CSP_C)$, $P \models (S \mapsto CSP_S)$, $P \models (U \mapsto CSP_U)$
4	$C \models \#(t_C)$, $C \models \#(t_{C2})$, $S \models \#(t_S)$, $S \models \#(t_{S2})$, $P \models \#(t_P)$, $P \models \#(T_P)$, $U \models \#(t_C)$, $P \models \#(t_S)$

(b). Assumptions of Model 1.

After 1	$S \lhd \{CSP_C, t_C\}_{KCU}$
After 2	$P \lhd \{CSP_C, t_C\}_{KCU}$, $P \models CSP_S$
After 3	$U \models CSP_C$, $U \models \underrightarrow{K_P} P$
After 4	$P \lhd \{C \underleftrightarrow{K_{CS}} S, \#(C \underleftrightarrow{K_{CS}} S)\}_{KCU}$, $P \models \underrightarrow{K_U} U$, $P \models CSP_U$, $P \models C \underleftrightarrow{K_{CS}} S$
After 5	$S \models C \underleftrightarrow{K_{CS}} S$, $S \lhd \{C \underleftrightarrow{K_{CS}} S, \#(C \underleftrightarrow{K_{CS}} S)\}_{KCU}$
After 6	$C \models C \underleftrightarrow{K_{CS}} S$, $C \models S \models C \underleftrightarrow{K_{CS}} S$
After 7	$S \models C \models C \underleftrightarrow{K_{CS}} S$
Result	$C \models C \underleftrightarrow{K_{CS}} S$, $C \models S \models C \underleftrightarrow{K_{CS}} S$, $S \models C \underleftrightarrow{K_{CS}} S$, $S \models C \models C \underleftrightarrow{K_{CS}} S$

(c). Results after each step of Model 1.

Figure 3. Formal verification of Model 1 using BAN logic.

In comparison to Model 1, access control is simplified. The service provider does not differentiate which mobile client accesses the service, but only records which user's proxy asks for the service. The service provider generates a service access code. Using the service access code, a client obtains the right to use the service.

4. Protocols analysis and Formal Verification

During the several rounds of design and verification processes, we used BAN logic [18] to verify our security protocols formally and mechanically. It helps us make our protocol succinct and facilitates us to find subtle bugs. Moreover, it facilitates us to express assumptions more explicitly and to present protocols more clearly. When we next discuss the details of the verification, only Model 1 is used as an example.

The detailed BAN logic notation and rules explanation may be found in [18]. Our idealized protocol is shown in Figure 3 (a). Each step in the actual protocol (shown in Figure 2 (a)) is mapped to a step in the idealized protocol. As the convention of the idealize protocol in BAN logic, we leave out the clear text information, because it may be modified by an adversary party. The idealized protocol has the same goal as the actual protocol.

We present our assumptions about the protocols in Figure 3 (b). The first row states that the communication pairs trust their shared keys, while the second row declares the trust of the public keys. In row 3, we list the assumptions that a party trusts another party, who has control over the key creation or the message creation. The suspicious assumptions are that a service and its provider believe that a user's proxy will correctly create a session key for its client and the service. From the service provider's point of view, as long as a user's proxy signs or pays for the transaction, it believes that the user's proxy will generate good keys. An alternative approach is that the service provider creates the session key, but this will introduce another two messages between the service provider and the user's proxy. The last row of the assumptions is about using timestamps as fresh nonce. Synchronized clocks are required between a mobile client and its proxy and between a service and its provider. In our implementation, the two pairs synchronize clocks.

Now, we are ready to deduct from assumptions to conclusions. The deduction itself is lengthy. Thus, we only discuss intermediate results after each step as shown in Figure 3 (c). After step 1, a service sees an encrypted message from a client to its proxy, but the service is not

able to see the content. Then after the second step, by using the message-mean, nonce-verification, and jurisdiction rules, a service provider believes that a client requests to access its service. We also base our deduction on the assumption that the request, which the service provider sees, is a fresh service request from its service (not a replay message). As we have discussed above, synchronized clocks are required in this step. The service provider also sees the encrypted message forwarded from the service. During the processing of the messages in step 3, the user's proxy validates the service provider's certificate. (We omit the details of certificate verification, and suppose that the certificate is confirmed to be correct.) Meanwhile, based on a similar deduction as we discussed in process of step 2, the proxy believes that the client requests a service access.

Several deductions are needed after the service provider receives the messages in step 4. First, since the service provider possesses its private key, it sees the session key for the client and the service from the user's proxy (other rule). Next, the service provider verifies the proxy's certificate and validates the signature of the proxy. Last, we repeatedly use message-meaning for public key, nonce-verification, and jurisdiction rules along with the assumption that the user's proxy creates the session key correctly, we derive that the service provider believes the session key. The service provider also forwards an encrypted message from the user's proxy to the client. After step 5, the service learns the session key from the service provider's message. Meanwhile, it sees an encrypted message for the client. Afterward, from the sixth step, the client not only gets its copy of the session key, but also learns that the service believes the session key. Finally after step 7, the client starts to use the service. It is therefore straightforward that the service believes that the client believes the session key. In summary, we come to a strong conclusion that the client and the service believe the session respectively and believe that each other believes the session key respectively.

5. Conclusion and Future work

We presented a proxy-based approach to facilitate ad hoc communications in public environments, based on a service discovery protocol. To access unfamiliar public services securely, we proposed two models. The models utilize existing Internet connections to setup trust relationships and exchange security keys while keeping efficient ad hoc communications. We formally verified and improved our security protocols using BAN logic.

An ongoing work is to design and prototype models without using PKI, since many devices or services may not have certificates. The two models that are discussed in this paper are master-slave relationships. The new models focus on facilitating secure peer-to-peer communications. We are also going to experiment with heterogeneous environments, which have coexisting infrastructure and ad hoc service discovery.

Reference

[1] F. Stajano and R. Anderson, "The Resurrecting Duckling: Security Issues for Ad-hoc Wireless Networks," 7th International Workshop on Security protocols, Cambridge, UK, 1999.

[2] F. Stajano and R. Anderson, "The Resurrecting Duckling -- what next?," 8th International Workshop on Security protocols, Cambridge, UK, 2000.

[3] D. Balfanz, D. K. Smetters, P. Stewart, and H. C. Wong, "Talking to Strangers: Authentication in Ad-Hoc Wireless Networks," 9th Annual Network and Distributed System Security Symposium, San Diego, CA, 2002.

[4] R. Anderson, F. Bergadano, B. Crispo, J.-H. Lee, C. Manifavas, and R. Needham, "A New Family of Authentication Protocols," Operating Systems Review, 1998.

[5] F. Zhu, M. Mutka, and L. Ni, "Splendor: A Secure, Private, and Location-aware Service Discovery Protocol Supporting Mobile Services," 1st IEEE Annual Conference on Pervasive Computing and Communications,IEEE Computer Society Press, Fort Worth, Texas, 2003.

[6] M. Burnside, D. Clarke, T. Mills, S. Devadas, and R. Rivest, "Proxy-Based Security Protocols in Networked Mobile Devices," 17th ACM Symposium on Applied Computing, Madrid, Spain, 2002.

[7] M. Langheinrich, "A Privacy Awareness System for Ubiquitous Computing Environments," UbiComp 2002, GÖTEBORG, SWEDEN, 2002.

[8] B. A. Miller, T. Nixon, C. Tai, and M. D. Wood, "Home Networking with Universal Plug and PIay," IEEE Communications Magazine, December, 2001, pp. 104-109.

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

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

[11] 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.

[12] "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.

[13] 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.

[14] E. Amoroso, Fundamentals of Computer Security Technology New Jersey, PRT Prentice Hall, 1994.

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

[16] R. Anderson, Security Engineering: A Guide to Building Dependable Distributed Systems, John Wiley & Sons, 2001.

[17] M. Winslett, T. Yu, K. E. Seamons, A. Hess, J. Jacobson, R. Jarvis, B. Smith, and L. Yu, "Negotiating Trust on the Web," IEEE Internet Computing, Nov-Dec, 2002, pp. 30-37.

[18] M. Burrows, M. Abadi, and R. Needham, "A Logic of Authentication," ACM Transactions on Computer Systems, 1990.