pKASSO†: Towards Seamless Authentication providing Non-Repudiation on Resource-Constrained Devices

Ki-Woong Park, Hyunchul Seok and Kyu-Ho Park
Computer Engineering Research Laboratory
Department of Electrical Engineering and Computer Science
Korea Advanced Institute of Science and Technology
{woongbak,hcseok,kpark}@core.kaist.ac.kr

Abstract

PKI is generally considered as the most appropriate solution for e-commerce and mutual authentication, owing to its digital signature and non-repudiation features. Asymmetric key operations of PKI require by far more CPU cycles than a symmetric cryptographic algorithm. It hampers the usability of PKI on resource-constrained devices. To overcome these limitations, we propose a new PKI-based authentication protocol and security infrastructure enhanced with single sign-on and delegation technology for a device with a restricted computing power. Although a conventional delegation mechanism cannot support non-repudiation mechanism against malicious user's behavior, our proposed protocol and security infrastructure can provide the mechanism by devising a referee server that generates binding information between a device and authentication messages, and retains the information in its local storage for future accusation.

1. Introduction

Authentication systems play a key role in achieving high security so that various ubiquitous services have been built on the top of the system. In our ubiquitous environment, we have been developing a wearable computer[1] and its interoperable computing environments where various devices such as U-Kiosks, U-Print, Zigbee-enabled appliances, etc are deployed to provide a user with ubiquitous services, mainly focusing on a university campus [1, 8]. In the ubiquitous computing environment, pervasive devices frequently take part in communication with other previously unknown devices for providing services, exposing themselves to an unfortified and insecure environment. Therefore, the importance of security in the ubiquitous computing environment cannot be overstated; it needs to be addressed in order to attain privacy and confidentiality of a user living in the environment [2]. As a fundamental way to enable security, authentication and authorization are the two most widely used mechanisms among devices.

In the ubiquitous environment that makes connections between service devices and users dynamically, authentication, authorization, and accounting services should be provided by a security infrastructure [3]. To offer these services, PKI is generally considered as the most appropriate solution for the requirements. However, the computational complexity of PKI results in high deployment costs and operation overhead since those operations are performed on resource-constrained devices. Besides, the frequency of the users' authentication request increases rapidly along with the number of the service devices that require mutual authentication [5].

By investigating our ubiquitous environment and conventional security systems thoroughly, we present limitations of conventional security system and our approach as follows.

1. Obstructive Authentication Latency in Low Computing Power Device: There is a challenge with thin clients where the restricted computing power device demands frequent and dynamic authentications over PKI. In order to conquer this disadvantage, we propose a security infrastructure which is based on PKI and a Single Sign-On(SSO) protocol for a cost effective diminutive security device. The proposed SSO protocol named pKASSO[2] provides users with a secure and seamless security mechanism using a delegation technology and a non-repudiation mechanism by devising a referee server that generates binding information between a device and authentication messages. Also, using the proposed authentication mechanism, we can achieve the more secure mechanism in accordance with the security policy requirement without replacing devices.

[1] UFC : Ubiquitous Fashionable Computer [1]

[2] † pKASSO: Public Key-based A^3-providing Single Sign On

21st International Conference on
Advanced Information Networking and Applications Workshops (AINAW'07)
0-7695-2847-3/07 $20.00 © 2007 IEEE

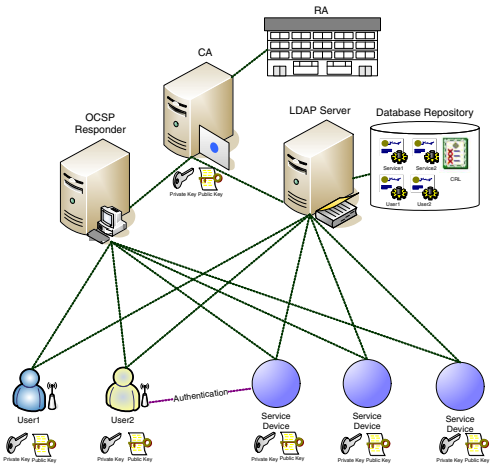

Figure 1. The Architecture of a Conventional Public Key Infrastructure

2. Security Interface Limitations: In the ubiquitous environment, users' devices will access service devices to receive services frequently and dynamically. As a consequence, the number of the authentication and authorization operations will increases drastically in proportion to the number of services and sensor devices, necessitating much higher computing power in all computing devices. Even though RFID and smart card solutions are widely used for an access control in these days, their utilization is restricted by the resolution of physical proximity and limited complexity of circuitry. To overcome these limitations of the conventional authentication system, a diminutive security device that is capable of providing interactive communication with other devices, PKI-based authentication, and location based services [6, 7] is developed, whose name is PANDA[3].

The remainder of the paper is organized as follows. We present relevant work in Section 2. In Section 3 we present overall system design and components of the proposed security infrastructure and in Section 4 we describe the proposed single sign-on protocol. Evaluation of performance of pKASSO is presented in Section 5 and ubiquitous services coupled with pKASSO are illustrated Section 6 and Section 7 concludes this paper.

2. Related Work

- **SFLASH**^{v2,v3}: In a ubiquitous and mobile environment, much effort has been made to facilitate the deployment of a security infrastructure required for verifying the authenticity of communicating parties and transferring trust among devices over the Internet. Among them, PKI has been considered as a promising foundation for the requirements. The organization of a traditional PKI is illustrated in the Fig.1. Even though PKI provides full security features including AAA and non-repudiation, it has a severe drawback when it is used by a resource-constrained device. That is, a user service latency mainly determined by an authentication and authorization latency is exacerbated by not only the restricted resources of the device but also extremely high complexity of RSA operations for encryption, decryption, and certificate validation in PKI.

  Therefore, in order to adopt PKI in a ubiquitous environment, $SFLASH^{v2,v3}$[9] proposed a new cryptography for the purpose of reducing the complexity of the RSA algorithm. The $SFLASH$ is a signature scheme based on a trapdoor function introduced in $SFLASH^{v3}$ specification[10]. They reduced the authentication latency by using the verification mechanism based on SHA-1 algorithm. It has, however, two drawbacks: that the size of its public key is much larger than that of a conventional RSA and that it is troublesome to provide the interoperability because of its deviation from the PKI standard.

- **M-PKINIT [11] :** It is lighter version of the PKINIT[12] that is an extension to the Kerberos protocol for using public key authentication between user and KDC instead of using the symmetric key based authentication. M-PKINIT was proposed in an attempt to reduce a significant overhead of authentication operation when public key protocol operations are invoked in a mobile device. It is a combination of the public key based Kerberos PKINIT and Charon, which is an authentication mechanism providing secure communication between a lightweight PDA client and a Kerberos server using an intermediary system, called a proxy. It aims at enhancing the security of the Kerberos protocol by using a minimal number of public key operations along with a proxy for load distribution. However, this scheme requires public/private key operations whenever a user moves to other Kerberos realm. Besides, three interactions are required between a mobile device and the proxy server to get a TGT(Ticket Grant Ticket) and a SGT(Session Grant Ticket) for each authentication.

- **Kerberos Assisted Authentication:** A conventional authentication system based on symmetric cryptography named Kerberos is shown in Fig.2. Kerberos can provide a shorter authentication latency than PKI in virtue of the lesser overhead of symmetric key operations. Kerberos is, however, inapplicable to our environment since it does not support a digital signa-

[3] PANDA: Personal Authentication Network Device Architecture[6, 7]

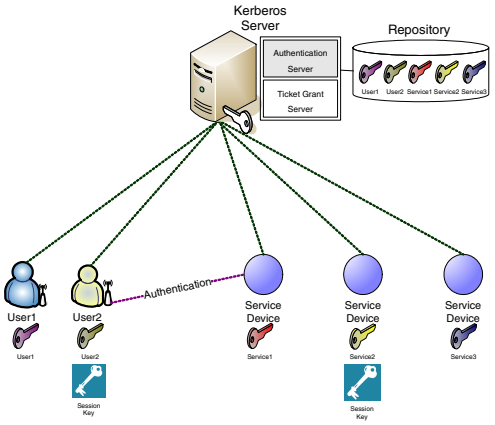

Figure 2. The Architecture of a Conventional Security Infrastructure based on Kerberos

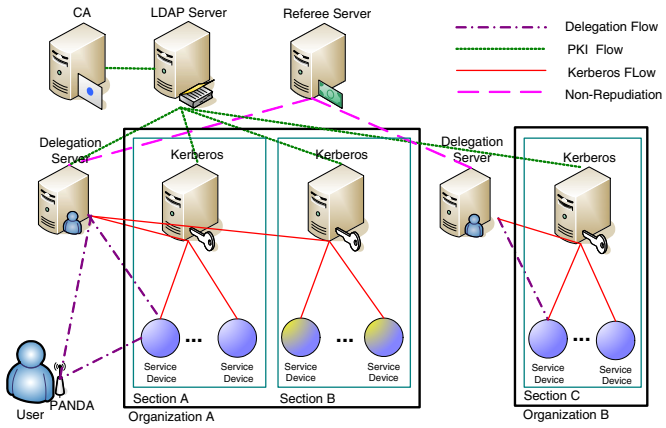

Figure 3. The Proposed Security Infrastructure, pKASSO

ture and non-repudiation mechanism that are essential functionalities in security applications. [13] studied how to deploy Kerberos into mobile ad-hoc networks. They presented a secure key exchange scheme for use in ad-hoc networks that is based on Kerberos protocol and introduced measures like replication and elections, so as to ensure maximum connectivity of the clients with servers. It cannot, however, provide an Internet-wide interoperability and AAA because Kerberos uses the authentication mechanism based on symmetric cryptography with a pre-shared secret key.

3. Design and Component of Security Infrastructure

3.1 Overall system design

Deliberating on the aforementioned system requirements, we design our security infrastructure as follows,
1) Our system adopts PKI as an underlying security infrastructure so as to provide a digital signature and non-repudiation mechanism for AAA.
2) In order to reduce authentication latency and user's intervention, we propose a single sign-on protocol that deploys a delegation mechanism using a proxy certificate [14]. For the protocol, we devise an intelligent delegation server which is responsible for performing prohibitively expensive PKI operations on behalf of a diminutive security device so as to minimize computational overhead at the security device. It also exploits user's context information from ubiquitous sensors for a context-aware authentication technology without compromising any security level of PKI.
3) In an attempt to provide a way to manage a large number of devices and sensors in a ubiquitous environment efficiently and cost-effectively, we employ Kerberos [15] in collaboration with the intelligent delegation server. Kerberos divides the physical ubiquitous environment into a set of sections to disperse the authentication traffic and to achieve scalability as well.

3.2 Internal of the proposed security infrastructure

The overall architecture of our proposed security infrastructure is illustrated in Fig.3. An organization, wherein a number of service devices and sensors are embedded, is divided into a set of sections. In each section, a Kerberos server is placed to manage all devices and sensors pertaining to the section, and their cryptographic information. The Kerberos server collaborates with the intelligent delegation servers and PKI entities such as CA, LDAP directory server, OCSP responder. In this architecture, the number of generated public/private key pairs can be reduced drastically since each device uses a symmetric key and only a public/private key pair for a Kerberos server is generated.

Seven major components of our architecture are a user, service device, CA, LDAP directory server, Kerberos server, delegation server, and referee server.

- **User** is a mobile entity receiving provided services in our ubiquitous security environment. In the environment, in order to utilize the services including authentication, authorization, and accounting, each user should carry a diminutive security device, PANDA, that is a type of smart card enhanced with ZigBee[16]-based interactive communication capability and a location sensing capability required for context-aware services. PANDA is shown in Fig.4.

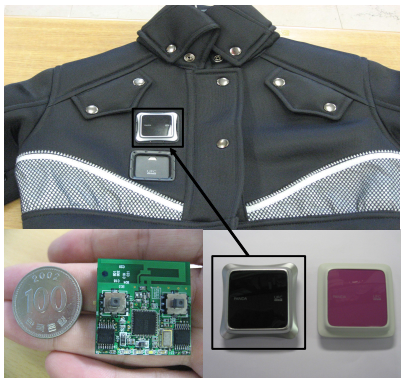

Figure 4. Top: Docked PANDA with UFC[1], Bottom-Left: PANDA Ver2.0, Bottom-Right : Packaged PANDA[6][7]

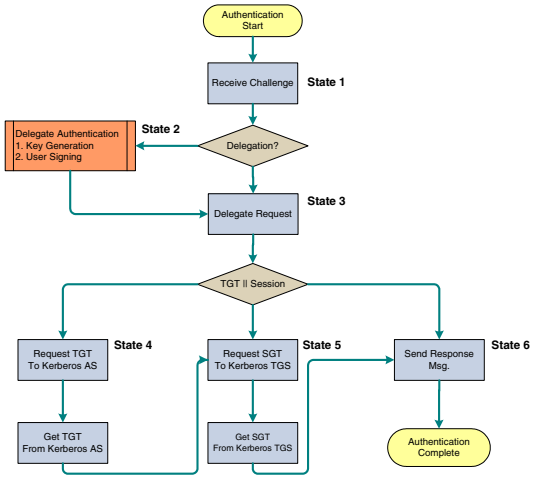

Figure 5. The Proposed SSO Protocol Flow Diagram

- **Service device** is permeated in surroundings for providing services. In our environment, it has a ZigBee communication module in order to interact with PANDA.
- **CA** is an entity which issues digital certificates which state that the CA attests that the public key contained in the certificate belongs to an entity noted in the certificate.
- **LDAP directory server** is responsible for storing and distributing certificates published by a CA.
- **Kerberos Server** is a widely-used authentication server based on a symmetric cryptography. In our environment, it is located in each section, and manages and generates TGTs and SGTs for single sign-on.
- **Delegation Server** is designed to offload complex PKI-related operations from PANDA to the infrastructure, making it possible to develop PANDA with cheap and simple hardwares. It also maintains all proxy certificates containing private keys and public keys that are delegated and signed by PANDA. When a user enters into the security infrastructure for the first time, the user will delegate his authentication operations to the delegation server by following RFC3820[14]. Afterward, the delegation server takes over all authentication operations until the users' proxy certificate is expired.
- **Referee Server** provides a non-repudiation mechanism against a malicious user's behavior. The non-repudiation mechanism will take effect on as long as PANDA uses its own private key in an authentication process. But, after PANDA delegates its operations to the delegation server, it will not use its private key any more so as not to provide the non-repudiation mechanism any more. In order to bring back the mechanism into our system even when delegation is on-going, we devised a referee server. The server investigates all authentication process, generates binding information between a device and the authentication messages on-the-fly, and retains the information in its local storage for future accusation.

4. Proposed SSO Protocol

4.1 SSO protocol flow diagram

The flow diagram of the proposed single sign-on protocol is shown in Fig.5. The protocol consists of six states.

- **State 1**: If a user wants to receive a service from a service device, the user will accept a beaconing challenge message from the service device.
- **State 2**: If it is the first time when the user accesses the security infrastructure, the user delegates his or her authentication operations to the delegation server. For this purpose, a public/private key pair is generated by the delegation server and then the public key is transmitted to the user. On the arrival of the public key, the user generates a proxy certificate containing the public key and sign it using his private key. Lastly, the user sends the proxy certificate back to the delegation server.
- **State 3**: After the delegation process is completed, the user encrypts the received challenge message using AES two times and sends the encrypted message

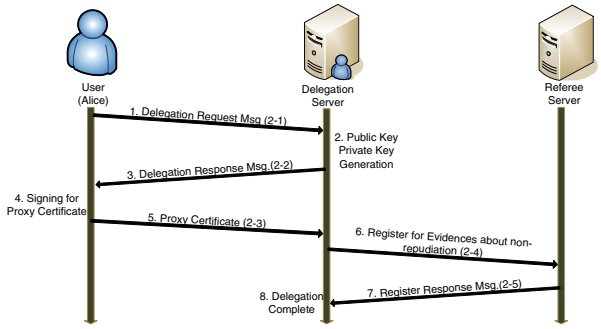

Figure 6. State2: The Proposed Delegation Mechanism

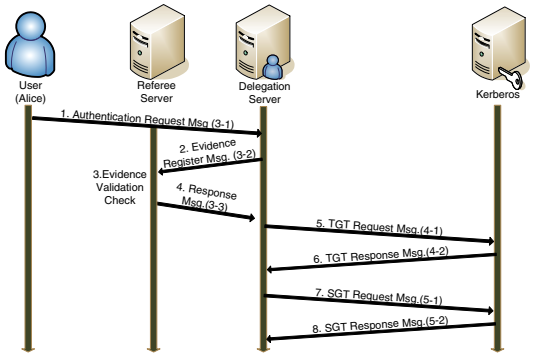

Figure 7. Message Flow Diagram from the State3 to the State5

which is an authentication request message to the delegation server.

- **State 4, 5, 6**: On the reception of the authentication request message, the delegation server contacts to the Kerberos server to get a TGT ($State4$) and an SGT ($State5$). Finally, the delegation server generates and sends a response message to the service device. After the validation check of the service device, the authentication operation is terminated ($State6$).

4.2 Description of the SSO protocol

In this section, we describe the authentication and the non-repudiation mechanism of the proposed Single Sign-On protocol. More specific description of the protocol is described in [17].

- **State 1:** In this state, a service device (Bob) keeps beaconing **Message 1-1** that is comprised of his service ID and $Bob\_Capsule$ periodically until a PANDA bearer, ($Alice$), initiates an authentication process after receiving **Message 1-1**. The $Bob\_Capsule$ is the hashed value of two inputs, a unique serial number and a randomly generated nonce.

- **State 2:** The message flow of $State2$ is shown in Fig. 6. $State2$ is the phase for conducting a user delegation by generating a proxy certificate. If $Alice$ has already delegated herself, the state transits to $State3$. Otherwise, $Alice$ needs to start delegation by entering $State2$. In order to delegate authentication operations, $Alice$ sends a delegation request message, **Message 2-1**. In response, the delegation server generates a private/public key pair and sends the public key back to $Alice$. Lastly, $Alice$ generates a proxy certificate containing the public key and then signs the certificate with her private key in **Message 2-2**. In this phase, a referee server stores a secret key signed by $Alice$ and the evidence, the proxy certificate, for non-repudiation in **Message 2-3/2-4**.

- **State 3:** The $State3$ is the phase of generating and sending an authentication request to a delegation server. On the reception of a challenge message, **Message 1-1**, from the service device, $Alice$ generates **Message 3-1** by combining the challenge message with the subkey (a symmetric key) that will be secretely shared with the service device (Bob) and then sends it to the delegation server. Subsequently, the delegation server generates **Message 3-2** as a proof that states $Alice$ sends **Message 1-1** for authentication; send it to the referee server for non-repudiation. On the arrival of the message, the referee server checks the validity of the authentication request message and sends the result to the delegation server in **Message 3-3**. If the delegation server receives the OK message, the state moves to $State4$.

- **State 4:** The delegation server sends the TGT request message to the Kerberos server in case that the delegation server does not have previously issued the TGT for Bob **(Message 4-1)**. On the other hand, if the delegation server already holds it, the protocol moves to the $State5$ promptly without doing anything in this state. The Kerberos server responds to the TGT request message and sends the TGT to the delegation server in **Message 4-2**. Because each Kerberos server and a delegation server have their own certificates, they can authenticate each other by using an existing PKI.

- **State 5:** If the delegation server obtains TGT in $State4$, the delegation server sends a SGT request message to the Kerberos server in **Message 5-1**. In response to the request message, the Kerberos server sends a new SGT for Bob to the delegation server in **Message 5-2**. If the delegation server gets SGT

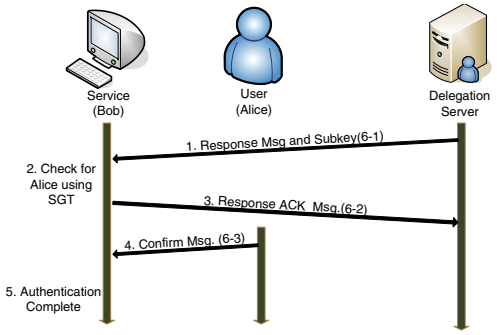

Figure 8. Message Flow Diagram of the State6

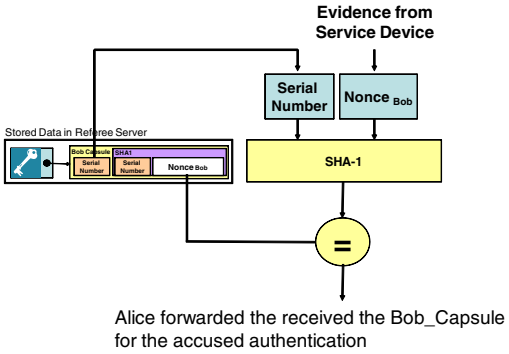

Figure 9. Non-Repudiation Mechanism in pKASSO

for Bob in advance using the prediction mechanism for the user authentication request [18], this state can be skipped and the protocol can proceed directly to the $State6$ immediately. The message flows from the $State3$ to the $State5$ are represented in Fig. 7.

- **State 6:** $State6$ is the phase where the delegation server sends the final response message to the service device, Bob, in **Message 6-1** and confirms the authentication for Bob. Then, Bob checks the validity of $Alice$ using the $Bob\_Capsule$ and shares the subkey generated by $Alice$ in **Message 3-1**. After that, Bob sends the response message to the delegation server for the mutual authentication in **Message 6-2**. Finally, the authentication is completed by the $Alice$'s confirm message, **Message 6-3**. As a result, $Alice$ and Bob can share the subkey after the authentication. The message flow of the $State6$ is represented in Fig.8.

4.3 Non-repudiation mechanism

The history data that states $Alice$ authentication requests is retained in the referee server.

- Data stored per delegation
  - ID_{Alice} : Message 2-4
  - $K_{Refe,Alice}$, $K_{Delg,Refe}$: Message 2-4
  - Seq_{Alice} : Message 2-5

- Data stored per authentication
  - $Bob\_Capsule$: Message 3-2
  - $E\{K_{Refe,Alice}, Bob\_Capsule\}$: Evidence data

How to prove that $Alice$ did send an authentication request message to the delegation server by using the history data is illustrated in Fig. 9. Let's assume that a service device asserts that $Alice$ repudiates an authentication of the service device. In this case, the service device can put in a claim for a justice with the serial number included in the challenge message ($Bob\_Capsule$) to the referee server. Then the referee server requires the evidence data that were used to generate the challenge message with the serial number to the user and inputs the evidence data and the serial number to SHA-1. If the output of SHA-1 is identical with the stored data in the referee server, it proves the fact that $Alice$ forwarded the received $Bob\_Capsule$ for the accused authentication. Therefore, the referee server can refute $Alice$'s repudiation.

5. Performance Evaluation of pKASSO

The proposed protocol deploys PKI, the delegation mechanism using the proxy certificate and Kerberos protocol based on a symmetric key cryptography. In this section, we analyze the proposed protocol considering authentication latency and safety of the protocol.

5.1 Aspect of the authentication latency

As we mentioned earlier, the proposed protocol can provide seamless authentication after the delegation. If a user accesses the security infrastructure for the first time, the user is required to delegate his authentication. Therefore, the initial authentication takes longer than delegated authentications. On the other hand, the authentication latency can be shortened drastically after the delegation because the user can be authenticated using a lot simpler symmetric cryptography. Therefore, the flow of the authentication can be changed as following cases.

- The first access to the security infrastructure
  State1 - State2 - State3 - State4 - State5 - State6

- Authentication after the delegation
  State1 - State3 - State4 - State5 - State6

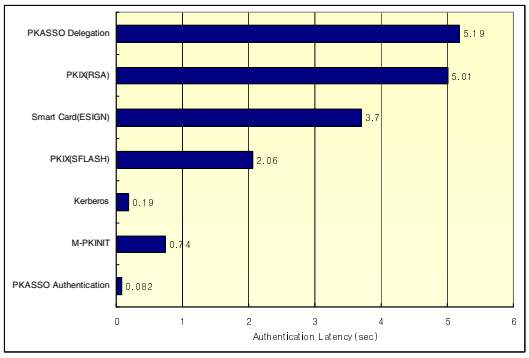

Figure 10. Authentication latency for PKIX (RSA, SFLASH), Kerberos, M-PKINIT, and pKASSO.

- If the delegation server has obtained TGT of the user State1 - State3 - State5 - State6
- If the delegation server has already obtained TGT and SGT thanks to predicted location information State1 - State3 - State6

The reduction of authentication latency with our scheme is illustrated in Fig. 10, which is compared with a general PKIX(RSA, SFLASH) operation equipped with PANDA and a smart card [23]. Let's assume that a user operates the authentication about 100 times per day. In case that 1024-bit RSA algorithm and SFLASH algorithm are processed on PANDA equipped with 8-bit processor (16Mhz)[21], the authentication latency is on average 5.01 sec, 2.06 sec respectively. In case that an authentication with Kerberos and M-PKINIT is executed on above platform, the authentication latency is on average 0.19 sec, 0.74 sec respectively. It is the reason that Kerberos is authentication protocol based on symmetric key and M-PKINIT should operate asymmetric key operations to obtain a TGT. In case of acquiring a SGT in M-PKINIT, the user can get it without asymmetric key operations so that the authentication latency of M-PKINIT can be much shorter than PKIX(RSA) and PKIX(SFLASH). The latency of a contact-type smart card is estimated as 3.70 sec[20], that is faster than PKIX(RSA) on our security device. On the other hand, the delegation operation takes about 5.19 sec, that is longer than a general PKIX(RSA) authentication. However, the authentication latency using our SSO protocol in PANDA can be reduced to 0.082 sec for the specified period after the delegation. As we described in the previous section, the reduction of the authentication latency of PANDA is due to offloading complex operations from the devices to the infrastructure. As a result, we can decrease the authentication time from 5.01 sec to 0.082 sec in average so as to meet our system requirements from the user's aspects.

5.2 Protocol Safety Analysis

We analyzes protocol safety considering replay attacks and Man-in-The-Middle Attacks (MITM). When a user accesses the security infrastructure for the first time, the user performs a mutual authentication with the delegation server and exchanges a key pair used for the specified period using PKI. In the delegation mechanism, the user can specify the delegation period and all of the messages include the nonce data against replay attacks. In this section, we show the safety of our proposed protocol from replay attacks and MITM attacks.

[Proposition 1] The proposed protocol is safe from the replay attacks.

Proof: Let's assume that the authentication path is [$Alice$⇔Delegation Server⇔Kerberos Server⇔Bob]. We prove the safety for the next attack types as follows.

1) Replay attack for the delegation request message (Message 2-1)

2) Replay attack for the delegation response message (Message 2-2)

3) Replay attack for the authentication request message (Message 3-1)

4) Replay attack for the response message between the delegation server and the service device (Message 6-2)

- In case 1), an intruder can try to attack by sending the captured delegation request message. However, the key to share with the delegation server cannot be read by the intruder because the key is encrypted using the public key of the delegation server. Therefore, the intruder cannot proceed the delegation mechanism.
- In case 2), an intruder can try to attack by sending the captured delegation response message. However, the intruder cannot succeed to attack because the delegation request message includes the nonce data enclosed in the delegation request message.
- In case 3) and 4), an intruder cannot reuse the authentication request message(Message 3-1) and the response message(Message 6-1) because the $Bob\_Capsule$ included in the challenge message is altered per authentication.

[Proposition 2] The proposed protocol is safe from MITM attacks.

Proof: We prove the safety for the next attack types as follows.

1) MITM attack between a user and a delegation server

2) MITM attack between a user and a service device

3) MITM attack between a delegation server and a service device

- In case of 1), each entity performs a mutual authentication over PKI before a delegation and shares a key for secure connection. Therefore, the intruder cannot forge the authentication request message of the user.
- In case of 2) and 3), each service device generates the $Bob\_Capsule$ that is altered per authentication, and this capsule is encrypted and transmitted using the shared key that is generated in a previous state. Therefore the intruder cannot succeed to masquerade as the user or the service device.

6. Conclusion

This paper presented our effort in designing a new PKI-based security infrastructure, pKASSO, that offers an efficient authentication technology for an ubiquitous environment, wherein a large number of devices and sensors are scattered for providing various services. Our security infrastructure features two main achievements: 1) PKI-based single sign-on protocol especially tailored for managing efficiently a large number of devices and sensors in the ubiquitous environment, 2) an intelligent delegation server with a newly devised referee server that ensures non-repudiation of any transaction between a delegator and delegatee. As a consequence, our infrastructure enables a cost-effective but uncompromisingly secure development of a diminutive security device. Furthermore, our delegation mechanism significantly improves an authentication latency as well. According to the performance evaluation, the authentication latency(Avg. 0.082sec) is much shorter than a contact type smart card(Avg. 4.31sec) and a general PKI authentication latency(Avg. 5.01sec). As a result, our security infrastructure and protocol can be applied to the ubiquitous security environment. Based on our design and performance evaluation, we developed PANDA and a security infrastructure, pKASSO, and are currently implementing services for a ubiquitous campus.

References

[1] K. H. Park and UFC Group, "UFC: A ubiquitous fashionable computer," Intl. Conference on Next Generation Computing, 2005.

[2] S. L. Jason I. Hong, Jennifer D. Ng and J. A. Landay, "Privacy risk models for designing privacy-sensitive ubiquitous computing systems," in Proceedings of the 2004 conference on Designing interactive systems: processes, practices, methods, and techniques, 2004.

[3] I.F. Akyildiz and S.Mohanty, "A Ubiquitous Mobile Communication Architecture for Next-Generation Heterogeneous Wireless Systems," IEEE Radio Communications 2005.

[4] P. Horster, M, Michels, "Hidden signature schemes based on the descrete logarithm problem and related concepts," Proc. of Communications and Multimedia Security 1995.

[5] Mike Fraser, "Mobile and Ubiquitous Computing," COMSM0106 Mobile and Ubiquitous Computing.

[6] Ki-Woong Park, S.S. Lim, H.C. Seok, and K.H. Park, "Ultra-Low-Power Security Card, PANDA, for PKI-based Authentication and Ubiquitous Services," Intl. Conference on Next Generation Computing, November 2006.

[7] K.W. Park, H.-J. Choi, and K.H. Park, "An interoperable authentication system using ZigBee-enabled tiny portable device and PKI," Intl. Conference on Next Generation Computing, 2005.

[8] Hyunchul Seok, Ki-Woong Park, S.S. Lim, and K.H. Park, "Implementation of U-Kiosk based on PANDA and VNC," in 33th KISS Conference on Computer System, October 2006.

[9] N.T.Courtois, L.Goubin1 and J.Patarin , "SFLASH^{v3}, a fast asymmetric signature scheme, Proceedings of ASIACRYPT" 1998, LNCS n1514, Springer, 1998, pp. 35-49.

[10] J. Patarin, L. Goubin, N. Courtois, "$C^{*\pm}$ and HM: Variations around two schemes of T. Matsumoto and H. Imai, in Advances in Cryptology", Proceedings of ASIACRYPT'98, LNCS n1514, Springer, 1998, pp. 35-49.

[11] Harbitter, A. and Menasce, D. A., "The performance of public key enabled Kerberos authentication in mobile computing applications", Proc. of the 8th ACM conference on Computer and Communications Security 2001.

[12] Tung, B., et al., Public Key Cryptography for Initial Authentication in Kerberos, 2001: http://www.ietf.org/internet-drafts/draft-ietf-cat-kerberos-pk-init-12.txt.

[13] Asad Amir Pirzada and Chris McDonald, "Kerberos Assisted Authentication in Mobile Ad-hoc Networks" ACM International Conference Proceeding Series; Vol. 56 , 2004

[14] S. Tuecke, V. Welch "Internet X.509 Public Key Infrastructure (PKI) Proxy Certificate Profile" RFC3820, 2004

[15] J. Linn "The Kerberos Version 5 GSS-API Mechanism" RFC1964, 1996

[16] Z. A. B. of Directors, "ZigBee Specification v1.0." ZigBee Alliance,2005.

[17] Ki-Woong Park, S.S. Lim, H.S. Song, and K.H. Park, "A New PKI-based Single Sign-On Protocol for a Diminutive Security Device, PANDA, in a Ubiquitous Security Environment", 8th International Symposium on Systems & Information Security, November 2006.

[18] Woo-Min Hwang, S. Lim, K. Park "A Context-Aware Replacement Page Cache for Wearable Computer" Next Generation PC, 2006

[19] M.Mimura, S.Ishida, Y.Seto , "Fingerprint verification system on smart card," International Conference on Cpmsumer Electronics, 2002.

[20] C.Yang, H.Morita, and T.Okamoto , "Security and Performance Evaluation of ESIGN and RSA on IC Cards by Using Byte-Unit Modular Algorithms," IEICE Transactions., Vol.E88-B, No.3 March 2005.

[21] ATMEL, "ATmega128(L) Datasheet: 8-bit Microcontroller with In-System Programmable Flash." ATMEL., 2005.

[22] Infinion, "infinion SLE66 DataSheet. Infinion.", 2006.

[23] Hitachi Single-Chip Microcomputer H8/2168 Group Hardware Manual 3.0 ,Renesas Technology CO. Ltd., 2004.

[24] CORE Street Lab, "Smart Card Reader System Functional Specification Document", CORE Street press, 2006.