IEICE TRANS. INF. & SYST., VOL.E92–D, NO.5 MAY 2009

PAPER Special Section on Information and Communication System Security

Development of Single Sign-On System with Hardware Token and Key Management Server

Daiki NOBAYASHI[†a)], ***Student Member***, **Yutaka NAKAMURA**[†], **Takeshi IKENAGA**[†], ***and*** **Yoshiaki HORI**[††], ***Members***

SUMMARY With the growth of the Internet, various types of services are rapidly expanding; such services include the World Wide Web (WWW), the File Transfer Protocol (FTP), and remote login. Consequently, managing authentication information, e.g., user ID/password pairs, keys, and certificates– is difficult for users, since the amount of required authentication information has been increased. To address this problem, researchers have developed a Single Sign-On (SSO) system that makes all the services available for a user via a one-time authentication: however, existing authentication systems cannot provide such SSO services for all kind of services on the Internet, even if the service provider deploys the SSO server. Further, existing systems also cannot provide the SSO service which does not make it conscious of a network domain to a user on secure network environment. Therefore, in this paper, we propose a new SSO system with a hardware token and a key management server to improve the safety, ubiquity, and adaptability of services. Further, we implement the proposed system and show its effectiveness through evaluation. Adding any functions for this system provides various conveniences to us. We also explore the ability to add functions to this system; for example, we add high trust connection functionality for a Web server and show its effectiveness.

key words: *single sign-on, hardware token, key management server, authentication, security*

1. Introduction

A Single Sign-On (SSO) system, which provides one-time authentication for one or more services for a user, has been developed in recent years [1]–[6]. Indispensable network-based service in enterprises and academic organizations are expanding and becoming increasingly diverse. Most services identify users via an authentication scheme. Currently, authentication schemes typically require a variety of authentication information, including UserID/Password pairs, Private/Public keys, a certificate, and so on. It is difficult for users to manage their authentication information because users maintain such information independently for each service. Each service provider requires an authentication scheme to reduce management costs for authentication, and users want simpler authentication schemes for their convenience.

There are two fundamental SSO architectures: one is provided as part of the infrastructure constructed by service providers; the other is provided by a user-side tool each user can utilize to automatically manage various authentication credentials. To adopt the former architecture, service providers have to build a common SSO environment to share the authentication scheme, requiring additional investment among service providers and coordination between such providers to use the common SSO environment. Such an approach provides benefits for its users because it requires no additional cost for the users. The latter architecture requires no additional investment from service providers. Instead, it requires an additional cost on the user side; however, it provides several benefits for the users, summarized below:

- Applicability to a variety of services provided by different service providers
- Flexibility to interface with a new service
- User mobility

In this paper, we propose a new SSO system that does not interrupt the functionality of various legacy services on the Internet, such as remote login, File Transfer Protocol (FTP), and World Wide Web (WWW) Services. Our proposed system consists of a Key Management Server (KMS), a Hardware Token (HT) and a Client Agent (CLA). The KMS maintains user authentication information, such as UserID/Password pairs, Private/Public keys, and a certificates. The Hardware Token maintains two keys, (1) a symmetric key to encrypt/decrypt user credential, and (2) a private key to access the KMS. The CLA mediates the authentication process between a service provider's authentication scheme and a user. The CLA takes in authentication credentials from the KMS, when the CLA needs to provide access to services. Users on a network can obtain the authentication information in the KMS by using the CLA. Further, when a user access a Web server, the CLA is able to validate the Web server's legitimacy from such credentials as a server certificate. With this functionality, our SSO system is effective against phishing attacks.

This paper is organized as follows. In Sect. 2, we discuss existing SSO systems and their inherent problems. In Sect. 3, we show our proposed SSO system. Section 4 describes the prototype construction of our system. We show an additional functionality of the CLA in Sect. 5. We discuss our proposed system in Sect. 6, and, finally, we summarize our work in Sect. 7.

Manuscript received August 4, 2008.
Manuscript revised December 29, 2008.

[†]The authors are with the Kyushu Institute of Technology, Kitakyushu-shi, 805–8550 Japan.
[††]The author is with the Kyushu University, Fukuoka-shi, 819–0395 Japan.
a) E-mail: nova@net.ecs.kyutech.ac.jp

DOI: 10.1587/transinf.E92.D.826

2. Single Sign-On

Andreas et al. classify SSO systems in [4], categorizing the structure and how to operate SSO systems. In this paper, we focus on who provides the SSO service. A SSO system is classified into the service of the server side and the application at the user side.

2.1 Application of a SSO System for Users

User-side SSO systems can be used by installing software on a client computer. Such a user-side SSO system has the advantage of adapting to various services, because a user can configure this system at any time. This system has the disadvantage of increased costs in constructing the SSO environment, resulting in increased management costs of the user. For example, commercial system products include PassOne [7] by NEC Software Tohoku and e-z-Login [8] by NTT Data, both password managers. These examples both automatically input authentication information instead of the user; however, these products differ in how they manage authentication information. In PassOne, a user sets up a database server that stores the authentication information in a network domain provided to the SSO. The e-z-Login product uses the management server much like the PassOne. Both products provide SSO within the same network domain as that of the database server. Software product, such as ID Memory [9] by goo and ID Manager [10] by WoodenSoldier Software, are password manager in which users must manage their UserIDs and Passwords personally. Therefore, installing such software on an external hard drive, password managers provide a SSO service for a user everywhere; however, if a user loses this external hard drive, others may misuse the authentication information.

2.2 SSO by a Service Provider

In this subsection, we describe SSO systems made available by service providers providing FTP and WWW access. Such service construct a SSO environment that supports multiple user applications. This approach has the advantage of managing all users' information; however, users can only use services that the service provider offers.

Some systems provide unifying authentication against several Internet services; such systems include Network Information Service (NIS), Lightweight Directory Access Protocol (LDAP) and Kerberos. In these cases, users only manages one user UserID/password pair; however, if this pair is leaked the user is negatively affected at all services. Furthermore, users also cannot use such SSO environments on different service providers.

Another approach to server-side SSO is to set the authentication server as an intermediary between users and the services' server. Users can use all services via this mechanism. This approach is classified into reverse-proxy type and agent type, as described in the following subsections.

2.2.1 Reverse-Proxy Type

Applying the reverse-proxy approach to server-side SSO, a reverse-proxy server exists between a server and a user terminal. When a user authenticates to the server, the reverse-proxy server obtains certificate information from an authentication server and authenticates services instead of the user [4]. An example of such a product is the HP IceWall SSO [11] by Hwelett-Packard. As shown in Fig. 1, all communication to the services' server goes through the reverse-proxy server. When the server requests login information, the reverse-proxy server queries the authentication server to obtain login information. The reverse-proxy server obtains authentication information from an authentication server and authenticates the server instead of the user. For Web services, the authentication server answers such a query by publishing an "access ticket" used as a cookie on the user's Web browser.

An advantage of using a reverse-proxy type approach is that no additional SSO software is required on the service'server. Therefore, this approach is platform-independent; however, because all traffic goes through a proxy server, this system has scalability problems.

From the user's perspective, this approach reduce the need to input a password or other authentication information; however, if authentication information is stolen, all applications may be access ed illegally. As a countermeasure to such an authentication information leak, an authentication server using Public Key Infrastructure (PKI) and a one-time password may be used. Further, to improve security, we can apply the method using a hardware token.

2.2.2 Agent Type

A software called "Agent" is installed on a service's servers, and this software acts as the authentication server's deputy. Entrust Get Access [12] by Entrust Japan is an example product that uses this approach. Figure 2 shows an example of SSO of agent type SSO. For Web services, a user logs in to the authentication server. The client's terminal

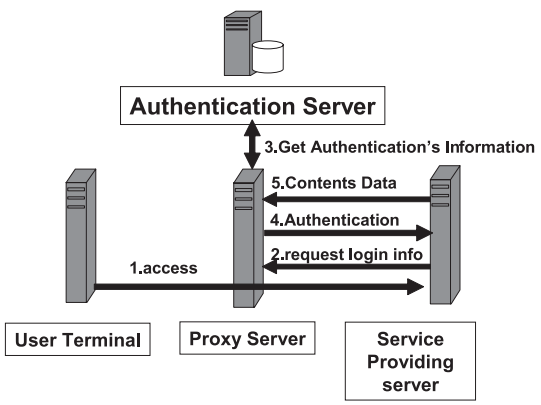


Fig. 1 Example of reverse-proxy type SSO.

obtains an "Access Ticket" from the authentication server, which the Web browser uses to authenticate when the user accesses the services' servers. The agent in the services' server obtains the user's credentials from the authentication server and confirms the access ticket. In such agent SSO systems, the load on the server from user activity does not increase, thus this approach is superior in term of scalability; however, this approach requires agent software to be installed on each server. The service provider has to build and support the SSO environment.

2.2.3 Hybrid Type

To help overcome the respective problems of the reverse-proxy and agent types described above, a combined approach may be used. For example, when the services' server cannot install the agent application, the reverse-proxy type operates; otherwise, the agent type used.

2.2.4 Summary

In the above subsections, we described existing SSO methods. Password managers are superior to other systems in term of mobility, because password manager applications are carried around in the outside storage. SSO implemented by a service provider doesn't have the mobility of a user. When several providers cooperate about user authentication, user mobility improves. The password managers are scalable; however, other SSO systems can only support one application. For personal authentication, we can improve user credential security by using Public Key Infrastructure (PKI). The password managers cannot use the PKI. However, reverse-proxy type and agent type can use the PKI.

Judging from the above summary, we conclude that future SSO systems require ubiquity, security, and adaptability for many services. Ubiquitous SSO implies location-independent authentication. Secure SSO guarantees secure communication, preventing information leaks to external entities. Adaptable SSO integrates services without the user taking action to integrate such services.

3. Proposed SSO System

We propose a new SSO system that is able to manage large amounts of authentication information, providing users the ability to manage unified authentication information. Many services' servers can use the proposed SSO environment without installing SSO software on services' servers. Instead, we build a Key Management Server on the Internet to provide the necessary SSO environment. Our system uses a Hardware Token to authenticate personal credentials of each user.

3.1 Components

As shown in Fig. 3, our SSO consists of the following three components:

- Key Management Server (KMS)
- Hardware Token (HT)
- Client Agent (CLA)

3.1.1 Key Management Server (KMS)

The Key Management Server (KMS) maintains an Authentication Information Database (AIDB). Available on the Internet, the KMS also has an original Certificate Authority (CA) that generates a client certificate and a server certificate. Communication between the KMS and the Client Agent (CLA) is encrypted via Secure Socket Layer (SSL). When a client machine communicates to the KMS, both client host and the KMS verify the client certificate and the server certificate of each other. The AIDB maintains the UserID/password pair and service identifier per user. To encrypt or decrypt the authentication information, the CLA uses a symmetric key on the client machine. This authentication data is stored in the KMS. To identify the user, the KMS stores the user certificate generated by user's Hardware Token. The CA signs the user certificate to verify the user's identification.

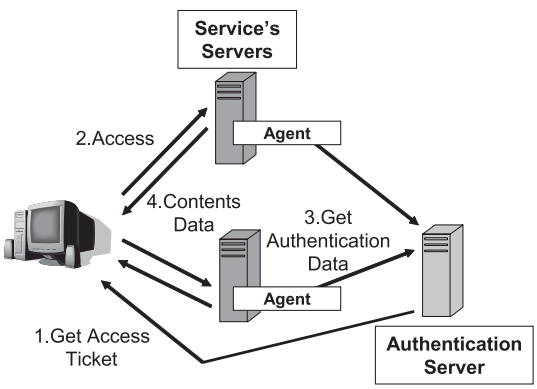


Fig. 2 Example of agent type SSO.

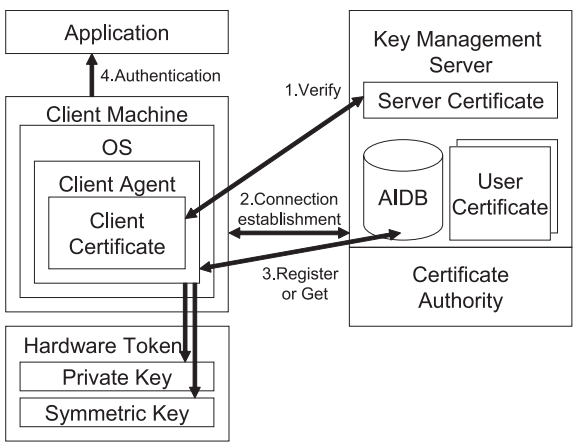


Fig. 3 Components of proposed system.

Table 1 Compare the method of SSO.

	SSO with a user		SSO with service provider		Our proposal
	PassOne etc.	password manager	reverse proxy	agent	HT & KMS
Safety	Good	Not good	Good	Good	Good
Ubiquity	Not good	Good	Depend on the situation		Good
Adaptability	Depend on the situation	Good	Not good	Not good	Good

3.1.2 Hardware Token (HT)

The user carries the Hardware Token (HT) usually anywhere. The HT maintains two keys to authenticate a user and encrypt or decrypt information in the AIDB. The first is a symmetric key to encrypt or decrypt user credentials, the second is a private key to access the KMS when the CLA needs to access provided services. Users must enter a personal PIN code to make the HT available. If a user inputs an incorrect PIN code a configurable number of times, the HT is locked to prohibit further access. Therefore, if a user loses the HT, the others cannot use it to authenticates. To further confirm ownership of the HT for a user, the KMS check the user certificate via the KMS. If a user loses his or her hardware token, the CA in the KMS will expire the user certificate. Therefore, authentication information in the KMS is strongly protected.

3.1.3 Client Agent (CLA)

The CLA mediates the authentication process between a service provider's authentication scheme and a user with an HT. Independent from other client applications, such as a Web browser or Mail User Agent (MUA), the CLA takes in a credential information from the KMS, when the CLA needs to access provided services. The CLA enters authentication information into the services' server instead of a user.

3.2 Characteristics

Table 1 compares our proposed method to the aforementioned existing SSO methods in term of safety, ubiquitously, and adaptability for services.

Our system is superior to other system in term of ubiquity due to the HT, because users can easily carry their respective HTs. Users with the HT can use the SSO environment wherever they can connect to the Internet.

Our system is superior to other system in term of safety. The HT consists of the private key to authenticate the user, as well as a symmetric key to encrypt or decrypt information in the AIDB. Users must use their HTs to authenticate to the AIDB of the KMS.

Another feature in our system is the CLA, which provides adaptability for a variety of services. The CLA provides the UserID/Password pair on behalf of the user.

A disadvantage of our system is the reliance on the HT. If a user loses his or her HT, the CA in the KMS expires the user certificate, thereby avoiding any misuse or security breaches. Further, our system avoids misuse via the AIDB; if the KMS is cracked, the AIDB is unreadable because it is encrypted using the symmetric key in the HT.

Our proposed system belongs to the Pseudo-SSO category introduced in [4]. A Pseudo-SSO system manages UserID/Password pairs for services. Such Pseudo-SSO system are further categorized as local Pseudo-SSO and proxy-based Pseudo-SSO systems. Our SSO system belongs to neither the local Pseudo-SSO nor the proxy-based Pseudo-SSO, because our SSO system stores authentication information to the AIDB on the network. If authentication information exists on a local computer, a malicious user may be able to gain access to its security information. Therefore, our proposed system is a new category of Pseudo-SSO.

3.3 Process Overview

In the following subsections, we show several processes of our system, including system preparation, registration, the login process, and others.

3.3.1 Preparation

Before using our system, a user must perform the following operations. A client host and a server host must obtain host certificates to verify themselves. The CA working with the KMS publishes these certificates, and the client and server certificates are kept on each computer. These certificates are used to verify the client host and the server host to each other.

A user generates a private key and a user certificate from the HT. The private key is kept in the HT, whereas the user certificate is stored in the KMS and is signed by the CA. These keys are utilized to perform personal authentication.

A user must make a symmetric key to encrypt or decrypt authentication information in the AIDB of the KMS. Because the plain text does not have security, we encrypt or decrypt information in the AIDB by using this symmetric key. Our system maintains this key in the HT.

3.3.2 Login to Client Agent

Figure 4 shows the login process. A user inserts a HT into a slot on the client host machine, then inputs a UserID and PIN to access the HT. The CLA sends the UserID to the HT to encrypt it via the private key. We show *UserID* that is encrypted by the private key K_{priv} as $Enc_{K_{priv}}(UserID)$. The $Enc_{K_{priv}}(UserID)$ and *UserID* are send to the KMS which decrypts $Enc_{K_{priv}}(UserID)$ by using the public key in the

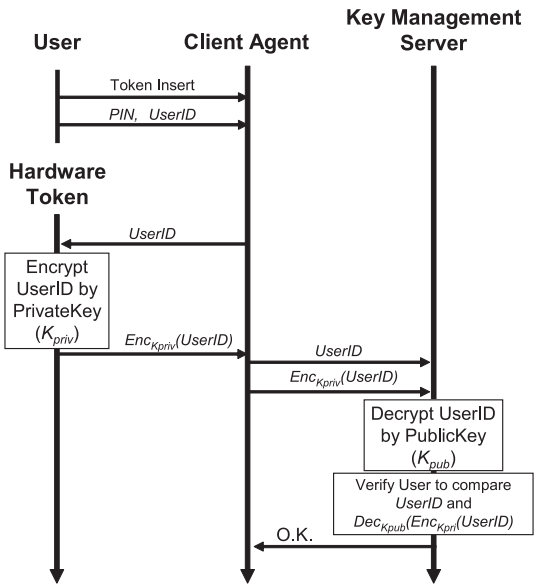


Fig. 4 Flow diagram of login processing.

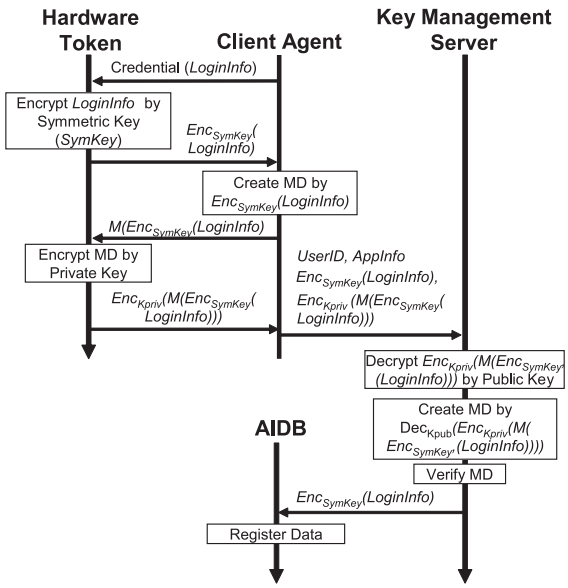


Fig. 5 Flow diagram of registration processing.

user's certificate. The *UserID* that is decrypted by the public key K_{pub} is shown as $Dec_{K_{pub}}(Enc_{K_{priv}}(UserID)$ The login process finishes and the user is authenticated by comparing $Dec_{K_{pub}}(Enc_{K_{priv}}(UserID)$ and *UserID*.

3.3.3 Communication between the CLA and the KMS

Our system uses Secure Socket Layer (SSL) to validate and encrypt communication between the CLA and the KMS. Each client and the server keep host certificates published by the CA in the KMS. Each host identifies one another's reliability by using these certificates. The communication line is verified and encrypted by the SSL and the use of these certificates. Whenever a user connects to the KMS, communication between the KMS and the CLA occurs via a new encrypted session.

3.3.4 Process of Registering Login Information in the KMS

Our system use authentication information including user name, application name, application options, login ID, password, credential key, and so on. Figure 5 summarizes the registration process in which such login information is stored in the AIDB.

First, given the login ID and password (*LoginInfo*), the HT create Enc_{SymKey}(*LoginInfo*) by encrypting *LoginInfo* using symmetric key *SymKey* in the HT. Second, the CLA creates $M(Enc_{SymKey}$(*LoginInfo*), the message digest (MD) by applying the symmetric key to the encrypted text. Third, this MD is encrypted by the private key in the HT, shown as $Enc_{K_{priv}}(M(Enc_{SymKey}$(*LoginInfo*))). Fourth, the CLA sends the UserID, Enc_{SymKey}(*LoginInfo*), $Enc_{K_{priv}}(M(Enc_{SymKey}$(*LoginInfo*)) and application information (*AppInfo*) to the KMS. Fifth, the KMS decrypts

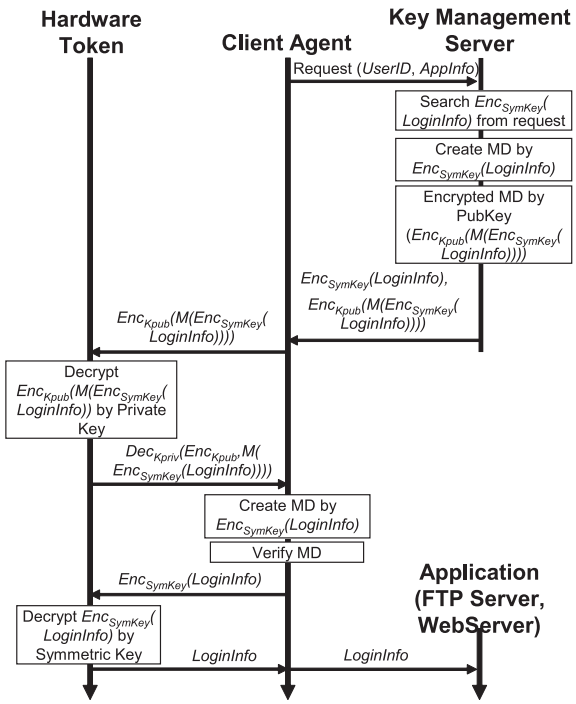


Fig. 6 Flow diagram showing process of obtaining login information from the KMS.

the encrypted MD to obtain the MD of the user certificate ($Dec_{K_{pub}}(Enc_{K_{priv}}(M(Enc_{SymKey}$(*LoginInfo*))))). Sixth, the KMS generates $M(Dec_{K_{pub}}$ ($Enc_{K_{priv}}$ ($M(Enc_{SymKey}$ (*LoginInfo*)). Given these last two MDs, if the MD generated by the CLA (step 5) and the KMS (step 6) are the same, Enc_{SymKey}(*LoginInfo*) are transmitted to the AIDB.

3.3.5 Process of Obtaining Login Information from KMS

Figure 6 shows the process of retrieving the authentica-

tion information from KMS, described below. First, when a user connects to the services' server, the CLA sends the request (*UserID* and *AppInfo*) to the KMS. Second, the KMS searches the AIDB for encrypted login information $Enc_{SymKey}(LoginInfo)$. Third, the KMS generates $M(Enc_{SymKey}(LoginInfo))$, the MD of the searching login information. Fourth, the $M(Enc_{SymKey}(LoginInfo))$ is encrypted using the user certificate stored in the KMS $(Enc_{K_{pub}}(M(Enc_{SymKey}(LoginInfo))))$. Fifth, the KMS sends $Enc_{SymKey}(LoginInfo)$ and $Enc_{K_{pub}}(M(Enc_{SymKey}(LoginInfo)))$ to CLA. Sixth, the CLA obtains the $Dec_{K_{priv}}(Enc_{K_{pub}}(M(Enc_{SymKey}(LoginInfo))))$ to use the private key in the HT. Seventh, the CLA generates the $M(Enc_{SymKey}(LoginInfo))$. Eighth, the MD generated by the KMS (step 6) is compared to the MD generated by the CLA (step 7) to ensure a match. Ninth, the CLA decrypts the $Enc_{SymKey}(LoginInfo)$ using the symmetric key in the HT, thus obtain the authentication information which an application can then use via the CLA.

3.3.6 Logout

When a user detaches the HT in the client's computer and logs out of the client machine, the CLA immediately closes the encrypted connection to the KMS. Therefore, other cannot illegally gain access to the CLA.

4. Implementation and Experimentation

In this section, we describe a prototype implementation of our proposed SSO system, and show the results of our experimentation.

4.1 Prototype Implementation

We developed a prototype system consisting of three components (as shown in Fig. 3); a KMS, a CLA and a HT. Server and client applications were implemented in C++. The KMS runs on a Linux operating system. The CLA application is available on both Windows XP and Linux operating systems. We use USB token ePass1000 by the FEITIAN Corporation [13] as a HT because it has an access control functionality using a PIN code and supports the PKCS#11 API published by RSA Corporation [14].

4.2 Experimentation

Figures 7 and 8 show screenshots of our implemented SSO system, executing in debug mode such that processing may be monitored. In particular, Figs. 7 and 8 show a series of operations, including when personal information is registered in AIDB, and when such personal information is obtained from the AIDB. During execution in debug mode, the encrypted binary data is displayed in base 64 form so that people can check. From Figs. 7 and 8, we verified that the CLA and the KMS which were implemented operate as showed in Figs. 5 and 6. The CLA registers the encrypted credential information to AIDB in the KMS in process of

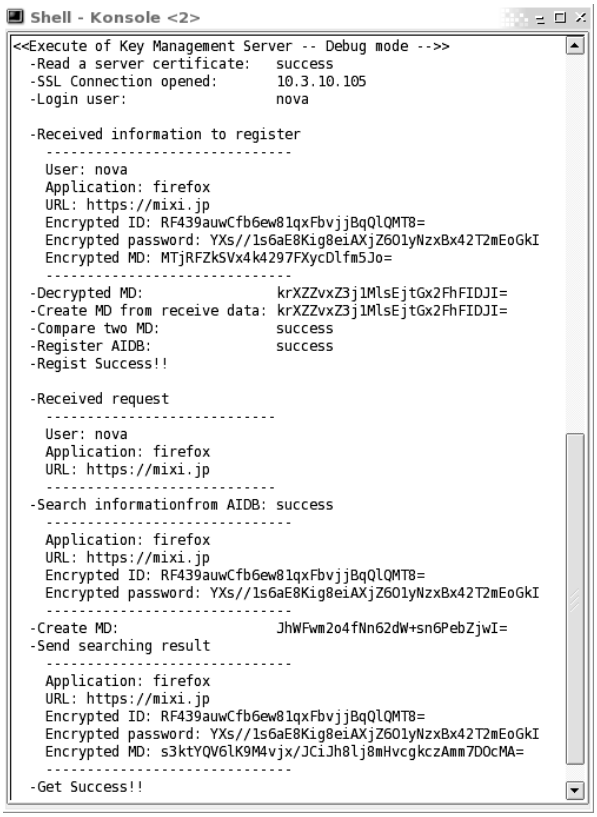


Fig. 7 Screenshot of the KMS.

"regist". Moreover, when the CLA needs credential information, CLA obtains the information from the KMS according to process of "get" and decrypted the information We verify that the KMS, and the CLA, and a HT can manage a user's credential information.

Further, we tried to log in to a Social Networking Site (SNS) Web site called "mixi" that is the most popular SNS site in Japan. In addition, we verified behavior logging in to the MSN Messenger and the POP3 server using the Thunderbird as our MUA. Integrated into our SSO system, we registered authentication information such as UserID, Password and service identifier with the KMS. We then tested using this authentication information through the CLA, verifying the successful operation of our SSO environment.

To investigate the overhead of storing and retrieving authentication information using our SSO system, we measured the time from the occurring service request and the finish of retrieving process. This time is approximately 1.2 seconds (excluding the database search time), which adds minimal time to the user's wait time. Therefore, the user can use our proposed method without feeling as stressed.

5. Application -Safety Web Basic Authentication

We expand our proposed Single Sign-On system by providing anti-phishing/anti-pharming features. Phishing is a problem in which a user is tricked into entering his or her secret information (e.g. a password or other such sensitive

Fig. 8 Screenshot of the CLA.

information) into an illegitimate Web server [15], [16]. To prevent such threats, we propose a scheme the automatically logs in, on the user's behalf, to validated a Web server.

5.1 Additional Mechanism

We expand the functionality of our SSO system to validate a Web server. Our system verifies a server's certificate with a Web server when the user connects to the Internet. When this server's certificate is valid, the user can log in to the Web site without inputting his or her UserID and password. We avoid leaking authentication information to illegitimate Web sites because the user does not input his or her authentication information. We add the new feature as follows:

- The CLA keeps a list of the root CA
- The CLA verifies a certificate of the Web server

5.2 Anti-Phishing/Anti-Pharming Process

The CLA of Our proposed system validates the legitimacy of a Web server by using the aforementioned functionality. We describe the flow of the anti-phishing/anti-pharming process in Fig. 9 and in the following subsections.

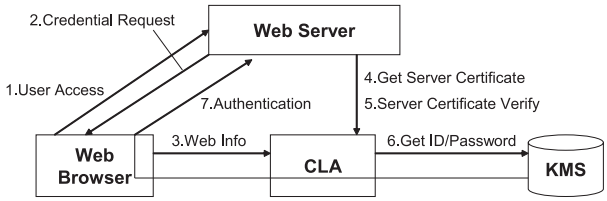


Fig. 9 Processing of Web basic authentication on our proposal.

5.2.1 Preparation

A user registers the URL and an authentication information for the Web service with our system. This registered data is stored in the KMS.

5.2.2 Case of a Legitimate Web Site

When a legitimate Web site is to be accessed, the following steps are performed. First, a user accesses the Web site on the Internet. Second, the Web server requests the login information to authenticate the user. Third, the Web browser forwards the URL to the CLA when the user opens the Web page containing the login request. Fourth, the CLA queries the Web server for a server certificate. Fifth, the CLA checks the server certificate from the Web server and the list of the root CA. Sixth, the CLA requests the UserID and password from the KMS (the CLA's request generated from Web server certificate). Seventh, the KMS obtains the user credentials in the AIDB and sends the authentication information from the KMS to the Web server through the Web browser.

The user can log in to the Web server without inputting his or her UserID and password. Further, our system avoids Web phishing because our system verifies the legitimacy of the Web server by the list of the root CA.

5.2.3 Case of Fake URL

In the case of accessing a fake URL, the CLA sends a request for authentication information to the KMS. The KMS searches for the user credentials in the AIDB; however, the KMS cannot find the user credential for the fake URL. The KMS sends a warning message to the CLA, which forwards this warning message to the Web browser. Our system detects the fake URL because our system validates each given URL.

5.2.4 Case of Fake Server Certificate

In the case of accessing a Web server with a fake server certificate, the CLA sends a request for the server certificate to the fake Web server. The CLA checks the server certificate based on whether or not it is published by a reliable root CA. The CLA judges the server as illegitimate if the server certificate is not legitimate. In such cases, the CLA sends a warning message to the Web browser without accessing the

KMS. Our system detects the fake certificate because our system validates both the server certificate and the list of the root CA.

6. Discussion

In this section, the security and reliability of our proposed SSO method is described.

6.1 Security Analysis

The KMS has manages a user's personal information. Therefore, the KMS must handle illegitimate access, eavesdropping, and other such security problems, strictly from the third party. In the following discussion, we analyze typical attacks for personal information acquisition, such as an eavesdropping, an alteration, a replay attack, and a session hijack.

An eavesdropping attack is an act in which a malicious party peruses the packet flow of a communication path and uses such information illegally. As a countermeasure, communication between a CLA and the KMS is protected by PKI and encryption using the secret key in a HT. Through these methods, the attacker cannot peruse information that flows across the communication path unless the key in the HT is illegally acquired.

An alteration attack alters information that flows across a communication path, sending strange or malicious information to a destination node. As a countermeasure, our proposed method uses a message digest; detection is immediately discovered when a transmitting error is found in the communication stream. For an attacker to successfully achieve an alteration attack, key information in the HT is required.

Finally, we describe a replay attack and a session hijack. A replay attack is an attack in which a third party illegally accesses a server by copying a user's log-in information. A session hijack is an attack in which an attacker illegally accesses a server by stealing a session ID used at the time of the connection was initially established between client and server. As a countermeasure to both types of attack, our proposed method uses a client certificate of SSL when the connection between client and server is established; our proposed method also uses the challenge response method. The KMS specifies a user by verifying the client certificate of the CLA. Therefore, the attacker cannot carry out a session hijack. Moreover, since information flowing across the connection uses the challenge response method, such information is encrypted with a different key for every login and is difficult to access illegally using copied information.

6.2 Reliability

In this section, we discuss about the management and deployment of our proposed system. A KMS manages authentication information for users. Therefore, a user cannot be provided with SSO if a server fails to operate due to scheduled maintenance a failure, a denial-of-service (DoS) attack, and so on. Deployment of the KMS may take one of two approachs: a Centralized Management approach or a Decentralized Management approach. Authentication information may be centralized or decentralized. The centralized management approach has the advantage that all users are managed by one server, but has the disadvantage of being weak against an outage and high load caused by the increase in the number of users. The decentralized management approach has the advantage of having high fault tolerance; however, since authentication information is distributed, management of such information is difficult.

6.2.1 Server Mirroring

The KMS may be set up by an Internet Server Provider (ISP) in such a way that the management of user information is provided by an authenticator function. Users with HTs can access the KMS from all networks and can use a single sign-on. When KMS falls into a situation with limited or no service after an accident, disk crash, DoS attack, etc., it will become impossible for all users to use a single sign-on. Moreover, the KMS has a load corresponding to the user increment. Providing stable services in these situations is difficult for KMS. When the KMS manages authentication information of all users, some of the aforementioned problems can be avoided by preparing the mirror server for the KMS, all user information that all KMSs have must be synchronized. Moreover, assigning the server that a user accesses according to load is necessary. Since these mechanisms are already used in various Web system, realization of mirroring and load-balancing technologies for our proposed system is straightforward.

6.2.2 P2P Technology for Distributed KMS

As another approach to implementing and operating our SSO system, each network administrator installs a KMS. Figure 10 shows an overview of the distributed method to set up multiple KMSs. A user registers authentication information in the network in which he or she belongs. The user can usually use a service of single sign-on when he is

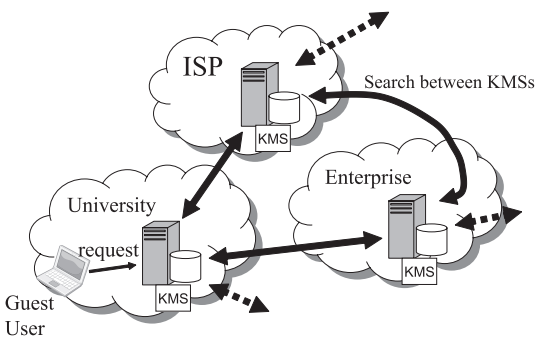


Fig. 10 The overview of distributed management method to set up KMSs.

in the network; when a user uses the KMS from another network, the local KMS obtains authentication information from the KMS in the network in which a user belongs. When the local KMS does not have the information a user needs, the local KMS searches other KMSs across the Internet. The searching methods in P2P, such as a Distributed Hash Table, can offer a high-speed searching strategy to the KMS [17]. Moreover, a KMS can search efficiently by setting up KMSs in consideration of network hierarchy, such as Domain Name Service. Therefore, the user can use the services from KMS in other networks. By this method, since secret information must not be revealed the safe communication between KMSs is required. When KMSs are set up in a distributed manner, secret information must be transmitted as safely and securely as possible.

As mentioned above, to use this system effectively in a networked environment, various decentralized administration methods are required. Since these methods are beyond the scope of this paper, we will not discuss this topic further.

7. Conclusion

In this paper, we discussed existing SSO systems and their limitations. We proposed a new SSO system without modifying the functionality of various legacy services available on the Internet. Our proposed system consists of a KMS, HTs, CLAs. User on a network can store and obtain authentication information in the KMS by using the CLA and a HT. We showed that our proposed system could be applied to various services by implementing a prototype of the proposed system, and showed its effectiveness through evaluation.

Moreover, added a function specifically for Web systems. The CLA validates a Web server's legitimacy from a credential such as a server certificate when users access such a Web server. We showed that our SSO system with this additional functionality was effective against phishing attacks.

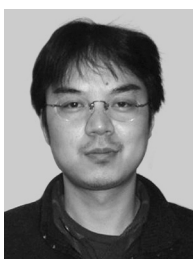
Acknowledgment

This research was partially supported by the Ministry of Education, Science, Sports and Culture, Grant-in-Aid for Scientific Research on Priority Areas, New IT Infrastructure for the Information-explosion Era, 19024062, 2007.

References

[1] A. Volchkov, "Revisiting single sign-on — A pragmatic approcah in a new context," IT Pro, pp.39–45, Jan./Feb. 2001.
[2] S. Chu, D. Good, M. Mamajek, and D. Washington, "Web-based single sign-on solutions: An SSO product matrix," Comput. Secur. J., vol.16, no.1, pp.39–49, 2000.
[3] J.D. Clercq, "Single sign-on architectures," Proc. InfraSec 2002, LNCS 2437, pp.40–58, 2002.
[4] A. Pashalidis and C.J. Mitchell, "A taxonomy of single sign-on system," Proc. ACISP 2003, LNCS 2727, pp.249–264, 2003.
[5] S.S. Sandhu, "Single sign on concepts & protocols," SANS Institute 2004, 2004.
[6] J. Miyoshi and H. Ishii, "Network-based single sign-on architecture for IP-VPN," Proc. PACRIM 2003, pp.458–461, 2003.
[7] PassOne, NEC Software Tohoku, http://www.tnes.co.jp/product/passone.html
[8] e-z-Login, NTT Data, http://www.nttdata.co.jp/release/2004/012000.html
[9] ID Memory, goo (NTT Resonant Inc.), http://idmemory.goo.ne.jp/
[10] ID Manager, WoodenSoldier System, http://www.woodensoldier.info/soft/idm.htm
[11] HP IceWall SSO, Hewlett-Packard Development Company, http://h50146.www5.hp.com/products/software/security/icewall/index.html
[12] Entrust Get Access, Entrust, http://www.entrust.com/internet-access-control/index.htm
[13] ePass1000, FAITIAN Technologies Co.Ltd, http://www.FTsafe.com
[14] PKCS#11, RSA Security, http://www.rsasecurity.com
[15] R. Dhamija, J.D. Tygar, and M. Hearst, "Why phishing works," Proc. SIGCHI Conference on Human Factors in Computing Systems, Montre'al, Que'bec, Canada, April 2006.
[16] R. Dhamija and J.D. Tygar, "The battle against phishing: Dynamic security skins," Proc. SOUPS 2005, pp.77–88, July 2005.
[17] S.A. Theotokis and D. Spinellis, "A survey of peer-to-peer content distribution technologies," ACM Comput. Surv., vol.36, no.4, pp.335–371, 2004.

Daiki Nobayashi received the B.E. and M.E. degrees from Kyushu Institute of Technology, Tobata, Japan in 2006, and 2008. He is Graduate student of Graduate School of Engineering, Kyushu Institute of Technology in Japan. His research interests include wireless network, network architecture, and network security.



Yutaka Nakamura received the B.S. degree from Kyoto Institute of Technology, Kyoto, Japan, in 1996 and M.E. and D.E. degrees in information systems from Nara Institute of Science and Technology in 1998 and 2001, respectively. From 2001 to 2002 he was a Research associate in Engineering Science from Osaka University, Osaka, Japan. From 2002 to 2004 he was an Assistant professor at Information Technology Center in Nara Institute of Science and Technology, Nara, Japan. Currently, he is Associate Professor at Information Science Center in Kyushu Institute of Technology, Fukuoka, Japan. His research interests include technology for network monitoring, network operation and network security.

Takesih Ikenaga received B.E., M.E. and D.E. degrees in computer science from Kyushu Institute of Technology, Iizuka, Japan in 1992, 1994 and 2003, respectively. From 1994 to 1996, he worked at NEC Corporation. From 1996 to 1999, he was an Assistant Professor in the Information Science Center, Nagasaki University, From 1999 to 2004, he was an Assistant Professor in the Department of Computer Science and Electronics, Faculty of Computer Science and Systems Engineering, Kyushu Institute of Technology, Since March 2004, he has been an Associate Professor in the Department of Electrical, Electronic and Computer Engineering, Faculty of Engineering, Kyushu Institute of Technology. His research interests include performance evaluation of computer networks and QoS routing. He is a member of the IEEE.

Yoshiaki Hori received B.E., M.E., and D.E. degrees from Kyushu Institute of Technology, Iizuka, Japan in 1992, 1994, and 2002, respectively. From 1994 to 2003, he was a Research Associate in Common Technical Courses, Kyushu Institute of Design, Fukuoka. From 2003 to 2004, he was a Research associate in the Department of Art and Information Design, Kyushu University, Fukuoka. Since March 2004, he has been an Associate Professor in the Department of Computer Science and Communication Engineering, Kyushu University. His research interests include network security, network architecture, and performance evaluation of network protocols on various networks. He is a member of IEEE, ACM, and IPSJ.