

A New Design for a Practical Secure Cookies System*

JONG-PHIL YANG AND KYUNG HYUNE RHEE[†]

Department of Computer Science

[†]*Division of Electronic, Computer and Telecommunication Engineering*

Pukyong National University

Nam-gu, Busan, Korea

Because of the stateless character of HTTP, cookies were invented to maintain continuity and states on the Web. Cookies which have user-related information are transmitted and stored, so an attacker can easily copy and modify them for his own purpose. Therefore, cookies are exposed to serious security threats such as network threats, end-system threats, and cookie-harvesting threats. In this paper, we present a secure cookie system for solving these security weaknesses of typical web cookies. Since our system is based on the Public Key Infrastructure (PKI), it provides mutual authentication between clients and servers, and ensures the confidentiality and integrity of user information. We have implemented our secure cookie system and compare it here to the Secure Socket Layer (SSL) protocol that is widely used to provide the security in the HTTP environment.

Keywords: secure web service, security, authentication, cookies, public key infrastructure

1. INTRODUCTION

Nowadays, Web browsing and on-line shopping are rapidly increasing in popularity due to their convenience. A user can personalize a web page and have his/her own shopping cart, and then the web page can be automatically authenticated without entering a username or password repeatedly. A commercial web site has to provide a payment system, so it needs a security system to prevent attackers from exposing personal information or credit card numbers.

HTTP does not support continuity for browser-server interaction between successive visits of a user due to its stateless character. That is, when a web server finishes sending a response to a request of a client, it loses the information related to the client. Therefore, cookies were invented to maintain continuity and states on the Web. They are sent to the user's hard drive or RAM via the browser when the user visits a cookie-using website. The web server retrieves the user's information from those cookies when the user later returns to the same website. The cookies' purpose is to acquire information for use in subsequent server-browser communication without requesting the same information repeatedly [1, 6].

A merchant web server could use a cookie that contains the user's name and credit card numbers. Although this would be convenient for users, it would also be risky. Be-

Received December 9, 2003; revised September 1 & November 23, 2004; accepted December 20, 2004.

Communicated by Ja-Ling Wu.

* The preliminary version of paper was presented in the 3rd International Conference in India (INDOCRYPT 2002), Hyderabad, India, Dec. 16-18, 2002.

cause they are stored and transmitted in plain, cookies are readable and easily forged. One way to solve this problem is to apply cryptographic techniques, such as encryption, one-way hashing, or digital signatures, to typical web cookies. When some cryptographic techniques are applied to cookies, user information, such as user IDs, user passwords, and credit card numbers, can be stored in cookies, without being saved to a database on a server. Therefore, this method has some advantages in that it reduces maintenance cost of a database on the server-side and is more secure for typical cookies.

Due to the growth of e-commerce and e-banking markets, the user gradually possesses his/her own certificate for necessity. Therefore, a practical secure implementation which is based on the user's certificate can be implemented in the real Internet environment. Therefore, we have designed and implemented a new cookie system that is based on PKI. Our system provides several cryptographic services as well as improved defense against security threats. The rest of this paper is organized as follows. In section 2, we refer to typical cookies, security threats, and related works. In section 3, we propose a new secure cookie system and discuss its security issues. In section 4, we add security features of our system from the viewpoint of session tracking on a single server or in the Internet domain. In section 5, we present implementation of our system and make a performance comparison between it and SSL. Finally, we draw conclusions in section 6.

2. SECURITY THREATS AND RELATED WORKS

In this section, we will discuss typical cookies and some threats to them. Additionally, we will introduce our proposed solutions to security threats against cookies.

2.1 Typical Cookies and Security Threats

Cookies serve many purposes on the Web. For example, they are used to select display modes (for example, frames or text only), maintain shopping cart selections, and store user identification data. All cookies on the Web are fundamentally similar. Fig. 1 shows typical cookies structure and their fields [3]. The well-known security threats against typical cookies on the Web are as follows [6]:

	<i>Domain</i>	<i>Flag</i>	<i>Path</i>	<i>Cookie_Name</i>	<i>Cookie_Value</i>	<i>Secure</i>	<i>Date</i>
<i>Cookie</i> <i>1</i>	paper.net	True	/	Name_cookie	Yang	False	12/31/2003
			•		•		
			•		•		
<i>Cookie</i> <i>n</i>	paper.net	True	/	Role_cookie	student	False	12/31/2003

Fig. 1. Typical cookies on the web.

1. **Network threats:** Cookies transmitted in plain on a network are susceptible to snooping (for subsequent replay) and to modification.
2. **End-system threats:** Once cookies are in a browser's end system, they reside on the

hard drive or memory in plain. Such cookies can be trivially altered by users and easily copied from one computer to another, with or without the cooperation of the user on whose computer the cookie was originally stored. The ability to alter and copy cookies lets attackers easily forge cookies' information and impersonate other users.

3. **Cookie-harvesting threats:** If an attacker collects cookies by impersonating a site that accepts cookies from users (who believe that they are communicating with a legitimate web server), then the attacker can later use those harvested cookies on all other sites accepting them.

2.2 Related Works

V. Khu-Smith and C. J. Mitchell [8] distinguished two security schemes for cookies. One is the *server-managed cookie* approach, and the other is the *user-managed cookie* approach. The server-managed cookie has the major benefit of user transparency. If it is implemented appropriately, no changes to servers will be required. A disadvantage of this approach is the replay attack. But, with the user-managed cookie approach, a user has the benefit of control over what, when, and how security mechanisms are applied. However, a special web browser or additional software is required in order to enable users to perform security procedures.

J. S. Park and R. Sandhu [6] proposed a server-managed cookie scheme that consists of three types of secure cookies: For address-based authentication, the cookie value of the cookie grabs a user's IP address. When the user's IP address is dynamically assigned on his computer or the user's domain uses a proxy server, this is not desirable. In addition, it cannot prevent IP spoofing. For password-based authentication, the cookie value of the cookie grabs a user's hashed password. Password-based authentication supports dynamic IP addresses or proxy servers and prevents IP spoofing. However, this mechanism is inherently vulnerable to the dictionary attack. Digital-signature-based authentication does not verify the public key of the communicating entity by itself.

V. Khu-Smith and C. J. Mitchell [8] proposed a user-managed cookie scheme that is based on symmetric cryptography or asymmetric cryptography. This scheme provides several security services, such as authentication, integrity, and confidentiality, by using additional software.

Ours is basically a user-managed cookie approach. However, an extension of our system, which will be introduced in section 4, is a server-managed cookie approach. In order to implement public key encryption and digital signatures properly, we apply a public key certificate to our system. Moreover, when a user logs in a server with the proposed system, a password-based mechanism also can be applied to defend the whole system against end-system threats and cookie-harvesting threats.

3. DESIGN OF THE SECURE COOKIES SYSTEM

In this section, we will propose a new architecture for secure cookies based on a public key certificate and discuss some applicable secure protocols for our proposed secure cookies.

3.1 Notations and Architecture

We introduce some notations that are used to represent the proposed system in the following:

- C : the identity of a client (user);
- S : the identity of a server;
- $Passwd$: the password which is required for a user to log into the server;
- R : the random value which is generated by C ;
- PR_X : the private key of a communicating entity X on a public key cryptosystem;
- PU_X : the private key of a communicating entity X on a public key cryptosystem;
- SK : the secret key on a symmetric cryptosystem;
- $Cert_X$: the public key certificate of a communicating entity X ;
- T_X : the timestamp value of a communicating entity X ;
- $H(m)$: the one-way hash value of a message m ;
- $E_K(m)$: the message m encrypted with key K ;
- $SIG_K(m)$: the message m digitally signed with key K .

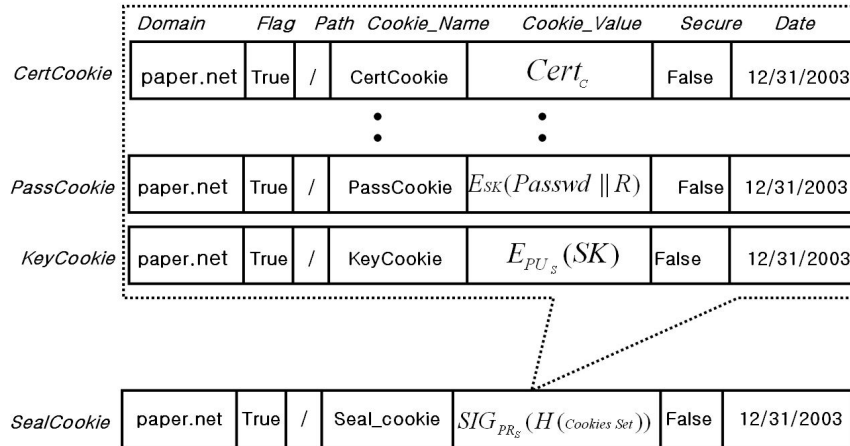


Fig. 2. The architecture of a secure cookies set.

Fig. 2 shows the architecture of the proposed secure cookie system. Some cookie families and their roles are given as follows:

- *CertCookie*: the cookie value is a certificate of a client (user);
- *PassCookie*: the cookie value is encrypted $Passwd$ and R with SK used to log into the server;
- *KeyCookie*: the cookie value is an encrypted SK with the server's public key PU_S ;
- *SealCookie*: The cookie value is a signed message digest of cookies with the server's private key PR_S .

Remark 1: In this paper, the set of cookies such as CertCookie, PassCookie, or Key-Cookie, is called a *cookie set*. A set of cookie and SealCookie are called *secure cookies sets*.

Additionally, it is possible for the administrator of a server to generate new cookies and securely add them to a *cookie set* when necessary (for example, for access control or to save user related information on the client-side) through encryption with *SK*. A secure cookie set provides integrity, confidentiality, and authentication services by using several cryptographic techniques.

3.2 Issuing a Secure Cookie Set

Fig. 3 shows the procedure for issuing a secure cookie set. We assume that each communicating entity has a public key certificate that is issued by the CA (Certificate Authority).

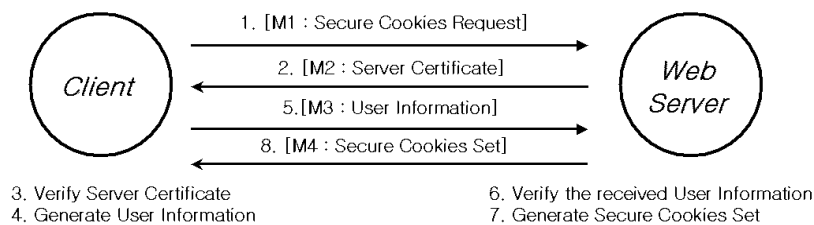


Fig. 3. The procedure for issuing secure cookie set.

As shown in Fig. 3, when a client requests that a server issue a secure cookie set (that is, a client sends message-1, *M1*, to a server), the server sends message-2 (*M2*) to the client:

$$M2 : Cert_s.$$

Now, the client can authenticate the server by verifying the server's certificate and obtain the public key of the server, PK_s . It can generate a secret key, SK , that can be used to encrypt some cookie values and a random value, R , which can be used to prohibit an attacker from performing a dictionary attack. Then, it requests that a user define a password which is then used to log into the server. The client configures message-3 (*M3*) and sends it to the server. The client deletes SK from local memory immediately after sending *M3*:

$$M3 : Cert_C \parallel E_{PK_s} (Passwd \parallel R \parallel SK \parallel SIG_{PR_C} (H(Passwd \parallel R \parallel SK))).$$

The server authenticates the client by verifying the client's certificate and obtains the public key of the client, PK_C . The server decrypts the encrypted part of *M3* with its private key PR_s and obtains the client's secret information, such as $Passwd$, R , and SK .

The server verifies the signed part of $M3$ with PU_C , identifies the origin of $M3$, and confirms no fault. After verifying $M3$, the server configures a secure cookie set and sends message-4 ($M4$) to the client. The server deletes $Passwd$, R , and SK from the client's local memory immediately after sending $M4$:

$$M4 : CertCookie \parallel \dots \parallel PassCookie \parallel KeyCookie \parallel SealCookie .$$

In $M4$, the cookie value of $SealCookie$ is

$$SIG_{PR_S} (H(CertCookie \parallel \dots \parallel PassCookie \parallel KeyCookie)) .$$

3.3 A Login Procedure with a Secure Cookie Set

Fig. 4 shows a login procedure that uses a secure cookie set. When a client requests that a server log in (that is, a client sends message-1, $M1$, to a server), the server sends message-2 ($M2$) to the client:

$$M2 : Cert_S \parallel T_S .$$

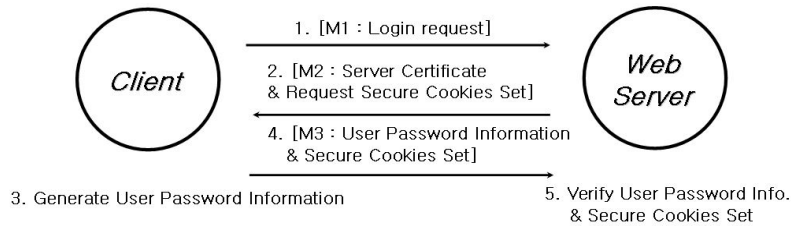


Fig. 4. The login procedure using a secure cookie set.

After receiving $M2$, the client verifies the certificate of the server, authenticates the server, and obtains the valid PU_S . To log into the server successfully, the client requests that a user input his password. The user's password and the received T_S are encrypted with the public key of the server. The client composes message-3 ($M3$) and sends it to the server. $M3$ consists of the previously encrypted part and a secure cookie set:

$$M3 : E_{PU_S} (Passwd, T_S) \parallel CertCookie \parallel \dots \parallel SealCookie .$$

The server decrypts the encrypted part of $M3$ with PR_S and obtains $Passwd$ and T_S . The server compares the received T_S with the one which is in $M2$. If the two values are equal, then $M3$ is fresh. Next, the server calculates a hash value of a cookie set, such as $CertCookie$, $PassCookie$, or $KeyCookie$ in $M3$, and then signs the hash value. The server compares this value with the cookie value of $SealCookie$. If the two values are equal, the received secure cookie set has not been forged by any attacker, and it is the secure cookie set which has been issued by the server itself.

Now, the server decrypts the cookie value of *KeyCookie* with PR_S and obtains SK , and it decrypts the cookie value of *PassCookie* and obtains $Passwd$. The server compares $Passwd$ in *PassCookie* with the encrypted $Passwd$ with PU_S in $M3$. If the two values are equal, the server confirms that a legitimate user is sending the secure cookie set. Finally, the server verifies the validity of the certificate in the cookie value of *CertCookie* and authenticates the client.

3.4 Security Properties

We assume that all of the cryptographic techniques used in this approach provide enough security to defend against attackers' computational ability. The proposed secure cookie set provides the following security services:

- **Mutual Authentication:** Whenever a secure cookies set is issued and a user wants to log in, the client and server always exchange their public key certificates and verify them. Hence, the proposed system solves the key verification problem of Park's scheme [6].
- **Confidentiality:** The cookie values in a cookie set can be encrypted with SK .
- **Integrity:** The cryptographic one-way hash function is applied to generate the cookie value of *SealCookie*.
- **Originator Authentication:** The digital signature technique is applied to generate the cookie value of *SealCookie*. Hence, the signer of the cookie value of *SealCookie* is uniquely identified.

We can also solve the following security threats which were introduced in section 2.

1. **Network threats:** Because of confidentiality, integrity, and originator authentication of our scheme, it is possible to defend cookies against network threats. Moreover, since the timestamp value shown in Fig. 4 is used, it is impossible for an attacker to replay.
2. **End-system threats:** An attacker cannot modify cookies, but can copy them to someplace else (for example, his hard-drive). However, if the attacker wants to forge the original user of cookies, he should know the user's password $Passwd$. Moreover, it is difficult for an attacker to succeed with a dictionary attack on our system.
3. **Cookie-harvesting threats:** If an attacker wants to know the plaintext of the cookie values of cookies, he must know the private key of the server. Then, if the attacker wants to reuse the collected cookies for malicious purposes, he must know the user's password, $Passwd$.

In the proposed system, through the use of public key certificates, all the communication entities can authenticate each other and verify the authenticity of the public key directly in each certificate. Also, through $Passwd$, the proposed system can prevent an attacker from using cookies for his own profit.

4. EXTENSION OF THE SECURE COOKIE SET

In this section, we will extend the function of the secure cookie set that was introduced in section 3.

4.1 Authenticated Session Tracking in a Single Server

When a user moves from one web page to another web page in a single server or makes a new connection in the same HTTP session, cookies can be used for session tracking. After the login procedure shown in Fig. 4 is performed, the server issues *STCookie* and sends it to the client with the web page.

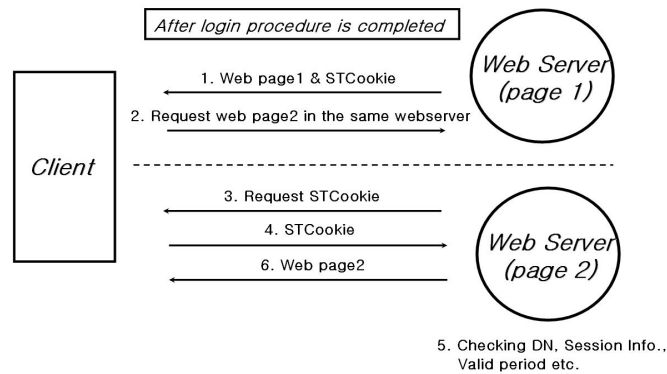


Fig. 5. The procedure for authenticated session tracking in a single server.

Fig. 5 shows the procedure for authenticated session tracking in a single server. *STCookie* is only used for session tracking during a single session that lasts for a short period of time. The cookie value of *STCookie* is

$$SI \parallel C_{DN} \parallel VP \parallel \text{SIG}_{PR_S}(H(SI \parallel C_{DN} \parallel VP)), \text{ where}$$

- *SI*: the HTTP session information;
- *C_{DN}*: the *DN* (Distributed Name) in the certificate of a client;
- *VP*: the valid period *STCookie*. If the valid period is too short, the possibility of a replay attack is very low, but the user has to perform the login procedure frequently. If the valid period is too long, although the user is convenient, the possibility of a replay attack will be increased.

4.2 An Authenticated Login in a Multiserver

If there are several servers in the same Internet domain, it is possible for a client to securely move from one server to another by using cryptographic cookies. We make the following three assumptions. First, a client has securely logged into a server through our

system. Second, there are several servers (S_i , where $1 \leq i \leq n$) in the same Internet domain. For example, *a.paper.net*, *b.paper.net* and *c.paper.net* may be servers which are in the same Internet domain *paper.net*. Third, servers in the same Internet domain trust each other; that is, each server knows the public keys of the others.

When a client wants to move from one server (S_i) to another server (S_j), a user clicks on a hyperlink that is linked to a web page of S_j , and the client can simply log in S_j through *CrossCookie*. Fig. 6 shows how *CrossCookie* is used when a client moves S_1 to S_2 , which is hyperlinked to a web page of S_1 . Using *CrossCookie*, a client can securely move from S_1 to S_2 . The cookie value of *CrossCookie* is

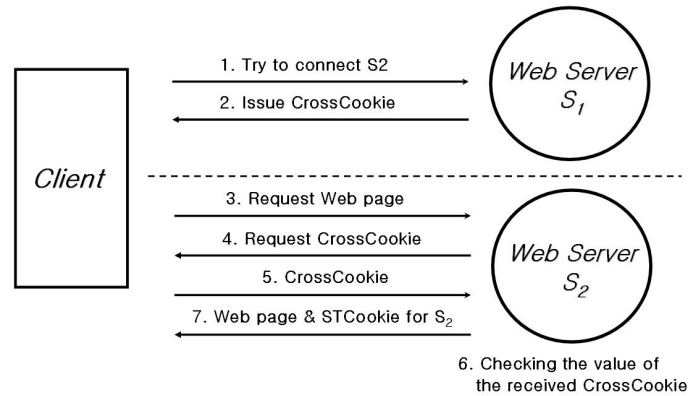


Fig. 6. Authenticated login in a multiserver through *CrossCookie*

$$S_{i_{DN}} \parallel S_{j_{DN}} \parallel C_{DN} \parallel VP \parallel SIG_{PR_{S_i}}(H(S_{i_{DN}} \parallel S_{j_{DN}} \parallel C_{DN} \parallel VP)), \text{ where}$$

- $S_{i_{DN}}$: the certificate *DN* (Distributed Name) of S_j that is connected to a client at present, where $1 \leq i \leq n$;
- $S_{j_{DN}}$: the certificate *DN* of S_j that a client is willing to connect, where $1 \leq i \leq n$;
- C_{DN} : the certificate *DN* of a client;
- VP : the valid period of *CrossCookie*. There is only a very short time period for considering *STCookie* since the time which a client has to move to a different server is very short.

5. AN IMPLEMENTATION OF THE SECURE COOKIE SET

In this section, we will present our implementation of the secure cookie set and consider its performance.

5.1 The Implementation Environment

In our implementation, the following environment is used:

- language: Java (JDK 1.3), JSP, CryptixJCE, Java Web Start V1.01;
- hardware: Pentium IV 1.80 GHz, 256 MB RAM;
- web server: Apache 1.3.19 (for Win32), Jakarta Tomcat (3.2.1);
- network: 10Mbps Ethernet;
- cryptographic algorithm: RSA 1024bits, MD5, DES.

HTML and JSP are used on the server-side, and Java Web Start is used on the client-side. Since the client program is implemented using Java Web Start, it does not require any modification of an ordinary web browser, and it can naturally cooperate with an ordinary web browser to support the proposed protocols. Also, we use a self-signed certificate for the client and server [2, 4, 5, 7].

5.2 The Implementation Results and Considerations

When a user connects to a web page of a server and moves to a web page for issuing cookies, a *cookie issuing program* is automatically executed on the client. The user inputs his real name, which is the same one in the DN of his/her certificate, and two passwords to log in. One is a password (*Passwd*) used by the server to identify the user, and the other enables the user to use his private key correspond to the public key in his/her certificate. After completing the issuing procedure, the client obtains a secure cookie set on the local hard drive.

When the user tries to log into the server, the *login program* is automatically executed on the client. In this case, the user only inputs a password (*Passwd*) which is used by the server to identify the user and to defend cookies against the threats that were introduced in section 2. After completing the logging procedure, the user succeeds in login into the server. Fig. 7 shows the two programs which are used by the client. Fig. 8 shows the secure cookie set that is stored on the client's local hard drive.

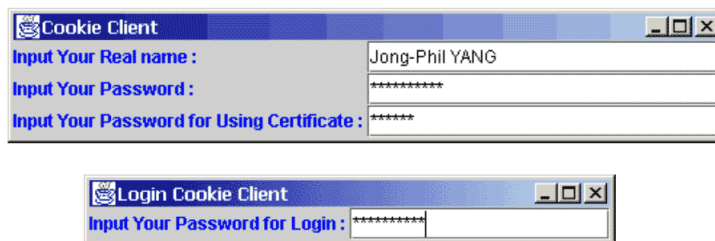


Fig. 7. User agent program for issuing a secure cookies set and for logging in.

We compared the performance of SSL (Secure Socket Layer) with that of the implementation of our proposed system. Currently, many web sites still use SSL for temporary protection of critical information, such as IDs or passwords, during a login procedure. One can see an example in Microsoft web mail service (Hotmail: <http://www.hotmail.com>). Therefore, when we consider some environments where temporary protection of information is required (e.g., when exchanging login information or session

```
CertCookie
B9C5F6647BA91D8A84CA4DA06159AFE434F4FD5573FF9EF3C82438F3A2305F7A17A4E6C682BFCADC
CED34C2E91E99957C7141A1669652F06F7349EA0263667195446C67B250E59AFA5D19474A98CD9F8C
229ECA9B6B765C0FC7D23D238DF7C5FCE9E670273A5408BDD59CE02B946ABED7DED4B43459D5F78
7FCD0435C65AFCB6B343CFA9E3D3F5ADC78ED4AAAA36F116E87943BF852A6FF54AAEC3E5332156B
1EB5B35231E6975722CF24B2CD9404013BF24F84F36EB450D2EA5AD40943A8D2C404808960ACDA019
C79BF4E3BD7227B8EFBE35CE4ECBF1E849E821A248529BD9E7E0E177BBE07A58E7CF48DE07E82AE6
00FB362C1B853F2533F7FD9FD1EE86C372E1DBCC7E22694DDCAC648E37BB47645EB01CC48CF4A6
AD40A6F09E62D202DE87943BF852A6FF531346D18ECC35E3A963559A5E376036845FE6965C864041D
722BB378AE75FDC2280457E092D750989EC79DB3A0014886C32E83B746247E06AD54058F968036364A
4D6797731C4BCEED0F6D3579B4077E77498988C134B1A5F8A2B19D34526C4D2DC86F35BB21C64B90
DEE82146A29A77371EF4B2D1FE9BB2AF2E13918C3CB654FEA65BF636BF4D4E13B65B1717FCD8AD8E
6921967143B62EA53AF38862CE99560A3BA3D45542FB52BB34B848B270CFFABE7A66D1712B0EF6F3
BD557BD546821C86F3A1EF0F2C48C5642D6C46EB6A4360AA6CDB531CBDF80EC6CB5AFC75BFC979B
1832BBBA054CAF5E5A4862163F02C4C09095DFFD072120D68923CC084C05136623D244DEE58FF957E
34F96E1E8A13COD121F308EA022DE7D1972381BFBD6A1530AEA56835030F03D461F3A46DE38698
gamza.lisia21.net/
1024
2765120896
29540274
742829696
29466849
*
PassCookie
A5D567CD4216D2CE938C317270680C9A
gamza.lisia21.net/
1024
2765120896
29540274
742929696
29466849
*
KeyCookie
D5297234E25D506CFC8EC0A75C7DF7D7E6B7D59ED527424C50D46A784876E5ED0D84D9A364396F43
87244DCD8EF3A033260AD66D31493DAA5A8D8D869CF9FFD8550C87D9D10BB09708C4C973ABBD0C
906408CD420A7EA595F01CEAC2B7CD7828DC242EB49B467A134A7F3FA80F540EC754C33A585A940C
434938B176C63661F
gamza.lisia21.net/
1024
2765120896
29540274
743029696
29466849
*
SealCookie
4E94ED0230DEFDA649B040C912FAF1389A2880BE5B7BC3A0729DB36404F6EDE65C25CCCB7D7F358A
2BD965AC4336E84C9879F89E7611E9B8AF580BAA407B7782ADE3AFF98BC4343DB8E1EFD4821F2934
38D761DAD2C98879F458E2AC287AF5DCD82D778E0FC192D5F63CB9D5FCBA5B33E8CCC92608C2CA1
CD19302F4869AC1CF
gamza.lisia21.net/
1024
2765120896
29540274
743129696
29466849
*
```

Fig. 8. Secure cookie set which is stored on the user's local hard drive.

Table 1. A performance comparison between SSL and the proposed scheme.

Time for issuing a secure cookie set	4.439 sec
Time for logging in through a secure cookie set	4.201 sec
Time for establishing a connection through SSL	5.604 sec

tracking information), a performance comparison between SSL and our system is necessary and meaningful. To compare their performance, we used the SSL of JSSE 1.0.2 in the fullhandshake mode with client-server mutual authentication [7]. Table 1 shows the results of the performance comparison.

Our proposed system is based on the X.509 certificate in PKI for secure login and session tracking. Since ordinary cookies in HTTP are in plaintext, it is possible to produce cookies in one round-trip of HTTP communication. However, there is no way with the current cryptographic techniques to perform cryptographic operations (e.g., digital signatures, encryptions) without additional round-trips (e.g., the exchange of an X.509 certificate for authentication or a key agreement procedure for sharing a secret key). Therefore, our system also cannot avoid the need for additional round-trips to produce secure cookie set.

6. SUMMARY AND CONCLUSION

In this paper, we have presented a way to overcome security threats to cookies using a new cookie set which provides confidentiality, integrity, and mutual authentication between a server and a client. Moreover, in addition to *CertCookie*, *PassCookie*, and *KeyCookie*, it is possible for the administrator of a server to generate new cookies and securely add them when necessary through encryption using *SK*. Our system also provides authenticated session tracking in a single-server and authenticated login on different servers in the same Internet domain. Because user related information is stored in cookies instead of a database on the server, the maintenance cost of the server can also be reduced.

ACKNOWLEDGMENTS

This research was supported by the Program for the Training of Graduate Students in Regional Innovation which was conducted by the Ministry of Commerce, Industry and Energy of the Korean Government.

REFERENCES

1. <http://www.certcc.or.kr/advisory/ka2000/ka2000-041.html>.
2. <http://www.cryptix.org/products/jce/index.html>.
3. http://www.netscape.com/newsref/std/cookie_spec.html.
4. <http://java.sun.com/products/javawebstart/developers.html>.

5. <http://java.sun.com/products/jsp/download.html>.
6. J. S. Park and R. Sandhu, "Secure cookies on the web," *IEEE Internet Computing*, Vol. 4, 2000, pp. 36-44.
7. S. Oaks, *Java Security*, 2nd ed., O'Reilly, 2001.
8. V. Khu-Smith and C. J. Mitchell, "Enhancing the security of cookies," K. Kim, ed., in *Proceedings of 4th International Conference on Information Security and Cryptology (ICISC 2001)*, Springer-Verlag, LNCS 2288, 2002, pp. 132-145.



Jong-Phil Yang received his B.S. and M.S. degrees in Department of Computer Science from Pukyong National University, Korea in 1999 and 2001, respectively. He is currently a Ph.D. candidate in Graduate School of Computer Science, Pukyong National University. His interests are related with information security and network security; public key infrastructure, fault-tolerant system, security for mobility and applied cryptography.



Kyung Hyune Rhee received his M.S. and Ph.D. degrees from Korea Advanced Institute of Science and Technology, Daejeon Korea in 1985 and 1992, respectively. He worked as a senior researcher in Electronic and Telecommunications Research Institute, Daejeon Korea from 1985 to 1993. He worked as a visiting scholar in University of Tokyo, Japan in 1999, and visiting professor in University of California, Irvine, U.S.A. during 2001 to 2002. He has also worked as a chair of Information and Communication Division, Colombo Plan Staff College in Manila, Philippine, during 2002 to 2003. He is currently a professor in the Division of Electronic, Computer and Telecommunication Engineering of Pukyong National University. His research interests center on key management and its applications, mobile communication security and security evaluation of cryptographic algorithms.