Enhancing the security of cookies

Vorapranee Khu-smith and Chris Mitchell

Information Security Group, Royal Holloway, University of London,
Egham, Surrey, TW20 0EX, United Kingdom
{V.Khu-Smith, C.Mitchell}@rhul.ac.uk

Abstract. Cookies are pieces of information generated by a Web server to be stored in a user's machine. The information in cookies can range from selected items in a user's shopping cart to authentication information used for accessing restricted pages. While cookies are clearly very useful, they can also be abused. In this paper, security threats that cookies can pose to a user are identified, as are the security requirements necessary to defeat them. Various options to meet the security requirements are then examined. Proposed user-controlled approaches and their implementations are presented and compared with a server-controlled approach, particularly the 'Secure Cookies' method, to illustrate the relative advantages and disadvantages of the two approaches.
Key Words: Cookies; Internet security; Web security

1 Introduction

Today, browsing and on-line shopping are becoming increasingly convenient. A user can personalise a Web page, have his/her own shopping cart, and be automatically authenticated without repeatedly entering username and password. However, the stateless nature of the HTTP protocol does not support such features. Therefore, cookies were introduced to enable Web servers to maintain current session state and recognise individual users.

Cookies are pieces of information generated by a Web server to be stored in a user's machine. The information in cookies can be, for example, selected items in a user's shopping cart, authentication information used for accessing restricted pages, or account details. The first time a browser contacts a Web server, a cookie is sent from the latter to the former. It is then stored in the user's PC in a file called either cookies.txt, Cookies, or MagicCookie in the Windows, UNIX, and Macintosh operating systems respectively. The next time the browser requests a Web page from the Web server, it sends the corresponding cookies.

A cookie consists of six elements, namely Name, Expiration Date, Domain name, Path, Secure, and String Data. The first part is the name of the cookie. The expiry date defines the cookie's lifetime. Domain name and path are used when a browser searches for a cookie corresponding to the host of the requested URL. The path attribute specifies the subset of the URLs to which the cookie belongs. The secure attribute indicates whether the cookie is transmitted in secure mode such as TLS and HTTPS. The String Data field is where all other

information of the server's choice is stored. Detailed cookie specifications can be found in [6] and [8].

While cookies are clearly very useful, they can also be abused to impersonate a user, compromise user privacy and, in some cases, reveal confidential user information. Although a number of papers, e.g. [2, 3, 7, 11], point out potential security threats, most of them focus on facts about cookies, such as what they are and how they are used, and do not appear to provide a satisfactory security analysis and solution.

In [9] the 'Secure Cookies' method is proposed, in which security measures are applied to cookies by a Web server. Consequently, this approach allows Web servers to control what, when and how the security procedures will be performed. Whilst this may be appropriate for many applications, in some environments users will wish to control the security of their own cookies. This paper, therefore, examines possible user-controlled approaches to enhance cookie security.

In this paper, the threats that cookies can pose to a user are identified, as are the security requirements necessary to defeat them. Various options available to meet the security requirements are examined. A server-controlled approach is then outlined followed by proposed user-controlled approaches and their implementations. A comparison between the server-managed and user-managed approaches is subsequently given. The final section summarises and concludes the paper.

2 Cookie-Related Security Threats

In this section, security threats that cookies can pose to a user are identified. The threats can be divided into three main categories namely confidentiality of cookie contents, monitoring user behaviour using cookies, and malicious cookies.

2.1 Confidentiality of Cookie Contents

In order to allow users to browse among restricted pages without repeatedly identifying themselves, cookies are used to store authentication information such as username and password. Consequently, it is important to ensure the confidentiality and authenticity of cookies storing such information. Otherwise, anyone who can access such cookies can potentially impersonate the user. Although, in many cases, authentication information stored in cookies is in a server-specific format, and hence the contents are not immediately obvious to the reader, it is still possible for an attacker to simply replay an intercepted cookie and impersonate a user.

Implementation flaws, particularly in Web browsers, can also bring about security risks to a user. For example, a vulnerability in Internet Explorer Version 4 and 5 for Windows 95, 98, NT, and 2000 allowed any sites to see the content of other sites' cookies [4]. This is because the browser confused a site having a long URL ending with the domain name of another server with that other server. Consequently, it was possible for a malicious web site to give itself a

long URL ending with a sequence of characters identical to the URL of another site, and the malicious site was then able to access cookies stored by that other site. If a cookie contains personal information, e.g. confidential data such as account details, then the consequences of such a vulnerability can be significant. Although the Web browser flaw has been fixed, it is possible that there are other undiscovered vulnerabilities that can pose security threats to users.

Another example of an implementation flaw lies in the way that some users include a Netscape Navigator folder in a publicly accessible directory. In an environment where there are not as many computers as users, it is not unusual to provide public spaces for users to access their data from any computers within the environment. Such spaces are accessible to any users with a valid username and password. For example, a college can provide a drive for students to store their home pages. In this drive, each student has his/her own directory. Any students with a valid username and password can access the drive, and hence other students' directories. An empirical study [5] showed that a number of cookie files could be found in this publicly accessible drive. This is because some users had stored their Netscape Navigator folder in such a publicly accessible directory. Since Netscape Navigator stores user cookie files in the user's Netscape folder, this means that user cookies will also be stored in a publicly accessible location, i.e. accessible by any other users with a legitimate username and password. The study showed that it was possible to use this weakness to obtain personal details, ranging from user names to full user details including contact addresses and telephone numbers.

2.2 Monitoring User Behaviour Using Cookies

A particularly controversial issue concerning cookies is that they can be used as tracking devices to follow user movements across the Internet. Web-advertising agencies such as DoubleClick, Focalink, Globaltrack, and ADSmart run advertisement banners on various sites. Their clients add an <IMG> tag to the client HTML page, pointing to a URL on the advertising agency's server. When a Web browser sees this <IMG> tag, it contacts the advertising agency server to retrieve the graphic. The first time the graphic is downloaded from the site, the user browser will receive a cookie containing a random ID. From then on, every time the browser connects to a site containing the agency's advertisement banners, it sends the cookie (the random ID) along with the URL of the page that is being read [11].

After a period of time, the advertising agency will be able to generate a user profile, revealing user browsing habits and interests. This might be used to improve advertising campaigns, to target advertisements to user interests, and to avoid repeatedly showing the same advertisements to a user. The ability to track users is a potential violation of user privacy. It is also possible that advertising servers might share such information without user consent although, at the time of writing, there is no strong evidence of such behaviour.

Even though using cookies as a tracking device may not reveal the actual identity of a user, the fact that an advertising agency server can maintain a

list of URLs that a user has viewed can lead to a possible leak of user personal details. In particular, if a Web server uses the GET method to input data from an HTML form to a CGI script, the information will be sent as a part of the URL. Therefore, anyone who can read the URL will be able to obtain the information in the form.

2.3 Malicious Cookies

It is often rather misleadingly stated that cookies are just text files, and hence are harmless. Although cookies are application data files, cookies can include special tags that can introduce executable code, just as Microsoft Office application files can contain Macro viruses.

In HTML, in order to distinguish text from 'markup' symbols, a set of characters such as '<', which typically indicates the beginning of an HTML tag, are defined as special. Tags can either affect the formatting of a Web page or introduce a program that will be executed by Web browsers. For example, a <SCRIPT> tag introduces code from a variety of scripting languages [10]. Many Web servers use information stored in cookies to create dynamic pages. Therefore, if a cookie includes those special tags, when a page incorporating this cookie is displayed, a malicious program can be called and executed. The security effects of such a program can range from alterations of the submitted form to bypassing an authentication process. However, what exactly can be done by the called program depends on the language in which it is written as well as the Web server's security context configuration.

3 Security Requirements

In this section, a number of security requirements are identified to deal with the security threats discussed earlier. Note that these security requirements do not address the threat of cookies being used to monitor user behaviour. Such threats are typically tackled by using tools specially designed to monitor the activity of cookies.

3.1 Cookie Confidentiality

As stated in section 2.1, cookies can be used to store authentication data and personal details such as mailing addresses and credit card numbers. Therefore, it is important to provide confidentiality for such information. Information in cookies might be revealed in two major ways. Firstly, a cookie can be intercepted while it is transmitted, and, secondly, a cookie can be disclosed when it is stored in a user's machine. In general, it is a good practice to provide confidentiality for both scenarios.

3.2 Cookie Integrity

In order to prevent attacks such as cross-site scripting, i.e. where special tags are inserted into cookies as described in Section 2.3, maintaining the integrity of cookie data is vital. Moreover, if a cookie is used for user authentication and the content of the cookie is changed, then the authorisation process will fail. An attacker could modify such a cookie, and hence prevent a legitimate user from accessing a service. Domain name and Path in cookies are also important. If it is possible to make changes to these elements, cookies can be sent to an entity who is not a legitimate owner.

3.3 Cookie Authentication

Although the content of a cookie may be encrypted and protected from unauthorised modifications, there remains the possibility of an attack where one entity 'presents' a user authentication cookie copied from another party. Consequently, it is important to be able to verify if the entity that is supplying a cookie is the owner of that cookie.

4 Meeting the Security Requirements

There are various ways of meeting the security requirements stated above. This section examines the possible options in more detail, and considers their advantages and disadvantages.

4.1 Secure Channels

The first approach is to protect the cookie transmission channel, i.e. the link between a Web browser and a Web server, by using secure protocols such as TLS and HTTPS. This provides confidentiality and integrity by use of an encrypted channel and MACs respectively. However, a secure channel only protects the information against eavesdropping and modifications en route. Once the cookie reaches its destination it is stored in clear text, and hence this option provides only partial protection. Anyone who has access to the stored cookie file will be able to read, change, or replay it.

An advantage of this option is that it is transparent to a user. Moreover, it makes use of existing capabilities, and therefore does not require modifications to Web browsers. However, in order to send a cookie securely, the cookie's 'Secure' attribute must be set. Since cookies are generated by servers, it is completely in the server's hands whether the 'Secure' attribute is set. Given that most users are not aware whether or not cookies are sent via a secure channel, many Web servers send them in clear.

Secure channels can provide user authentication; however this does not guarantee the origin of a cookie. In order to provide cookie authentication, there should exist some means to link the user authentication used in the secure channel establishment process with the cookie itself.

4.2 Access Control for User PCs

Another way of providing security for cookies is to protect user PCs against unauthorised access. A user authentication technique, e.g. using passwords, can restrict access to a PC, thereby protecting stored cookies against unauthorised reading and modification. The main advantage of such an approach is its simplicity. It is relatively easy to implement since most users are accustomed to using passwords. There is also no need to modify Web browsers.

A disadvantage of this mechanism is that it only protects cookies while they are in a PC. Therefore, it may have to be employed with other mechanisms to enhance security. Furthermore, it does not provide cookie authentication, since a malicious user can still reuse a stolen cookie. The use of passwords may also require additional management, such as password tests to prevent dictionary attacks.

4.3 Cryptographic Protection within Cookie Files

A combination of cryptographic techniques can be used to meet all three security requirements. Cookie encryption can be used for confidentiality. Both cookie integrity and cookie authentication can be provided by a MAC or digital signature. As part of cookie authentication, cookie replay protection can be achieved by incorporating cookie transfer into an authentication protocol (e.g. using a time stamp).

Cookie encryption and integrity protection can protect a cookie both when it is stored and when it is transmitted, as opposed to the use of secure channels, which do not protect stored cookies (see Section 4.1). There is thus no need for additional access control to user PCs. However, a disadvantage of using cryptographic techniques is that keys are required. Key management issues, such as how to securely exchange the keys, where they should be stored, and who should keep them, have to be taken into account. Additionally, there is a possibility that Web browser modifications or additional software will be required.

4.4 Summary

The three techniques discussed above can be effective, and, in practice, a system can combine some or all of these methods to enhance security. However, the last technique, i.e. applying cryptographic protection to the cookie file itself, appears to offer the widest range of security services. For this reason, this approach is the focus of the remainder of the paper.

Applying cryptographic measures to cookie files can be performed by Web servers or Web browsers. Whoever does so will have control over what, when and how the measures are performed. In the remainder of this paper, we examine the two approaches, and consider their respective merits.

5 Server-Managed Cookie Encryption

Server-controlled cookie encryption has the major benefit of user transparency. If implemented appropriately, no changes to Web browsers will be required. A disadvantage of this approach is obviously that users will have little control over the protection of their own cookies. An example of this approach is the Microsoft Passport scheme [1] which was introduced to provide an online user-authentication service. It employs encrypted cookies as a means for exchanging user-authentication information between a Microsoft Passport server and participating web sites.

Another example of server-managed cookie encryption is the 'Secure Cookie' scheme proposed by Park and Sandhu [9]. Although these two schemes are similar in the way that they both use encrypted cookies, the latter is more general, in that it is a means of protecting all user cookies, and not just those cookies generated by a single application. As a result, we use the Park and Sandhu 'Secure Cookie' scheme as the basis for a comparison with the new cookie security scheme proposed in Section 6. In the remainder of this section, we provide a brief overview of the Park and Sandhu scheme.

In this approach, Web servers are required to use 'Secure Cookies' of specific kinds, each with predefined types of content and protection. Examples include Name Cookies, Life Cookies, Key Cookies, and Seal Cookies. A Name Cookie, for example, contains a user name that can be used for user authentication. A Key Cookie contains an encryption key. The integrity of all cookies is protected by a Seal Cookie that holds either a MAC or a signed hash of the other cookies.

In order to have a set of Secure Cookies, a Web browser needs to contact another server called the Cookie Issuer, which generates the Secure Cookies. The Web browser then sends the cookies to the Web server, which will verify or decrypt them as appropriate. Examples of Secure Cookies are listed in Table 1.

Table 1. Secure Cookie Components

Domain	**Flag**	**Path**	**Cookie_Name**	**Cookie_Value**	**Secure**	**Date**
acme.com	True	/	Name_cookie	Alice	False	12/31/2000
⋮	⋮	⋮	⋮	⋮	⋮	⋮
acme.com	True	/	Life_cookie	12/31/99	False	12/31/2000
acme.com	True	/	Pswd_cookie	Hashed password	False	12/31/2000
acme.com	True	/	Key_cookie	Encrypted key	False	12/31/2000
acme.com	True	/	Seal_cookie	Signed Message Digest of MAC	False	12/31/2000

This approach satisfies the security requirements of confidentiality, integrity, and user authentication by using encryption, a signed message digest or MAC, and a digital signature respectively. However, it does not provide protection

against replay attacks. A stolen Secure Cookie can be submitted to the Web server.

The Key Cookie, as stated earlier, stores a session key that is used to encrypt and decrypt other cookies. The session key can be encrypted using either a server public key or a server secret key. In either case, Web servers are responsible for key management.

While this approach may be appropriate in many applications, in some circumstances users may wish to read their own cookies and control their security. Consequently, in the next section we examine possible approaches that give users more control.

6 User-Managed Cookie Encryption

With the user-managed approach, users obviously have the benefit of control over what, when and how the security mechanisms should be applied. However, a special Web browser or additional software is required in order to enable users to perform the security procedures. The client may also have to store cryptographic keys, which could be a security threat in some circumstances. As a result, there is a need for a key management system to support the use of cryptography.

In this section, two possible approaches, using symmetric and asymmetric cryptography, are described. In order to provide cookie confidentiality, integrity and authentication, the schemes use encryption, MACs, signatures, and time stamps. The security mechanisms described below will be applied only to the cookie value, to minimise the complexity of the protocols.

6.1 Using Symmetric Cryptography

In this approach, a user selects cookie encryption by sending a request for cookie encryption to the Web server. This will trigger a key establishment protocol. If a user chooses to encrypt cookies, he/she will be required to authenticate him/herself to prevent unauthorised users, who may have access to the user's PC, from activating the security procedure and using cookies. This can be achieved by using simple schemes such as passwords. Key establishment can be performed by sending the key via a secure channel or using other key distribution techniques such as those involving a trusted third party. After successful key establishment, the user and the server then share a symmetric cookie key and the server will encrypt cookies with the key and send them to the user. The encrypted cookies are then stored locally in the user's PC.

The next time the Web browser requests a Web page from the site, it looks for corresponding encrypted cookies. A time stamp is generated and concatenated with the cookies, and a MAC is computed on this data. A page request, the time stamp, the encrypted cookies and the MAC are then sent to the server. On receipt of the request, the server verifies the MAC and checks whether the time stamp is within the acceptance window. The server can change information in

the cookies whenever the user requests a page, and the new encrypted cookie will be sent back to the user with the requested page.

A time stamp and a MAC are included in order to prevent an intercepted cookie from being replayed. Without knowing the secret cookie key, an adversary will not be able to create a valid MAC. There are no cryptographic requirements for time stamp generation — the time stamp only needs to be within the acceptance window.

Given that symmetric cryptographic operations are typically simple to compute, the encryption operation will not significantly increase the server workload. Users only need to decrypt a received cookie if they want to see the content. However, this approach needs a secure mean to distribute the symmetric key the first time the user and server communicate. Moreover, users and servers need to maintain the shared secret key. As the number of users (n) grows, the number of keys increases approximately to n^2, and the task of key management will therefore become increasingly complicated over time.

Since the security of this approach depends on the secrecy of the shared key, it is vital to store the key securely. A user can store the keys in a smart card or in his/her PC (password protected).

6.2 Using Asymmetric Cryptography

In this approach, users are required to have a certificate and an asymmetric key pair. If a user wishes to have his/her cookies encrypted, the first time the user requests a Web page his/her certificate and the page message will be signed and sent. The server then generates a secret key, encrypts the cookies with this secret key, encrypts the secret key with the user's public key, and sends the encrypted secret key with the encrypted cookies and the requested page. There is no need for the user to decrypt the cookie unless he/she wants to know the cookie content. The next time the user contacts the server, the encrypted cookie, a time stamp, and a MAC is sent with a Web page request.

As for the symmetric technique, this approach allows users to decide if they want to encrypt the cookie or not. If the user sends a certificate, the server will know that the cookie must be sent encrypted. The user will also be required to enter a password for user authentication, since his/her private key may be stored locally in the PC. However, if it is stored in another more secure way, e.g. on a smart card, user authenticity verification may not be required. As for the symmetric cryptography approach, the time stamp is used to prevent replay of an intercepted cookie.

A drawback of this technique is that certificates and key pairs are required for the client. A Public Key Infrastructure (PKI) will also be needed to create and manage public key certificates.

7 Protocols for User-Managed Cookie Protection

In this section, two protocols for user-managed cookie security are described in detail, building on the general approaches described in the previous section.

7.1 Cookie Encryption Using Symmetric Cryptography

In this approach, a secure channel is employed to distribute a cookie key. How this secure channel is established is outside the scope of this paper, but it could, for example, be provided using protocols such as TLS and HTTPS.

The main advantage of this method is convenience. It makes use of existing security protocols to distribute the cookie key. However, doing so requires the establishment of a secure channel. Another drawback is that key management is relatively complicated, since there will be a large number of cookie keys for users to manage and store securely.

The protocol is specified in Figure 1. In this figure:

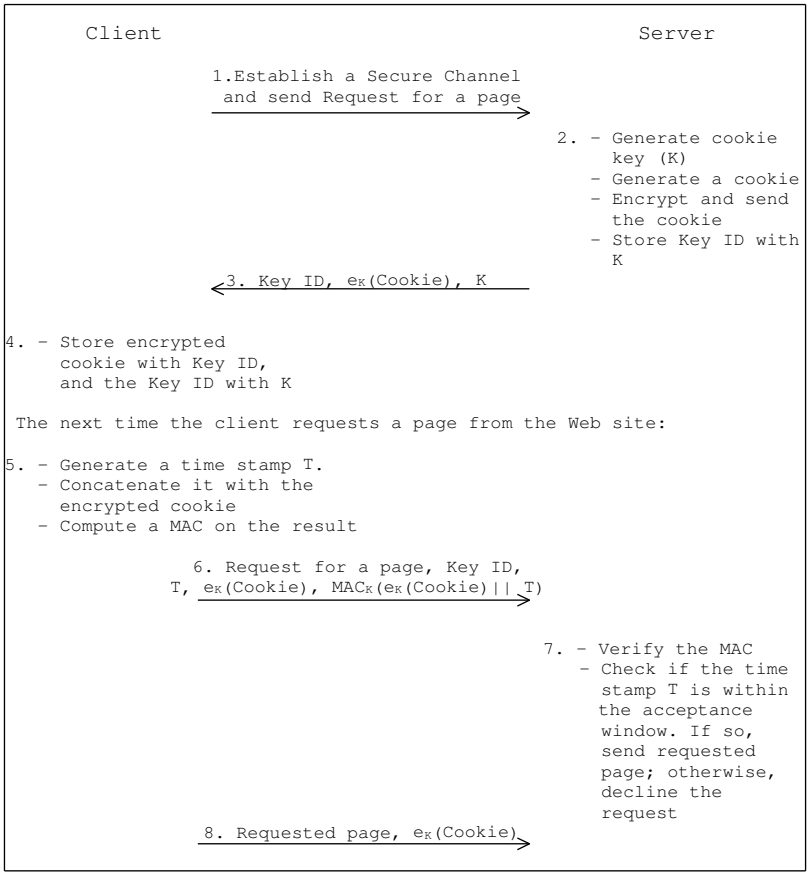

Fig. 1. Cookie encryption using symmetric cryptography.

– 'Client' can represent additional software, a modified Web browser, a plug-in, or an applet which performs security procedures for users,
– 'Server' represents a Web server,
– $X||Y$ denotes the concatenation of data items X and Y,
– K denotes a secret key used to encrypt cookies (the 'cookie key'),
– $e_K(M)$ denotes message M encrypted (using symmetric encryption) with key K,
– 'Key ID' identifies the cookie key K used to encrypt the cookie,
– T denotes a time stamp, and
– $MAC_K(M)$ denotes a MAC on message M using a variant of key K (note that it is important that the key used to compute the MAC is not precisely the same as the key used for encryption, particularly if the MAC is a CBC-MAC based on the same block cipher used to perform cookie encryption).

7.2 Cookie Encryption Using Asymmetric Cryptography

The public key based scheme is presented in Figure 2. In Figure 2, the following notation is employed (in addition to that used in Figure 1):

– $E_P(X)$ denotes data X encrypted (using asymmetric encryption) with public key P,
– $S_C(M)$ denotes the signature of the client on message M (computed using the client private key), and
– P_C denotes the public key of the client.

An advantage of this approach is that cookie keys are stored encrypted. The only key a user has to keep secret is the private key with which cookie keys are decrypted. Therefore, the key management task is not so complicated. There is also less difficulty in key distribution than in the system based on symmetric cryptography.

In order to allow servers to detect an attack where a malicious user deletes the request for cookie encryption (the user certificate), a signature is required on the request message indicating whether cookie encryption is enabled. Otherwise, an attacker can just delete the certificate. If an attack is detected, the server can send a message to inform the user and ask if the user wants to try again. This, however, introduces a risk of denial of service where a malicious user keeps modifying the message causing the page request to fail. Moreover, the additional computation of signing process can increase the user machine's workload and possibly slow the page request process.

8 Secure Cookies Versus User-Managed Cookie Security

In this section, a detailed comparison between the server-controlled and user-controlled approaches is provided.

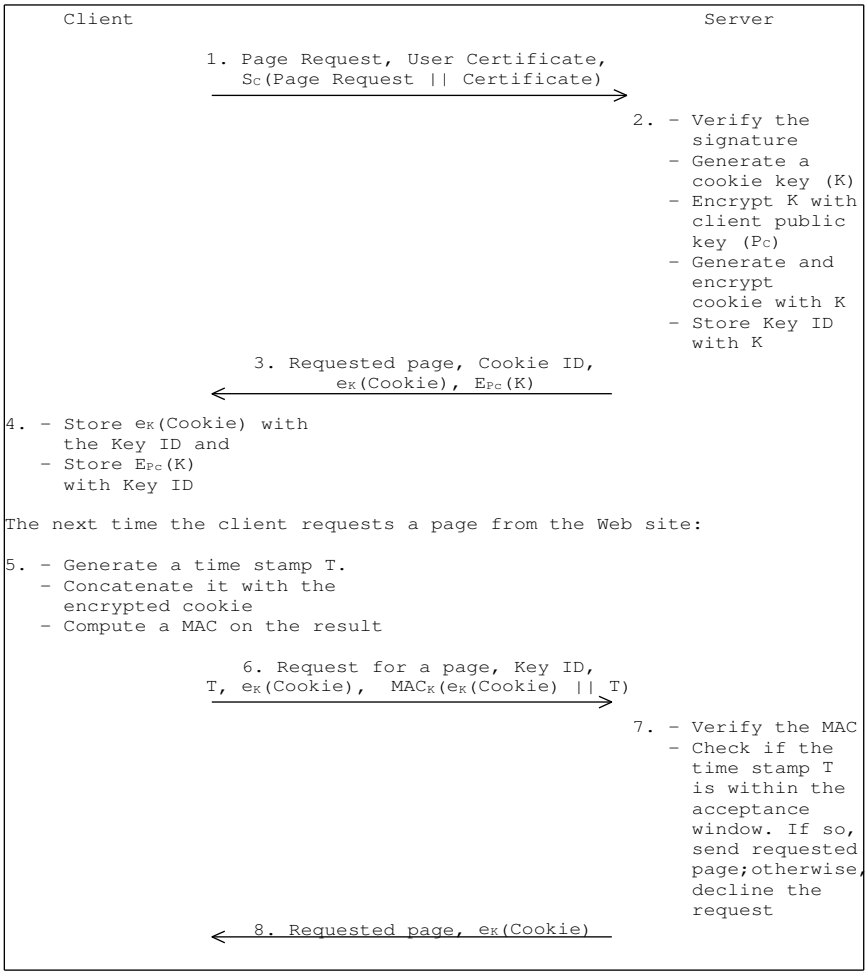

Fig. 2. Cookie encryption using asymmetric cryptography.

8.1 User Authentication

The Secure Cookies method [9] offers three choices for user authentication namely by IP address, by signature, and by password. The first alternative is not always appropriate since in some environments IP addresses are assigned dynamically. Moreover, it is prone to an IP spoofing attack.

The second user authentication mechanism is signature-based, and requires users to have a public key pair in order to sign a time stamp in a cookie. However, the signing process can increase the user workload, and hence possibly slow the web-browsing process.

Both Secure cookies and user-managed cookies employ user passwords. However, in the Secure Cookies technique the password must be sent via a secure channel to a Web server for verification. Therefore, in addition to the cookie security process, a secure channel may be needed. In the user-managed methods, on the other hand, the password is verified locally, which lessens its exposure and avoids the need for secure channel establishment.

8.2 Integrity and Confidentiality

The Secure Cookies scheme uses a signed message digest of cookies and MACs. The user-managed cookie security schemes also use either a signature or a MAC. In both cases, an additional computation for encryption process is required.

Both techniques offer a choice between symmetric and asymmetric encryption to provide cookie content confidentiality. However, one disadvantage of the user-managed scheme is that the required cryptographic keys will have to be stored by the clients.

8.3 Cookie Authentication

In the Secure Cookies approach, there is no mechanism for replay protection. The user-managed cookie security schemes, however, use a time stamp to provide protection against replay attacks. However, a MAC must be included in order to prevent a malicious user from replaying a stolen cookie with a new time stamp, potentially increasing the user's workload.

8.4 Other Issues

The main advantage of the user-managed approach over Secure Cookies is probably the fact that it allows users to have more control over the security of their own cookies. By contrast, with Secure Cookies, Web servers have full control over cookie encryption. Although this may be appropriate for many applications, in some environments users may wish to have more control over their security — indeed, if users wish to read their own cookies, e.g. to deal with the threat of user tracking, then a user-controlled approach is essential. Moreover, the Secure Cookies method may be more complex since a set of cookies is required for each Web site. There is also a need for an Issuing Cookie server to generate the Secure Cookies.

9 Summary and Conclusions

Although cookies are a very useful mechanism for maintaining session state, as discussed earlier they can pose a number of security threats. As a result, three cookie security requirements, namely confidentiality, integrity, and authentication, can be identified.

Secure Cookies [9] is a server-controlled approach for cookie protection that satisfies some of these security requirements. However, in some environments users may want to control their own cookie security, especially if the user tracking threat is to be effectively combated. In this paper, two user-controlled approaches, using symmetric and asymmetric encryption of cookies, are proposed. The main differences between how security services are provided in the server-managed 'Secure Cookies' and the user-managed approaches are summarised in Table 2.

Table 2. Comparisons

Security Requirements	**User-Managed Cookie Encryption**	**Server-Managed Secure Cookies**
User Authentication	Password-based	IP Address, Password, Signature
Integrity	Signature, MAC	Signed hash, MAC
Confidentiality	Encryption	Encryption
Replay Protection	Time Stamp	N/A

The 'Secure Cookies' approach is potentially more flexible since it offers various options to provide each security requirement. It can also be made transparent to the user's Web browser. However, it is relatively complex because it requires an additional server to generate Secure Cookies. Moreover, in most cases, it requires a number of cookies, while in the user-managed approaches a variety of information can be stored in a single cookie. Finally, the Secure Cookies scheme has the potentially major disadvantage that it prevents users examining the contents of cookies stored in their own machine.

If a password scheme is used, the Secure Cookies method requires it to be sent via a secure channel. On the other hand, the user-managed techniques verify a password locally, and hence minimise its exposure. The user-managed approaches also provide additional protection against replay, while the Secure Cookies scheme does not.

References

1. Microsoft Passport scheme. Available at http://www.passport.com/.
2. S. Garfinkel, and G. Spafford. *Web Security & Commerce.* O'Reilly, 1997.
3. B. Hancock. Security views: some cookies are not tasty. *Computers & Security*, **17**(5):374–376, 1998.

4. B. Haselton and J. McCarthy. Internet Explorer open cookie jar. http://www.peacefire.org/security/iecookies/, May 2000.
5. V. Khu-smith. An implementation flaw concerning Netscape Navigator and cookies. January 2001.
6. D. Kristol and L. Montulli. *HTTP State Management Mechanism — RFC2109*. IETF, 1997.
7. S. Laurent. *Cookies*. McGraw Hill, 1998.
8. Netscape. *Persistent Client State HTTP Cookies*, 1996.
9. J. Park and R. Sandhu. Secure cookies on the web, *IEEE Internet Computing*, **4**(4):36–44, 2000.
10. D. Ross, I. Brugiolo, J. Coates, and M. Roe. Cross-site scripting overview. http://www.microsoft.com/technet/security/, Febuary 2000.
11. D. Stein. *Web Security*. Addison Wesley, 1998.