A Plugin-based Privacy Scheme for World-wide Web File Distribution

Michael Jenkin and Patrick Dymond
Department of Computer Science,
York University, CCB 126
4700 Keele St., North York, Ontario, Canada
(jenkin,dymond)@cs.yorku.ca

Abstract

Existing security mechanisms for serving documents on the World Wide Web typically require use of either an underlying security transport mechanisms (e.g., SSL) or alternate servers, browsers and data streams (e.g., SHTTP). In this paper we introduce a simpler method using plugins which provides moderate security for serving private documents within the standard HTTP mechanism and socket layer. This new method operates by providing a security plugin within a standard web-browser environment. It provides a somewhat lower level of functionality and security than the alternatives mentioned above but requires much less overhead, especially on the server end, and appears to be very appropriate for serving low-security, non-public documents, files and images over the world wide web. The method can be easily adapted to provide other advantages, such as automatic "water-marking" of decoded material with the name of the decoder, and the deployment of content-specific compression algorithms.

1 Introduction

Ease of use, widespread availability and cost-effectiveness of the world wide web make it an ideal mechanism for providing publicly viewable documents around the world. At modest cost, an information provider's web pages can be installed at an Internet Service Provider's (ISP) facility and anyone in the connected world who is interested in the material can then beat a path to the provider's door. However, existing web mechanisms are less well-suited for providing private documents to a limited set of authorized users, where some degree of confidentiality or privacy is required. In many cases, especially for a group of mobile users, the web is the most convenient access mechanism, and a simple privacy scheme with low-level security would be very useful in such cases.

The need for secure communication over the internet for electronic commerce [4, 12], and for the private communication of sensitive material is well known (see [5, 3, 6] for example). For email transmitted over the internet, many security features have been proposed including the use of S/MIME [11] to encrypt email attachments, as well as the use of public key cryptosystems (especially PGP) to encrypt the contents of the mail message itself[6].

In terms of HTML, a number of security standards have been proposed to augment existing web technology with (degrees of) security[1]. Two primary standards include SSL (Secure Sockets Layer) proposed by Netscape [9] and SHTTP (Secure HTTP)[10]. These standards are designed to provide significant levels of security for internet- and web-based communication. For example, SSL defines an alternative to the standard underlying communication layer (known as sockets). The entire underlying communication stream is encrypted and complex negotiation strategies are used to ensure the authentication of both sender and receiver as well as determining the appropriate encryption algorithms to use. SHTTP on the other hand, provides a similar level of functionality but performs the task at the level of the (S)HTTP data stream. In effect, SHTTP defines an alternative to the HTTP data stream which provides the same service as HTTP but also provides encryption and authentication services.

For the most sensitive of transactions on the internet and the web, such as encoding financial information, this level of security is essential. For private information requiring only a lower degree of security, these systems are not without costs.

- SSL and SHTTP require a significant amount of computation to encode/decode messages on the fly and this may be difficult to provide in a timely and cost-effective way.

[1] Of course no mechanism can provide complete security, rather different mechanisms each provide different features and degrees of protection.

- SSL and SHTTP require a web server who is willing to provide the service. (The web server must be either SSL or SHTTP enabled.) This often requires that an information provider wishing to provide secure data must manage their own web server.

- The information to be provided is stored in clear text on the web server before being encoded. If the information provider does not own the web site this implies that the data is unsecured against viewing by the ISP operating the site.

- SHTTP in common practice and SSL in general both protect the entire data stream through encryption. In many applications not all of the data on a site needs to be protected to the same level of security.

- SSL cannot provide water-marking services to the data being served as it operates at the socket level. SHTTP would have to be modified significantly in order to provide water-marking services.

- Compression algorithms typically perform poorly on encrypted data[1]. Thus sytems such as SSL and SHTTP endure performance degradation during transmission as well as the costs associated with the encryption/decryption process. As not all of the datastream may need to be protected to the same level, an encryption mechanism which encrypts only that which needs to be kept private offers the potential for performance improvement both in the encryption/decryption process as well as at the data transfer level.

A fundamental requirement of SSL and SHTTP is control of the web server. Unfortunately this is not an option for many information providers on the Web. The vast majority of commercial web ISP's serve pages for more than one company or information provider and provide only limited 'non-standard' features.

Companies using an ISP provide information contained within documents and the ISP "serves" the documents within the standard HTTP. If local customizations are possible, they are usually limited to simple CGI script applications. This makes it difficult for companies to serve documents via the web to only "authorized" users.

This common practice of providing only unmodified HTTP also exists in the academic and government environments. University web pages often include "private" data which is protected by serving the pages through NFS rather than HTTP. This limits the pages to machines which mount the appropriate NFS pages – unfortunately this often excludes web browsers running on non-Unix platforms which may not be running NFS. Another common protection mechanism is use of a password to limit access to a set of pages. This has the disadvantage that cleartext information is transmitted over the net once the password has been used, as well as some of the disadvantages discussed above regarding ISP access to data. Finally, in some cases information may be stored in a standard encoded form at the ISP and provided to users in this encoded form over the net. An example would be information stored as a password-encrypted file. Users then must store, decode and examine the information off-line (i.e. outside their browser.)

Many different software packages support this type of mechanism. For example, Stuffit and ZIP archives can be encrypted by the sender and then transmitted to the recepient for later decryption. Such systems typically require the recepient to interact with the transmission software as well as to manually execute the archive software in order to decrypt the encrypted archive.

Although the Web has found considerable application in the dissemination of information throughout the world, a number of difficulties have emerged in terms of the relationship between the ISP, the information-provider, and the user of the web. These include,

- ISP's often prefer to view themselves as "common carriers", without specific involvement with the data of individual companies or other information-providers using their facilities. However this position may be viewed as inconsistent with the ISP's normal ability to directly view and control access to the information-provider's data. Any mechanism which restricts the ISP's access to the data being provided, while still permitting the data to be served, helps to strengthen the "common carrier" model of the ISP.

- In terms of the commercial information-provider, it is often the case that the information being provided is the commodity for which a price is being charged. Given the ease to which computers can be used to make copies of information and the ability of the web to then communicate pirated copies of the information throughout the world, mechanisms to aid in the control of commercial data are essential. Techniques such as water-marking can be used to uniquely associate the data with a particular purchaser in order to (i) reduce the incidence of inappropriate use of data and (ii) to assist in prosecution should theft of data take place.

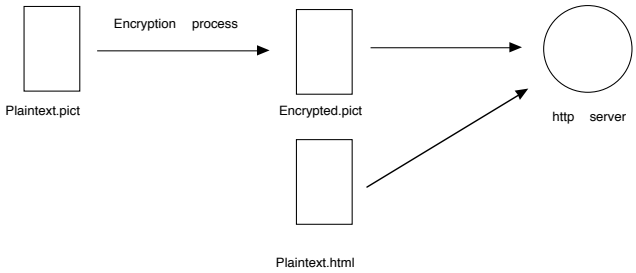
In summary, for private transactions requiring only

modest security SSL, SHTTP and other mechanisms may be unnecessarily complex. They are designed to provide highly secure communication but there are non-trivial costs involved. The alternative presented here provides a lower level of security but without the cost and overhead associated with existing mechanisms. We do not present any encryption scheme in this paper, but rather we demonstrate a method for serving information using plugins which may make use of any publicly available encryption scheme. The technique which is presented here also makes some progress in terms of strengthening the "common carrier" model of the ISP and in uniquely identifying the user of the data as it is accessed.

The specific issue addressed in this paper is the problem of an *information provider* wanting to serve *secrets* embedded within regular web pages to *authorized users*. The pages are to be served in such a way that the secret information can be readily obtained only by the authorized users.

(For illustrative purposes, we posit that the information provider's web pages which are to be served are located at an ISP, that the information provider is a customer of the ISP, and that the secret information is stored as an embedded region within regular web pages.)

In order to provide pages containing secrets through the standard HTTP/socket interface to be readily accessible by authorized users while simultaneously providing some degree of privacy of the secrets with respect to access by non-authorized users, the secrets can be pre-encrypted by the information provider and then served by the ISP using the standard HTTP mechanism. A mechanism to decrypt the secrets in a transparent manner is then needed for the authorized users. In this paper we describe one such system based on the design of a plugin and some variations on it. This is the first example known to us of using standard plugin technology (which works with many web browsers such as Netscape [7] and can be readily programmed using Netscape's Plugin API [8]) to provide transparent access to encrypted files within browsers. The method uses any standard legal encryption/decryption mechanism to serve encrypted documents from within a standard, unmodified HTTP server, and then to have the documents decrypted transparently at the client end by the browser plugin. Plugins can be tailored for each specific application to provide additional functionality such as content-based compression and decompression, water-marking with the user's name and time-limited password access. A prototype system has been written and is available on the web.



The information provider generates the site with both plaintext html documents as well as encrypted files. As this encryption can be performed off-site, the ISP need not have access to the plaintext version of the encrypted material being served.

Figure 1: Site setup

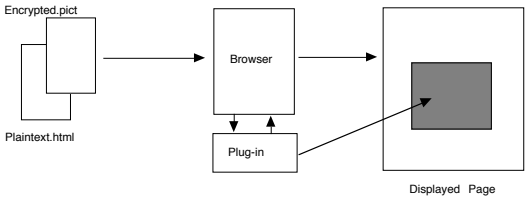
2 Encryption via a global key

As discussed above, our method is intended to solve the problem of serving secret information embedded within regular web pages in such a way that it can be readily obtained only by authorized users.

The process of serving pages with encrypted embedded regions involves two main phases; (i) the setup of the site and distribution of keys and plugins to authorized users, and (ii) serving the encrypted pages to the authorized user community.

2.1 Site generation and key distribution

The information provider (as opposed to the ISP) generates the content that makes up the web site. This site includes both plaintext and encrypted documents (Figure 1). The encryption process uses a standard encryption algorithm. One possible implementation would use a public domain version of the Unix `crypt` command, but other alternatives are possible. This encryption process can take place anywhere, and in particular it can take place on a different machine than the HTTP server machine. In this eventuality plaintext versions of the encrypted files never exist on the HTTP machine and thus the material is hidden from the server's machine. In the most straightforward implementation all of the encrypted files on the site are encrypted with the same password but other strategies are possible. Note that the encryption process takes place only once, not once per transaction which is the case for both the SSL and SHTTP approaches and that only the sensitive material is encrypted, rather than encrypting the entire HTTP or TCP/IP stream.



Encoded documents from the server are displayed in Netscape with the plugin decoding and displaying the encrypted document components.

Figure 2: Displaying a page

In the basic implementation described here, the information to be served consists of private *images* (say Macintosh PICT imagery) contained on otherwise plaintext pages. The images are stored on the server only as encrypted *crypto-pict* files (where *crypto-pict* is a new data-type). In the sample implementation *crypto-pict* files are identified by the MIME type *epict*.

The information provider then distributes usernames and passwords to the trusted user community. Different usernames are assigned to each user, and the password string encodes three pieces of information; the global decryption key to decrypt the encrypted pages, a verifier for the user name, and a time-code verifier if the passwords are only to be valid for a restricted period of time.

We do not concern ourselves with the task of distributing the passwords, but secure mechanisms such as SSL/SHTTP could be used to distribute the passwords electronically, or they could be distributed through other mechanisms, such as telephone, mail, or courier for example.

2.2 Serving encrypted pages

In order to receive an encrypted page, a user augments their web browser (the sample implementation uses Netscape Navigator) with a plugin (see Figure 2). This plugin allows for the transparent decryption of the encrypted material provided by the server. The user's browser is configured to recognize this new MIME-type and to assign the interpretation of images of this type to the plugin.

A *crypto-pict* is included in an html document through a standard html definition such as

```
<embed src='moon.epict' user='name'
key='password' type='image/epict'>
```

When this html is encountered by the browser, the decryption plugin is passed the image and the users name and password (specified by the `user` and `key` arguments in the html). One feature of the current implementation is that usernames and keys are *persistent* throughout a single page. The plugin can be designed to retain this information only while on the current page, throughout the current session, or retained from session to session, depending on the application.

Various techniques are available in order to prompt the user for their name and password. Due to the persistent nature of the key and password, it is possible to prompt for these values once per page and then subsequent encrypted pictures will be decrypted with the same password. Another alternative is to use a scripting language such as JavaScript to prompt for the username and password and then to store them as cookies on the clients machine. Yet another alternative is to have the plugin retain the information from one page to another.

If the password is invalid (due either to failing self-consistency tests or to being unable to decode the image into the correct format) the plugin displays a warning message. Otherwise the first portion of the password (the global key) is used to decrypt the document. The plugin utilizes this plus the decryption algorithm for the encryption process used by the information provider. The rest of the user's password encodes a verifier for the users name and an expiration date for the entire password. As the plugin renders the encrypted material, the displayed image is water-marked using the users name. The user name verifier is used to ensure that the appropriate identification information is included in the water-mark. The expiration date mechanism is used to allow time-limited access to the encrypted pages should that be desirable. (Our example implementation does not currently implement the verification or expiration date mechanism and the password used is exactly the decoding key.)

3 Sample session

Figure 3 provides the html document which generates two web pages shown in figure 4. Although the JavaScript that access the cookie structure has been omitted from the listing in Figure 3, the general structure of the document is straightforward. The username and password are extracted from the client cookie file and passed to the plugin. Subsequent decoded images do not require the password/username pair due to the persistent nature of the plugin.

Figure 4 shows the displayed page when the page is

```
<HTML> <HEAD>
   <TITLE>NetFence</TITLE>
</HEAD>
<BODY>
Welcome to the homepage of <b> NetFence </b>.
This page requires that you have installed
the NetFence plugin, which currently is only
available for the Macintosh.
If you have not yet set your password,
then you should do so through
<a href="NetFence2.html"> here </a>.
<script Language="JavaScript">
...
var nv;
var np;
// get the username/password pair
if(GetCookie("netfenceuser") != null){
   nv  = GetCookie("netfenceuser");
   np = GetCookie("netfencepasswd");
   document.write("Your username is set to " + nv
   + " and your password is " + np + "<br>");
   document.write(
"<EMBED
SRC='http://www.cs.yorku.ca/~jenkin/moon.epict' "
+ "WIDTH=100 HEIGHT=100 USER='" + nv +  "' KEY='"
+ np + "' type='image/epict'>");

} else
document.write
("Please set your username and password");
// -->
</script>
<hr>
<h3> Encoded with foo </h3>
<EMBED
SRC="http://www.cs.yorku.ca/~jenkin/light.epict"
WIDTH=150 HEIGHT=150 BORDER=5 type="image/epict">
<EMBED
SRC="http://www.cs.yorku.ca/~jenkin/moon.epict"
WIDTH=150 HEIGHT=150 BORDER=5 type="image/epict">
<hr>
<h3> Encoded with bar </h3>
<EMBED
SRC="http://www.cs.yorku.ca/~jenkin/light2.epict"
WIDTH=150 HEIGHT=150 BORDER=5 type="image/epict">
<EMBED
SRC="http://www.cs.yorku.ca/~jenkin/moon2.epict"
WIDTH=150 HEIGHT=150 BORDER=5 type="image/epict">
<br>
</body>
</html>
```

Figure 3: Portion of the html document that generates the pages shown in Figure 4 .

decoded using either the password 'foo' or 'bar'. The pictures 'light' and 'moon' are encoded with 'foo' while 'light2' and 'moon2' are encoded with 'bar'. The plugin attempts to decrypt all of the images with the decoding key provided and examines the decoded image to see if it is of the correct type. If it is, then the image is displayed, otherwise a warning image is provided. In either case the image is water-marked using the user's identification. Note that the images are trivially water-marked in the sample implementation used here. More sophisticated and less obvious water-marks are possible.

4 Discussion

This paper describes a plugin-based privacy scheme for file distribution on the world-wide web. This mechanism provides for the delivery of encrypted data within a web page and for decryption of this information within the user's browser. In this simple scheme there is only one global key known to all authorized users. The server stores the private data only in encoded form, identifying them using a new mime-type. As the data is decoded it is watermarked to uniquely associate the decoded data with the user.

In the implementation described here, data stored on the ISP is encoded with a single global encoding key. To change the encryption, the data must be re-encrypted with the new key and re-stored on the server, replacing data encoded with previous keys. Individual user keys incorporate the global encryption key, as well as as a verification password for the user name and an optional expiration date. As the encryption process only takes place once, and as no extra encryption takes place on the data stream between the ISP and the client, the mechanism presented here is significantly less expensive than approaches such as SSL and SHTTP.

The software and hardware is standard, both at the service provider ISP end and at the browser (user) end. Authorized users must have the plugin and must know the current password to decrypt and display the images. No plaintext secrets are ever sent across the net nor are they stored at the ISP.

It is possible to adapt the plugin-based privacy scheme described here in several ways. It is not necessary to base the mime-type on embedded imagery. The material could be text, and could consist of entire pages. Even the entire html page itself can be encrypted and sent in this way. It would also be possible to implement the decoder in Java rather than as a plug-in.

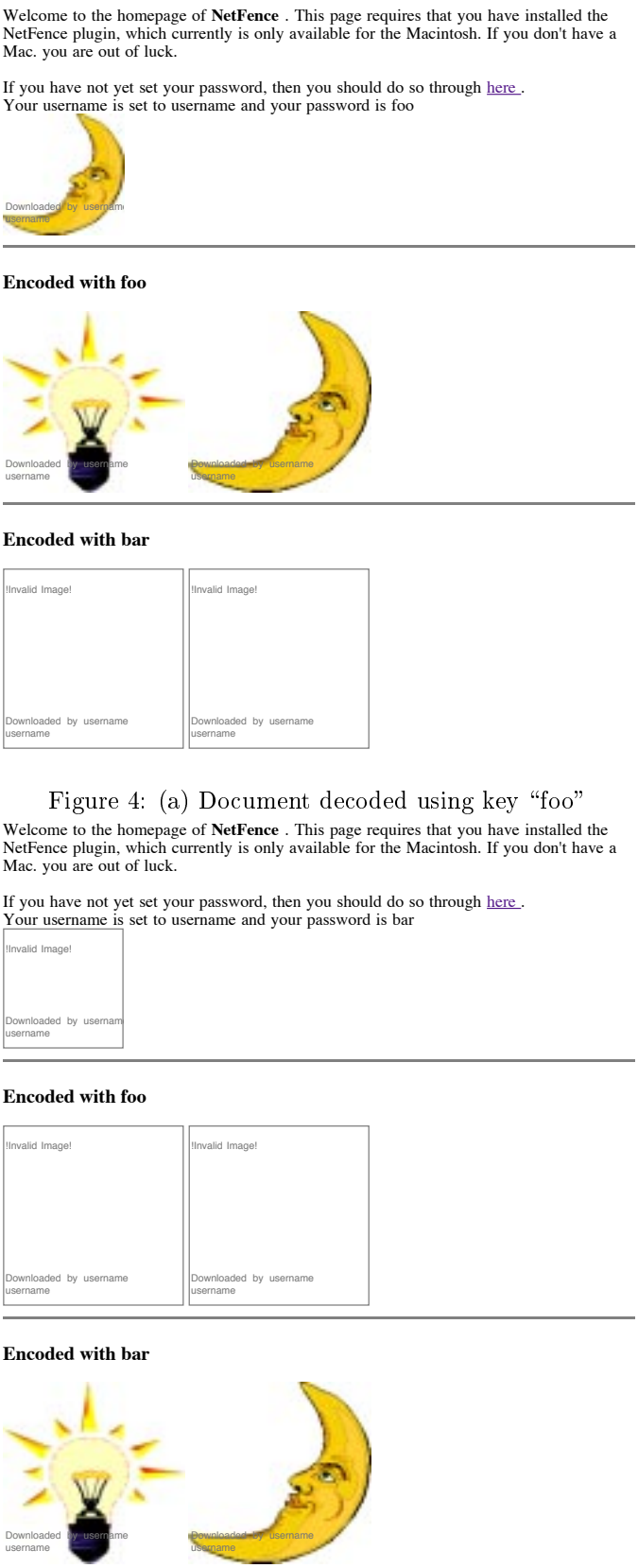


Figure 4: (a) Document decoded using key "foo"

Figure 4: (b) Document decoded using key "bar"

One very significant change would be to encode the images on-the-fly at the time the ISP serves them. Although this would have some disadvantages as discussed above, the primary advantage would be the possibility of using a distinct encryption key for each user. This requires modifications to the ISP, so that in serving an embedded region of type *crypto-pict* to a given user, the server encodes the embedded region using an individual key as the document is served. (They could still be pre-encoded with a universal encoding which prevents the service provider from accessing them.) This computationally more expensive approach would permit expiring access privileges of some but not all users, providing a hierarchy of private data based on user id, etc. This individual key-based scheme is currently under development [2].

Sample implementation

A sample implementation of NetFence for the Macintosh is currently available from http://www.cs.yorku.ca/~jenkin/NetFence.html. This page contains links to the plugin (for Netscape) and some sample documents. As currently implemented, NetFence only supports Macintosh "PICT" images.

Acknowledgments

We thank the referees and our colleagues N. Wilson and M. Robinson for helpful comments. Research supported by the Natural Sciences and Engineering Research Council.

References

[1] R. J. Atkinson. Towards a more secure internet. *IEEEE Computer*, 30(1):57–61, 1997.

[2] P. W. Dymond and M. Jenkin. Plug-in based web privacy method with individual user keys. York University Computer Science Technical Report (in preparation), 1998.

[3] C. S. Guynes, R. G. Vedder, and M. T. Vanecek. Security issues on the internet. *ACM Security Audit and Control Review*, 15(2):8–12, 1997.

[4] S. Hamilton. E-commerce for the 21st century. *IEEE Computer*, 30(5):44–47, 1997.

[5] D. J. Icove. Collaring the cybercrook: an investigator's view. *IEEE Spectrum*, 84(6):31–36, 1997.

[6] P. Merenbloom. Securing network communication is pivotal for growing remote technology. *Infoworld*, page 54, 1995.

[7] Netscape Inc. Netscape navigator handbook. http://home.netscape.com /comprod/products /navigator/version_3.0/index.html, 1996.

[8] Netscape Inc. Netscape navigator plugin guide. http://developer.netscape.com /library /documentation/communicator/plugin/index.html, 1996.

[9] Netscape Inc. Secure sockets layer. http://www.netscape.com /assist/security /index.html, 1996.

[10] E. Rescorla and A. Schiffman. The secure hypertext transfer protocol. Internet-draft draft-ietf-wts-shttp.01.txt, 1996.

[11] J. Schwartz. Group agrees on method to secure email. *Communications Week*, pages 1,88, 1995.

[12] M. Sirbu. Credits and debits on the internet. *IEEE Spectrum*, 34(2):23–29, 1997.