A Practical Customer Privacy Protection on Shared Servers

Pramote Kuacharoen
School of Applied Statistics
National Institute of Development Administration
118 Serithai Rd. Bangkapi, Bangkok 10240 Thailand
pramote@as.nida.ac.th

***Abstract*—Customer privacy protection is very important in an e-commerce site. Without such protection, the customers may not be willing to provide personal information or conduct business at the site. Therefore, the merchant should protect its customer information in order to gain customer trust. In this paper, an approach for protecting customer information on shared servers is presented. The proposed method provides customer privacy by encrypting the customer information at the client machine and allows the key to be shared between the customer and the merchant. The encrypted customer information is sent and stored on the shared server. Therefore, attackers cannot access the customer information while the merchant can easily obtain such information. To make it practical, the technique is implemented in software and is transparent to the customer. Furthermore, this solution to protect customer privacy does not significantly increase the cost to develop and deploy the system.**

Keywords-privacy; security

I. INTRODUCTION

A shared web hosting service or a web hosting service allows many websites to reside on one web server connected to the Internet. Sharing a web server has an economical advantage; the cost for maintenance is shared among subscribers. It is an alternative to a dedicated server where the server is not shared and can be located on the premises or at a hosting company which provides colocation hosting service. However, a shared server also has a disadvantage that information security is more difficult to achieve. As a result, the customer privacy may be compromised.

On a shared server, a common pool of server resources such as RAM, CPU, and server programs is shared by many websites. Server programs such as the web server and the database server can also be shared among the users. The shared host may run only a single instance of each program. Although the user has private storage space and the user rights are provided by the server's operating system, a rogue user has a greater chance to compromise the server than an attacker who does not have access to the server. The user may also be a database user. The database administrator should not allow users to be able to access another user's data. Another security concern is a rogue administrator of the web hosting company. The system administrator has a superuser or root privilege to view files in the system. Usually, the username and password which are used to access the database are in the configuration file or hardcoded in the application as illustrated in Fig. 1. This information can be easily viewed by the system administrator or an attacker who has gained access to the web server. Using the database username and password, the data in the database can be obtained.

```
<?php
$link = mysql_connect('localhost', 'mysql_user', 'mysql_password');
if (!$link) {
    die('Could not connect: ' . mysql_error());
}
echo 'Connected successfully';
// ...
?>
```

Figure 1. PHP code to open a connection to a MySQL server.

After the database has been compromised, an attack can obtain the customer's personal information such as name, Social Security number, or credit card number. The attacker can perform identity theft by using the stolen information to commit fraud or other crimes. Identity theft is a serious problem. It is prevalent in many countries around the world affecting economies. In order to fight against identity theft, consumer privacy must be protected.

In this paper, a practical customer privacy protection on a shared server is presented. The objective of the proposed approach is to protect the privacy of a customer who provides personal information to or makes a purchase from a website which is hosted on a shared server. To make it practical, the technique must be a software solution and must be transparent to the customer. Another important requirement is that the solution must not significantly increase the cost to develop and deploy the system.

This paper consists of five sections. The next section, Section II, describes background and related work in the area of privacy, anonymity, and vulnerability. Section III presents a security design which provides a privacy protection. Section IV describes the implementation of a website which ensures the customer privacy protection. Finally, Section V concludes the paper.

II. BACKGROUND AND RELATED WORK

The right to privacy is a part of many countries' privacy laws. In US, the right to privacy has been amended in the Bill of Rights. The California Security Breach Notification Law forces government agencies, companies, and nonprofit organizations conducting business in California to notify California customers when personally identifiable information has been compromised [1]. In Europe, the right to privacy is a highly developed area of law. All member states of the European Union (EU) must abide by this law. It enforces the protection of individuals with regard to the processing of personal data and on the free movement of such data. This protection covers the Internet privacy. In accordance with the principle of freedom of expression and the right to privacy, use of anonymity is legal. Users should be able to access data and browse anonymously so that their personal information cannot be recorded and used without their permission.

To ensure a high level of privacy, Internet anonymity can be employed so that when an individual communicates through the Internet, any third parties will not be able to link the Internet activities to personally identifiable information of the Internet user. The privacy-enhancing technology exists; however, it is not widely adopted. For instance, Onion routing provides real-time bidirectional anonymous connections that are resistant to both eavesdropping and traffic analysis [2]. Onion routing consists of a fixed infrastructure of onion routers. Even though some routers in the onion routing network may have been compromised, the connection between the sender and the receiver remains anonymous. The anonymity is accomplished by implementing the onion routing protocol at the application layer. When a sender wants to sends a message to a receiver, the sender chooses a random sequence of routers. The sender constructs the message and routing information as a payload. Routing information for each link is encrypted with the router's public key. Therefore, each router learns only the identity of the next router. For example, when a sender A wants to send a message M to a receiver B, the sender A selects onion routers R_1, R_2, R_3, and R_4 as shown in Fig. 2. Router R_3 is compromised.

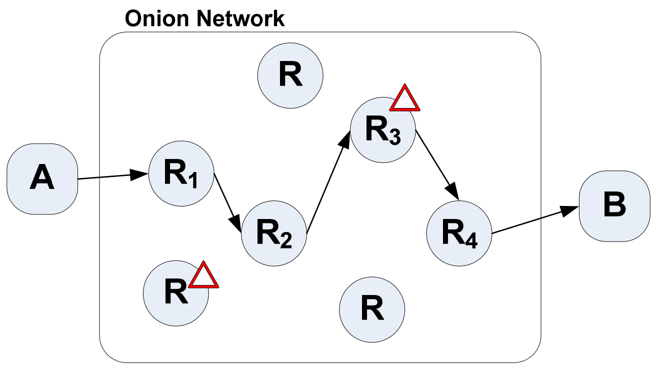

Figure 2. Sending a message through the onion network.

The sender A constructs the payload:

$$C = E(PU_{R1}, K_1|R_2)|E(K_1, E(PU_{R2}, K_2|R_3)|E(K_2, E(PU_{R3}, K_3|R_4)|E(K_3, E(PU_{R4}, K_4|B)|E(K_4, E(PU_b, M)))))$$

When the message is received by an onion router, it decrypts the message to obtain the session key and the next destination. The onion router uses the session key to decrypt the rest of the message and sends it to the next destination. In this example, R_2 sends R_3 the message $E(PU_{R3}, K_3|R_4)|E(K_3, E(PU_{R4}, K_4|B)|E(K_4, E(PU_b, M)))$. R_3 only knows that it receives the data from R_2 and will forward the message $E(PU_{R4}, K_4|B)|E(K_4, E(PU_b, M))$ to R_4. If R_3 is controlled by an attacker, the attacker can only learn limited information.

Tor is a second-generation Onion routing network. It is designed for low-latency anonymous Internet communications. It also addresses limitations in the original design by adding perfect forward secrecy, congestion control, directory servers, integrity checking, configurable exit policies, and a practical design for location-hidden services via rendezvous points [3]. For practical purposes, such a high level of privacy may not be necessary. A level of privacy through controlled disclosure of personal information may be adequate. The revelation of IP addresses, non-personally-identifiable profiling, and the like might be acceptable trade-offs for the convenience that the user could otherwise lose using a high level of privacy.

A dynamic web page usually loads content from a database for each individual viewing. It provides the user with the ability to personalize the web page. The website takes parameters from the web user and makes SQL statement to perform queries or update the information to the database. Since the input comes from the user, the attacker may be able to exploit a security vulnerability by sending the input crafted such a way that when it cause the database to process invalid data. This exploitation technique is called SQL injection. Using SQL injection, the attacker may be able to obtain other user information. Consider the sign-in web page where the user supplies username and password as "abcd" and "1234," respectively. After the user clicks the submit button, the following SQL query is generated:

```
SELECT * FROM user_table WHERE user_id = 'abcd'
AND password = '1234'
```

If the user enters "' OR 1=1--" as the password, the SQL query will become:

```
SELECT * FROM user_table where user_id = 'abcd' AND
password = '' OR 1=1--'
```

The generated SQL query when executed will return information of the username "abcd" including the password. The defense against this kind of attack is to validate the user input to ensure that it will not cause an unintended SQL statement and carefully review code that executes constructed SQL commands [4][5]. SQL Injection vulnerability detections have been proposed [6][7]. However, the threats still exists. To further enhance security, the user data should be stored as encrypted. Without the knowledge of the key, the attacker will have difficult time deciphering the information in the case of the SQL injection attack succeeds. Hence, the customer privacy is protected.

III. Security Design

In this section, the security design to protect customer privacy will be discussed. The design includes storing customer password, authenticate the customer, encrypting and decrypting customer information, and handling transactions.

A. *Storing Customer Password*

In a shared web hosting service, web applications authenticate a customer by comparing the username and password combination that the customer entered to the value in the database table containing customer information. Storing actual customer passwords inherits a prominent risk. It might be possible for an attacker to be able to inject SQL queries to the application to list the usernames and passwords or to obtain them directly from the database server. If an attacker gains access to the password storage, all passwords will be compromised. Therefore, the customer passwords should not be stored as clear text. Instead, a hashed form of the passwords should be stored.

When a customer registers to the website, the customer chooses a username and a password. If the username is unique, the application chooses a random number which is called salt [8]. The application then sends username, salt, and the hash of the combination of the salt and the password to the server. Since the server does not store the actual passwords as clear text in the database, the usernames and the passwords cannot be revealed if the database is compromised or the SQL injection is accomplished. With the use of salt, the attacker cannot efficiently attempt to crack the password database. The attacker has to search for each and every user's password which can potentially be an expensive operation. This might dissuade the attacker from making the attempt to perform a brute force attack.

B. *Authenticating the Customer*

When the customer logs in, a username and a password are entered. The username is sent to the server. If the username exists, the corresponding salt and a nonce are sent back to the browser. If the customer does not exist, the server sends a random salt and a nonce. The system will not reveal whether or not a customer exists. At the client machine, the hash of the combination of the salt and the password is generated. Then, the resulting hash value is concatenated with the nonce. The response is generated by hashing the previous value. The response is sent to the server. The server computes the hash value of the salted password and the nonce. The server verifies if the calculated value matches the received value. The customer is authenticated upon a successful verification. The challenge-response authentication protocol is illustrated in Fig. 3.

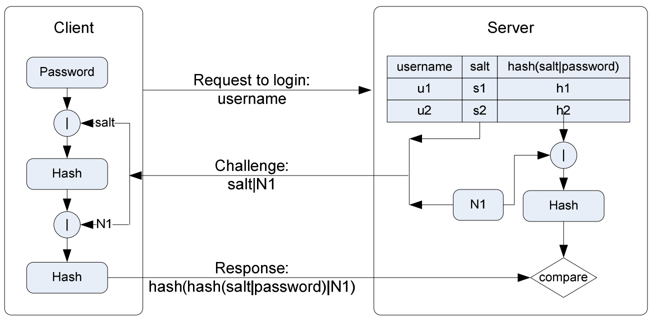

Figure 3. Challenge-response authentication protocol.

C. *Encrypting Customer Information*

The web applications may store the customer information such as name, address, preferences, and credit card information. To provide security and privacy, the customer information must be encrypted. However, this information should be easily revealed to the customer and the merchant. Certain information can be used as the personalized greeting upon the user logins while certain information such as name and shipping address must be known to the merchant in order to provide services.

The secret key is randomly generated for each customer. This secret key is used to encrypt and decrypt the user information. Therefore, the secret key must be known to both customer and merchant. However, the secret key must be protected. Since the customer and the merchant do not share any secret, the secret key must be separately encrypted for the customer and merchant. For the customer, the customer's secret key is derived from the customer password using password-based cryptography standard [9]. This provides convenience for the customer since the customer has to enter a password when logging into the website. The customer does not need to remember the customer's secret key. The merchant must also be able to obtain the secret key. This can be accomplished by encrypting the secret key using the merchant's public key as shown in Fig. 4. The merchant can obtain the secret key by decrypting it using the corresponding private key.

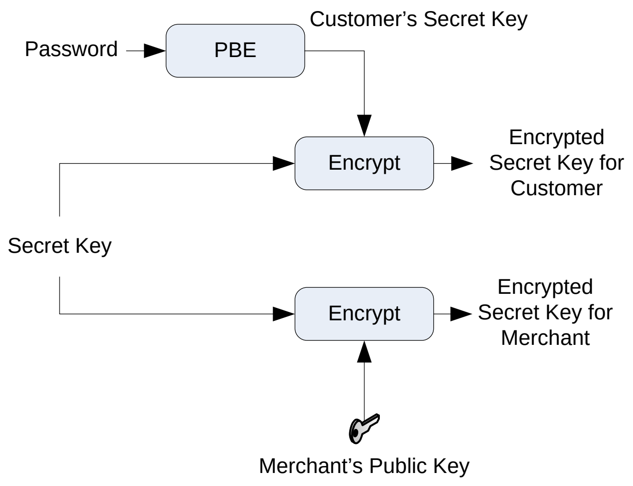

Figure 4. Generating the encrypted secret keys for the user and merchant.

When the customer enters the information, the customer's secret key is generated using the customer's password. The message digest of the information is created using a cryptographic hash algorithm. If the information is modified, the message digest is most likely to be different. The integrity of the information can be preserved if the message digest is stored securely. The secret key is used to encrypt the customer information and its message digest using a symmetric cryptographic algorithm such as AES [10] at the client machine. The encrypted customer information is sent to the server along with the encrypted secret keys. Since the server stores the encrypted customer information, this information is protected.

D. *Decrypting Customer Information*

After the customer has been authenticated, the personal information is decrypted at the client web browser. Web pages can be personalized based on the user profile. For example, a personalized greeting can be shown upon the successful login. The encrypted customer information together with the customer's encrypted secret key are sent from the server. At the client, a customer's secret key is generated using the user password. The customer's secret key is then used to decrypt the secret key. After obtaining the secret key, it is then used to decrypt the received data to obtain the customer information. Fig. 5 shows the process of decrypting customer information at the client machine.

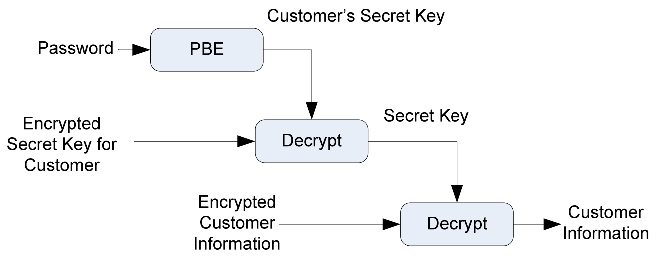

Figure 5. Decrypting customer information at the client machine.

The process of decrypting the customer information for the merchant is slightly different. As illustrated in Fig. 6, the merchant can use the private key to decrypt the encrypted secret key and use the secret key to decrypt the customer information.

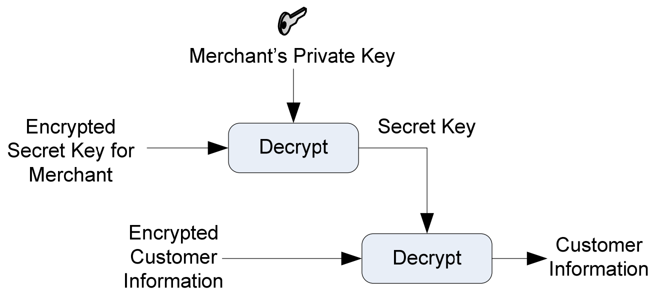

Figure 6. Decrypting customer information at the merchant's machine.

E. *Handling Transactions*

In a typical e-commerce site, the customer browses the site for goods and services, and makes a purchase. The order information is sent to the server. The merchant processes the order. To ensure that the order information is confidential, it must be encrypted. In this approach, when a transaction is made, the transaction information is encrypted using a secret key at the client side. The encrypted transaction information is sent to store on the server. Since the merchant can obtain the secret key, the merchant can reveal the order information by decrypting it with the secret key and process the order accordingly.

IV. Implementation

This section describes the implementation of a website hosted on a shared web server while protecting the customer privacy.

A. *Architecture*

As illustrated in Fig. 7, the architecture consists of the customer machines, the shared web server, and the merchant machine. The customer communicates with the shared web server securely through a HTTPS [11] connection. The transmitted data is protected against eavesdropping during transit. The customer information and transaction data are encrypted at the client machine and stored at the shared web server. Therefore, the customer information is securely stored. The merchant needs to synchronize transactions with the shared web server. The transactions are processed at the merchant's machine.

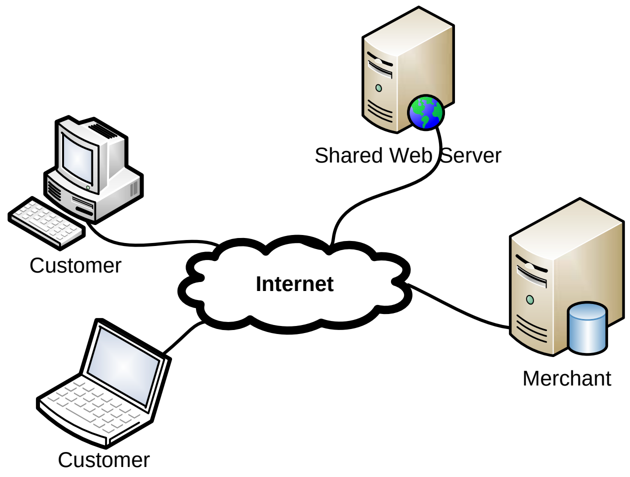

Figure 7. Architecture of the implementation.

B. *Web Application*

To provide privacy and security, the proposal implementation extensively uses Java applet and Java Script. The Java applet provides the functionality for encryption and decryption. JavaScript provides dynamic web presentation. The web page consists of two main components, namely, the crypto applet and an inline frame or iframe. The crypto applet is not visible to the customer. The contents are displayed in the iframe.

Java Cryptography Architecture (JCA) for Java Platform Standard Edition 6 [12] is used to implement the crypto applet. Password-based encryption provides a facility to convert the customer password to a secret key. AES is used for symmetric operations such as encryption and decryption. However, RSA [13] is used for asymmetric operations.

Fig. 8 shows the communication between the crypto applet and JavaScript. When the content in iframe is loaded, the encrypted data is passed to the crypto applet via JavaScript. The JavaScript obtains the decrypted data and modifies html elements accordingly and the content of the page is displayed in the iframe.

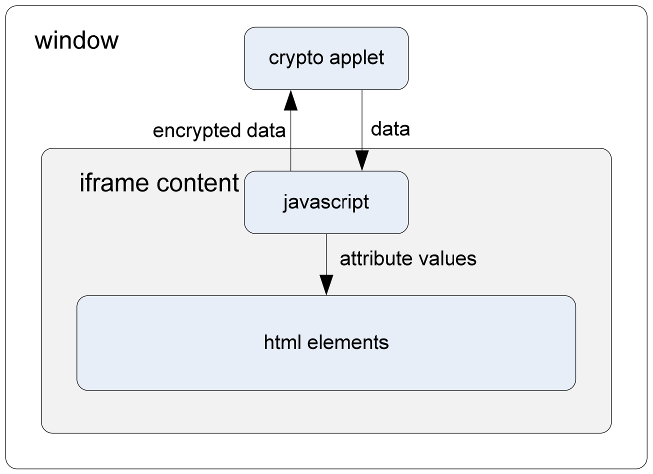

Figure 8. The process of displaying the web page.

By default the size of an iframe does not change after it is created. When the content is larger than the iframe, scrollbars will appear creating multiple scrollbars. To eliminate multiple scrollbars, the iframe may be dynamically resized based on its content size which leaves only the standard browser scrollbars. Fig. 9 shows the JavaScript function which can be used to dynamically resize an iframe by obtaining the size of content.

```
function resizeIframeToFitContent(iframe) {
  iframe.height =
    document.frames[iframe.id].document.body.scrollHeight;
  iframe.width =
    document.frames[iframe.id].document.body.scrollWidth;
}
```

Figure 9. JavaScript for dynamically change the size of an iframe.

When the customer submits information, the information is passed to the crypto applet for encryption and the encrypted information is then sent to the server. This can be accomplished by using onClick or onSubmit events in JavaScript. When the customer clicks a link or a button to submit information, a JavaScript function is called to encrypt the information and send the encrypted information to the server.

V. Conclusion

Customer privacy can be protected so that even the customer information is stored on a shared server. In this paper, a methodology for ensuring the privacy of the customer is presented. This approach provides a practical means and uses existing technologies in such a way that it is transparent to the customer. The customer information is encrypted at the customer's machine and the encrypted customer information is sent to be stored at the shared server. The merchant can synchronize the data with the shared server and decrypt the encrypted customer information at a secure facility. Therefore, customer privacy is guaranteed.

The approach was implemented and verified. The crypto applet was written using Java Platform Standard Edition 6.

References

[1] Security Breach Notice - Civil Code sections 1798.29, 1798.82, and 1798.84, California Office of Privacy Protection.

[2] P. F. Syverson et al., "Anonymous Connections and Onion Routing," in *Proc. 18th Ann. Symp. Security and Privacy, IEEE CS Press*, May 1997, pp. 44-54.

[3] R. Dingledine et al., "Tor: The Second-Generation Onion Router," in *Proc. 13th USENIX Security Symp.*, Aug. 2004, pp. 303-320.

[4] K. Amirtahmasebi et al., "A survey of SQL injection defense mechanisms," *Int. Conf. for Internet Technology and Secured Trans., ICITST 2009*, Nov. 2009, pp.1-8.

[5] R. Ezumalai and G. Aghila, "Combinatorial Approach for Preventing SQL Injection Attacks," *IEEE Int. Advance Computing Conf., 2009. IACC 2009*, Mar. 2009, pp.1212-1217.

[6] M. Junjin, "An Approach for SQL Injection Vulnerability Detection," *6th Int. Conf. Inform. Technology: New Generations, ITNG '09*, Apr. 2009, pp.1411-1414.

[7] N. Antunes and M. Vieira, "Comparing the Effectiveness of Penetration Testing and Static Code Analysis on the Detection of SQL Injection Vulnerabilities in Web Services," *15th IEEE Pacific Rim Int. Symp. Dependable Computing, 2009, PRDC '09*, Nov. 2009, pp. 301-306.

[8] C. Kaufman et al., *Network Security: Private Communication in a Public World*, 2nd ed. Upper Saddle River, NJ: Prentice Hall PTR, 2002.

[9] *Password-Based Cryptography Standard*, PKCS #5 v2.0, Mar. 1999.

[10] *U.S. Department of Commerce/National Institute of Standards and Technology (NIST), Advanced Encryption Standard (AES)*, FIPS PUB 197, 2001.

[11] E. Rescorla, *HTTP Over TLS*, IETF RFC 2818, May 2000, [Online]. Available: http://www.ietf.org/rfc/rfc2818.txt

[12] *Java Cryptography Archtecture (JCA) Reference Guide for Java Platform Standard Edition 6*, Oracle Corperation.

[13] *RSA Cryptography Standard*, PKCS #1 v2.1, June 2002.