Using WPKI for Security of Web Transaction

Mohammed Assora, James Kadirire, and Ayoub Shirvani

Anglia Ruskin University
Chelmsford, Essex CM1 1LL, UK
{m.assora, j.kadirire, a.shirvani}@anglia.ac.uk

Abstract. Today, a web transaction is typically protected by using SSL/TLS. SSL/TLS without compulsion for a client's public key certificate, which is the typical usage, is not able to fulfill the security requirements for web transactions. The main remaining threats for this use are client authentication and non-repudiation. This paper presents a scheme to address SSL/TLS security holes, when it is used for web transaction security. The focus is only on transaction that is carried out by using credit/debit cards. The scheme uses wireless public key infrastructure (WPKI) in the client's mobile phone to generate a digital signature for the client. Thus we obtain client authentication and non-repudiation. At the same time, no overhead is imposed on the client, there is no need for any change to the actual system when performing the transaction, and no connection, by using the mobile phone, is required to perform the transaction.

Keywords: Web transaction security, SSL/TLS, digital signature, wireless security services.

1 Introduction

Security is a major concern for all involved in e-Commerce and particularly in the case of web transactions using debit/credit cards. Although the secure electronic transaction (SET) protocol [10] is able to achieve all the security requirements for web transactions, the implementation difficulties have made it redundant and now the preferred method is that of using the secure socket layer (SSL) [3] or transport layer security (TLS) [2]. In particular, SSL/TLS already exists in most web browsers and does not require any special effort from the client. The main security services which SSL/TLS can provide include [5,9,11] :

- Authentication of the merchant by means of his public key certificate.
- Data confidentiality during the transaction by encrypting the exchanged data.
- Data integrity for transactions enabled by using a digital signature.

From the security requirements for web transactions we observe that SSL/TLS does not fulfill all the security requirements, in particular:

G. Psaila and R. Wagner (Eds.): EC-Web 2007, LNCS 4655, pp. 11–20, 2007.

- Authentication for the client. In SSL/TLS, it is optional and typically, it is difficult for a normal client to have a public key certificate. As a result, any one who knows the credit card details is able to complete the transaction on behalf of the client.
- Non-repudiation. Neither the client nor the merchant has any proof of the transaction. The two parties are able to repudiate the transaction or change the transaction details after it has been carried out.
- SSL/TLS provides data confidentiality and data integrity only during the transaction. After completing the transaction, even with a legitimate merchant, all the client's details will be in the merchant's database. Any security breach of this database will lead to the disclosure of all the transaction details.

To eliminate the security risks associated with SSL/TLS, a SSL/TLS connection should be built on mutual authentication i.e. the merchant and client must have public key certificates. If every client has a public key certificate then full public key infrastructure (PKI) should be available and implemented. From previous experience (SET for example), any protocol which uses public key cryptography (PKC), that assumes the availability and full implementation of PKI, will not succeed. This paper presents a scheme to support SSL/TLS as a means for web transaction security. The scheme uses the mobile phone as a tamper resistant and calculation device to store the client's private key and to calculate a digital signature.

2 Using a Digital Signature for Web Transaction Security

In this section the proposed protocol is described. We will first start by the registration or setup phase and then describe the transaction phase. In the scheme the client must have a device to generate a digital signature that we call a digital signature creating device (DSCD), which will be discussed later in Section 3. At this stage we assume the client has a DSCD and she must use it to complete a transaction.

2.1 Registration Phase

This phase is part of the usual procedure which the client must follow to open a bank account and get a debit/credit card from the issuing bank (issuer). In this phase the client submits her credentials such as proof of her identity, address, etc. In addition to the personal details, the client submits her public key to be part of her credentials. The client can generate a pair of public/private keys for herself or through any trusted third party. The client must only use the public key parameters, which are acceptable by the issuer. It would be a good practice to let the issuer generate the cryptographic keys for the client. The private key must be generated and transferred to the client by an authentic method, such as is used today to generate the personal identification number (PIN) for debit/credit cards.

After the keys generation, the client stores the private key in her DSCD and submits her public key to the issuer and there is subsequently no need for a digital certificate for the public key. As part of the trust for the public key during the registration phase, a test for this key is carried out. This test could be carried out by signing a message using the private key in the DSCD and then checking the signature using the public key. In other words, the trust for the public key is part of the trust for the client, which is conducted by her credentials. The issuer must keep the client data confidential and maintain its integrity. The public key needs data integrity only while confidentiality is not required. The client is able to change her keys pair and submit a different copy of her public key at any time in the same way as changing any other personal detail. By finishing this phase the client is ready to use the protocol.

2.2 Transaction Processing

The client browses the merchant's web site and when she decides to pay for goods or services, a secure SSL/TLS connection, based on the merchant's public key certificate, is established. The protocol's steps can be summarized as follows. (see Figure 1).

1. The merchant sends a purchase form (OI) to the client. This purchase form must include the transaction value, date, merchant's identifier, and a unique transaction identifier. In addition to these fields, there is a field which we will call the *challenge* field. This field contains a hash function (HOI) for some selected values from the purchase form. There is also another empty field which we will call the *response* field.
2. The client authenticates herself to DSCD, enters the *challenge* value (HOI) into her DSCD and computes a digital signature (sig). The DSCD will generate a response for the *challenge*. The client fills in the signature value (sig) to the *response* field.
3. When the client submits the purchase form, she will receive the card details form from the merchant.
4. The client fills in her credit card details (PI) and submits the form to the merchant.
5. The merchant submits the selected data from the purchase form (OI), the *challenge* value (HOI), the *response* value (sig), and the debit/credit card form (PI) to the issuer via the acquirer. The issuer must be able to check the validity of the *challenge* value, i.e. the *challenge* value is really a hash function for the selected received values from the purchase form. At the same time, the merchant should not send the goods or service details to the issuer to maintain the privacy of the client.
6. The issuer checks the card details and uses the client's public key to verify the *response*. If these checks are correct the issuer authorizes the payment (Authorization).
7. The merchant informs the client with the result of the transaction (Confirmation) and the transaction is completed.

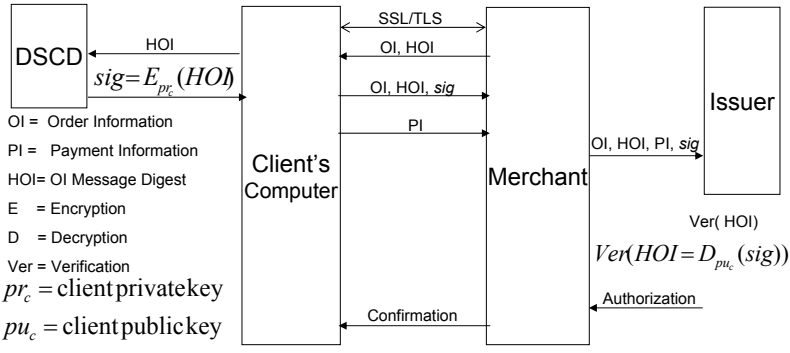

Fig. 1. Transaction process

2.3 Analysis

In this section we examine how much the new protocol covers the security holes of the SSL/TLS protocol, mentioned in Section 1.

- Authentication for the client. Since only the person who has the private key is able to generate the digital signature, so only the client, who has access to the DSCD, is able to complete a transaction. Thus authentication is achieved.
- Non-repudiation. The transaction details in addition to the merchant's identifier are joined together with their hash function (the *challenge* value) and signed by the client (the *response* value), so it is impossible to change any of these values after the transaction has been carried out. The client cannot repudiate the transaction because there is a copy from her digital signature joined up with the transaction. Hence, non repudiation is achieved.
- Data integrity and confidentiality after the transaction. Data integrity is provided after the transaction, as discussed above. The confidentiality of the client's data depends on the confidentiality of the merchant's database, which contains the transaction data. However, this is better than naive SSL/TLS because the user authentication value is not part of this database. If the protocol is widely used and the transaction is not possible without client authentication, then an attacker, who has access to this database, is not able to complete a transaction on behalf of the client.

As a result, by using a digital signature from the client in addition to SSL/TLS built on the merchant's public key certificate, all the security requirements for web transactions have been satisfied.

In practice, there is still a possibility of attack. If a fraudulent merchant changes some values (the price for example) in the OI so the merchant will have fake OI (FOI). Since the merchant generates the hash value for the transaction, the *challenge*, the merchant can generates a hash value for FOI (HFOI) and sends OI and HFOI to the client for signing. Because checking the hash function is too hard for the client, the client will check only the OI and sign the value HFOI. The merchant will submit FOI, HFOI, and the signature to the issuer. As

we can observe, the issuer will find all these values are correct and will authorize FOI. To deter the merchant from this attack, the client must be able to take a soft or hard copy of the order form with the *challenge* and *response* on it. This is equivalent to taking a copy from a contract, in the real world.

3 Digital Signature Creating Device (DSCD)

The DSCD must be able to attain two main objectives. First of all, it needs a secure method to store the client's private key, and secondly, it needs enough processing power to generate the digital signature. Normally, the private key is stored in the client's computer, which could be used to generate the digital signature. But this method has many disadvantages such as destroying the mobility of the client i.e. the client must use the same computer to perform a transaction. There is also the possibility of the private key being stolen by unauthorized access to the computer or by malicious software. Another possibility is to use flash memory to save the keys. The access to the keys can be protected by a password. The method is good because it maintains the mobility of the client but the keys must leave the flash memory to the computer's memory and as a sequence it is still prone to attack by malicious software.

Hiltgen, et. al. [6] proposed the following factors to protect the private key. The private key should be stored in a tamper resistant hardware token such as a microprocessor-based smart card. The smart card should not be connected to the computer and the connection with the smart card should only be achieved via a smart card reader with a keyboard and display. We can observe that a scheme, which uses a smart card with a smart card reader, is secure but it has one major disadvantage which is the need for a smart card reader to perform the transaction. In the next sections we will present a new solution which is able to fulfill the conditions mentioned above and does not add too much overhead to the client.

4 Mobile Phone Security

Today, the mobile phone has become ubiquitous. In every mobile phone there is a subscriber identity module (SIM) which is essentially a smart card and the mobile phone acts as a smart card reader. The SIM contains the necessary information to authenticate the mobile phone to the mobile network. The research in this area, which sets up the mobile phone to be a tool for a secure connection, has a lot of promise because the mobile phone is used to surf the Web and it is necessary to have the ability for a secure connection.

The wireless application protocol (WAP) [15] is a protocol stack for wireless communication and is similar to the Internet protocol (IP) stack. Our interest in the WAP is focused on the security services. WAP security services include wireless transport layer security (WTLS) and application level security (ALS). WTLS [16] is a light version of TLS and is able to achieve the same security services. ALS includes some wireless PKI functions, the most important of which

is the signText() function, defined in the Crypto library [17]. signText() generates digital signatures for the data.

Any parameter of mobile security such as a private key or a session key should be stored in tamper resistant hardware, called the wireless identity module (WIM). In addition to storing the security parameters, any use of these security parameters must be performed in WIM. WIM can be the SIM card itself or it can be another smart card. In the latter case, the mobile phone needs a second slot to install the WIM. Any access to the security functions in the WIM is protected normally by a PIN code. The client can change it at any time, and it is stored in the WIM and cannot be tampered with.

5 Mobile Phone as a DSCD

In this section we discuss how the mobile phone can be used to serve as a DSCD for the protocol mentioned in Section 2. After the client submits her details a *challenge* will be generated. The signText() function, which is used for the signature, is defined as follows [17]:

signedString = Crypto.signText(*stringToSign, options, keyIdType, keyId*)

stringToSign: is the text which will be signed, the *challenge* in our protocol.

options: contains several options for the output such as if the *stringToSign* or a hash function from the public key, which will verify the signature, will be included in the output. In our protocol the options value should be zero.

keyIdType and *keyID*: these two fields are used to identify the public key, which will verify the signature. For example, *keyIdType* may indicate that *keyId* contains a hash function of the public key. In our protocol these two fields are zero. *signedSting* is a string type. It contains the output of the signing function. To summarize, we can write:

signedString = Crypto.signText(*challenge,0,0,0*)

The *response* value consists of *signedString* in addition to some parameters such as the signing algorithm code, the certificate type, and the public key certificate. In our case, where no certificate will be sent, these parameters can be minimized to a few bytes. Thus, the *response* is nearly the same size as *signedString*.

As mentioned above, the *challenge* value is the output of a hash function. The most popular hash function nowadays is the secure hash algorithm 1 (SHA-1) [4]. The output of SHA-1 is 160 bits. The *response* is the result of a digital signature algorithm. The digital signature algorithm (DSA) and the elliptic curve DSA (ECDSA) require twice 2n bits for a security factor of n. Hence, if we require a security equivalent to a 64-bit key search, which is enough for our application, we need twice 128 bits of signature or 256 bits in total.

Crypto library uses Base-64 encoding to encode the *response*. In Base-64 encoding [1] every 6 bits need one character, thus the *response* will be a stream of 43 characters. If we use the same encoding for the *challenge* we need a stream of 27 characters. If the client has to copy one of these streams between the

computer and her mobile phone, first it will be inconvenient, and second there is a high probability of error.

Today, mobile phones are not only used to make phone calls, but also to store and transfer data. Therefore the methods to connect the mobile phones to the computers are very popular. Examples of such methods are Bluetooth, cable, infrared, and memory cards. By using one of these methods, the client can send the *challenge* from her computer to the mobile phone, generate the digital signature and send the *response* to the computer to complete the transaction. However, if the client prefers to exchange the *challenge* and *response* values with the merchant by other means such as a short message service (SMS), then an extra option can be added and the protocol is still applicable.

Asymmetric cryptography needs a lot of memory and is processor intensive and both are not available in mobile phones. The main asymmetric operations for WAP security are WTLS connection, digital signature generation, and digital signature verification. A WTLS connection contains many generations and verifications for digital signatures, so it is much harder than a single digital signature generation or verification. Digital signature verification is harder than generation because to verify a signature signed data must be parsed and split into data and signature parts, the public key must be read from the certificate and the signer's certificate must be validated [7].

The protocol uses only one digital signature generation so there is no excessive use for asymmetric cryptography. Furthermore, the data which will be signed is already hashed by the merchant, the *challenge* value in the protocol. In some cases the hash algorithm is integrated with the sign algorithm, even in this case the extra hash will not add too much overhead because the data is so short. If the calculation is carried out by using WIM, the calculation can be accomplished. The research carried out by Weigold [14] showed that 200 milliseconds are required to calculate the digital signature.

In the market today many mobile phones are ready to perform digital signature generation, examples of which are: the Sony Ericsson (T610 and T68i) and the Nokia (6170 and 6800,). To sign text, by using one of these mobile phones [8], the user should use her mobile phone to browse the merchant web page. The client selects a link to the required goods. The text to sign will be shown and this text includes the goods specification, price, date, etc. The client enters her signature PIN and confirms the signature. The signature will be generated and sent to the merchant.

Unfortunately, neither of the mobile phones in the market today, to the best of the author's knowledge, would be possible to control the input and the output of the signText() function, i.e. we cannot enter a value to the mobile, perform a digital signature to this value, and get the result. The control over the text, which will be signed, is not only required for our protocol but also for any protocol that uses the mobile phone as a general purpose DSCD.

A discussion of the features, advantages, and disadvantages of using the protocol with discussion for other possible options is presented below.

- By using a mobile phone to generate the digital signature, a cover for SSL/TLS security holes is achieved, as mentioned in the discussion in Section 2.3. In particular, there are now two factors in authenticating the client i.e. the client must have the mobile phone in addition to knowing her PIN code. At the same time there is no overhead on the client because the client already has (and always carries) her mobile phone. The only inconvenience is in exchanging some data between the computer and the mobile phone.
- No change is required to the mobile phone or the SIM card. The only need is to implant the Crypto library, which is already standard as part of the WAP security and already implemented in many mobiles phones in the market.
- As we have discussed in Section 2, the authentication for the public key is achieved by means of the client's credentials to the issuer during the registration phase. This gives an advantage that there is no extra cost on the client to obtain a public key certificate from a certificate authority, at the same time there is a disadvantage that the client cannot use her mobile as a general means to sign electronic documents with any other party. Nevertheless, if the client wants to use her mobile phone as a general purpose DSCD, she is still able to have a certificate for her public key and the protocol is still valid.
- From the merchant's view, the response value does not provide any authentication of the client, thus the merchant should wait for the authorization from the issuer to supply the service to the client. However, this is not a big drawback because the merchant should anyway check with the issuer the validity of the client's payment information.
- The generation of the digital signature is carried out locally, by using the mobile phone itself. Consequently, there is no need for any extra communication costs as would be the case if a mobile phone were used to communicate with a third party to perform the authentication.
- There is no need to change the software or hardware in the actual method of transaction. The only requirement is to add two fields to the purchase form to enter the *challenge* and *response*.
- If the client's computer is infected by a virus or Trojan horse, the virus or Trojan horse can collect the transaction details, including the credit card details. However, they are not able to reveal any information about the client's private key. Actually, the *challenge* and *response* values are not confidential. Viruses or Trojan horses can only deny the transaction to be completed by changing the integrity of the *challenge* or *response* values because it will result in a bad signature during the verification. This result is inevitable and can be achieved by other means such as changing the credit card number.
- The client can use her private key for strong authentication in other applications, such as a direct connection to the issuer, as is the case with internet banking. It is also possible for the client to submit her public key to another server and use the protocol above for authentication to that server. No security threat is raised in using the private key to generate a digital signature for more than one application.

– It is possible to share a secret key. Instead of the client storing a private key in her WIM and submitting her public key to the issuer, she shares the same secret key with the issuer. Such a scheme is presented by VISA [13]. In fact, this method is used between the SIM and the mobile home network, which provides the SIM, to authenticate the mobile phone to the mobile network. Sharing a secret key has many advantages such as; firstly, encryption operations by using the secret key are easier and need less memory and computation power compared with PKC. Secondly, the encryption input and output (the *challenge* and *response*) can be reduced to a few bytes, thus there is no need for any connection between the mobile phone and the computer, as the client can enter them manually. At the same time, there are many disadvantages for using a shared secret. Firstly, the client cannot use the same key to authenticate herself to other systems. Although, it is possible to store more than one shared secret in WIM, it is difficult for the client to manage these keys. Secondly, and this is the crucial disadvantage, there is no standard to store and access a shared secret in WIM [13].

6 Related Works

There are many schemes that use the mobile phone for authentication, examples of these schemes with brief review.

3-D Secure by VISA [13]. VISA presented many schemes to use the mobile phone for authentication, such as using a password, a shared secret key, and a public key. The schemes are good and have many ideas for further research but there is a major disadvantage that there must be a direct connection between the client and the issuer during the transaction, so that the client can authenticate herself directly to the issuer. This connection results in extra costs in addition to no solution being presented in the case of where a connection is not available between the client's mobile phone and the issuer. This maybe the case if the client is in a different country during the transaction.

The electronic ID (eID) scheme by Valimo [12]. The scheme uses the mobile phone as a DSCD for general use, i.e. there is a public key certificate for the public key and this certificate is obtained from a government authority (the police). The client can use the digital signature in the same way as a conventional hand written signature. The scheme is good and implemented in Finland. The main disadvantage is that there must be a connection to the mobile service providers (Valimo) to perform an authentication.

7 Conclusion

The simplicity of SSL/TLS makes it the preferable protocol for online transactions security, although it is not able to achieve all the security requirements. The actual use of SSL/TLS, which does not require a client's certificate, raises many risks such as the client authentication and non-repudiation. Performing a digital

signature, by using the client's private key, makes it possible to overcome these security threats. However, a digital signature needs a pair of private/public keys. The public key needs a certificate from a trusted third party, and the private key needs a secure method for storage. That makes it undesirable and difficult to adopt for most clients, because it will add too much overhead for the client.

By using the client's mobile phone as a means to generate the digital signature, a safe place to store the private key has been achieved. Furthermore, the mobility of the client is still maintained because the client always carries her mobile phone with her. The certificate for the public key from a trusted third party is not required because the public key in the protocol must be trusted only by the issuer. This trust is achieved between the issuer and the client during the registration phase, which is part of the procedure to open a bank account and obtain a debit/credit card. The protocol does not require any change to the mobile phone or any additional software or hardware. It only requires an implementation of the wireless security services, which is already standard and implemented successfully in many applications.

References

[1] Borenstein, N., Freed, N.: MIME (Multipurpose Internet Mail Extensions), Part One: Mechanisms for Specifying and Describing the Format of Internet Message Bodies. RFC 1521, IETF (1993)
[2] Dierks, T., Allen, C.: The TLS protocol. ver. 1.0. RFC 2246, IETF (1999)
[3] Freier, A.O., Karlton, P., Kocher, P.C.: The SSL protocol. ver. 3.0. Netscape (1996)
[4] Ferguson, N., Schneier, B.: Practical cryptography. Wiley, Indian (2003)
[5] Hassler, V.: Security Fundamentals for E-Commerce. Artech House, Massachusetts (2000)
[6] Hiltgen, A., Kramp, T., Weigold, T.: Secure Internet Banking Authentication. IEEE Security and Privacy 4(2), 21–29 (2006)
[7] Klingsheim, A.: JABWT and SATSA. NoWires Research Group, Department of Informatics, University of Bergen (2006)
[8] Nokia 6170 user guide, http://nds1.nokia.com/phones/files/guides/Nokia_6170_UG_en.pdf
[9] O'Mahony, D., Peirce, M., Tewari, H.: Electronic payment system for E-Commerce, 2nd edn. Artech House Publishing, Massachusetts (2001)
[10] SETCo: Secure Electronic Transaction Standard- Book, pp. 1–3 (1997)
[11] Stallings, W.: Cryptography and network security principle and practice, 4th edn. Prentice Hall, New Jersey (2006)
[12] Valimo LTD: Mobile Signature services-improving eID, http://www.Valimo.com
[13] Visa International Service Association: 3-D Secure Mobile authentication scenario. ver. 1.0 (2002)
[14] Weigold, T.: Java-Based Wireless Identity Module. University of Westminster, London, UK; IBM Research Laboratory, Zürich, Switzerland (2002)
[15] Wireless Application Protocol Architecture Specification, http://www.openmobilealliance.org/tech/affiliates/wap/wapindex.html
[16] Wireless Transport Layer Security Specification, http://www.openmobilealliance.org/tech/affiliates/wap/wapindex.html
[17] WMLScript Crypto API Library: WAP-161-WMLScriptCrypto-20010620, http://www.openmobilealliance.org/tech/affiliates/wap/wapindex.html