WebIBC: Identity Based Cryptography for Client Side Security in Web Applications

Zhi Guan, Zhen Cao, Xuan Zhao, Zhong Chen, Xianghao Nan
School of Electronics Engineering and Computer Science
Peking University, Beijing 100871, China
{guanzhi, caozhen, zhaoxuan, chen, nanxh}@infosec.pku.edu.cn

Abstract

The growing popularity of web applications in the last few years has led users to give the management of their data to online application providers, which will endanger the security and privacy of the users. In this paper, we present WebIBC, integrate public key cryptography into web applications without any browser plugins. The public key of WebIBC is provided by identity based cryptography, eliminates the need of public key and certificate online retrieval; the private key is supplied by the fragment identifier of the URL inspired from BeamAuth [6]. The test and evaluation shows that WebIBC is secure and efficient on theory and practice.

1 Introduction

With the increasing popularity of Web 2.0 applications like Google Gmail and Google Docs, people are moving their private data and communication information from their local storage to the online application providers. These online applications offer reliable storages and ease to access services. With the AJAX [19] techniques these applications only rely on browsers with common features include HTML, javascript and CSS, without the need of installing any browser plugins or software. These applications make the exchange, management and access of data much simpler than previous desktop applications.

While acquiring ease of use services, users will have to give the control of their data privacy to the application providers. Although application providers announced that these private data will not be abused and will be automatically handled without the involvement of administrator, these applications did not provide any mechanisms to guarantee this promise. Users have to trust the providers to be reliable and honest, and will "do no evil". But some providers have "done evil". One famous example is Yahoo providing user information in its email system to government that helped land a journalist in prison for 10 years [1]. And the leakage of private information will bring greater harm to enterprise users. Some providers like Google and Yahoo also provide services such as Google Apps for enterprise users to take the place of their own email servers and applications. The misuse of provider's privilege will bring huge loses for their customers.

1.1 Related Work

Public key cryptography based solutions for the desktop counterpart of the above web applications have been deployed widely for many years. Pretty Good Privacy (PGP) [20] and S/MIME [16]are two de facto standard, and have been implemented within applications inside many desktop mail clients. The key management of these solutions require ad hoc trust management such as PGP "Web of Trust" or centralized Public Key Infrastructure (PKI). Generally, these methods can be classified into desktop software and browser plugins. A collection of these tools are listed in [2].

1.2 Challenges

Public key cryptography is a fundamental building block for information security that can provide authentication, authorization, integrity and non-repudiation. But public key cryptography was seldom utilized in web applications. The challenges are in twofold:

1. The first challenge is how to get the recipient's public key. In traditional PKI, sender needs to visit an online database to find recipients public key and certificates. Or sender must keep a local database include all possible recipients public key, like PGP. But for web applications, none of these methods are practical. In web browsers javascript programs are restricted in a sandbox. Javascript can only access contents inside the

pages from the same origin. Which means javascript cannot access an LDAP from another server or access local public key database.

2. For the same reason, it is hard to import private key into javascript program. For some solutions, a plugin developed with native language will create a bridge between the browser and the local system. The plugins, for example, IE ActiveX, will provide a javascript object as the interface the access a local file or cryptography devices, such as smart card and USB secure token, which is applied in some e-band systems.

1.3 Our Contribution

In WebIBC, two mechanisms are integrated to resolve the above challenges and providing security and privacy for client side web users. The first one is Identity Based Cryptography (IBC), a type of public key cryptography in which the public key can be an arbitrary string. With IBC scheme, WebIBC can provide public key encryption and digital signature for the web applications without the need of online searching and retrieving of public keys or certificates. Because the recipient's email address, is his public key, can be easily read from the HTTP form in the message sending web page. The javascript implementation of IBC can make WebIBC to be easily integrated into any web applications, and run in all browsers, even text based http clients with javascript extension, such as w3m [5] and lynx [3]. The other is to provide the private key from the URL fragment identifier, the substring starting from the first "#" symbol of a URL. In WebIBC, the private key is encoded into the fragment identifier component of the web application URL.

1.4 Paper Organization

This paper is organized as follows: in section 2 we introduce WebIBC with related theory and techniques, followed by the description of the system architecture and implementation in section 3, and then the performance and security evaluation in section 4. At last section we conclude the paper and introduce future works.

2 WebIBC Basic

In this section, we will introduce IBC and fragment identifier in details firstly, then illustrate the system model of a WebIBC protected web application.

2.1 Identity-Based Cryptography

Identity-based cryptography (IBC) is a form of public key cryptography for which the public key can be an arbitrary string, include email address, domain name, and phone number and user name. The concept was first introduced by Shamir in 1984 [17], used to eliminate the complexity of public key and certificate management. In a scenario that Alice wants to send a message to Bob at bob@domain.com, Alice will not need to retrieve Bobs public key and certificates from a online LDAP (Lightweight Directory Access Protocol) server or from a secure channel, she just simply encrypts the message with bobs email address "bob@domain.com" by an identity based encryption (IBE) scheme. And Bob can decrypt the message with the same scheme. IBS can be classified into Identity Based Encryption (IBE), Identity Based Signature (IBS) and Identity Based Authenticated Key Agreement protocol, or classified by the complexity assumption the scheme based on [10]. After the concept was first suggested, IBS schemes [17], [15], [13] were sooner founded, but IBE scheme is a more challenging problem. Until 2001, the Boneh-Franklin IBE based on Weil pairing was suggested. After that, a serious of IBE schemes were provided include [12], [8], [18].

In the web browser and javascript environment, the schemes based on pairing are too complex and over kill. These schemes require at least 512 bits elliptic curve while only provide security similar to 160 bits Elliptic Curve Diffie-Hellman (ECDH) and Elliptic Curve Digital Signature Algorithm (ECDSA). This is the motivation for selecting Combined Public Key (CPK) cryptosystem as our IBC scheme. A revision version of CPK algorithm is introduced here, which simplify the origin one still remain the security.

2.2 Combined Public Key

The CPK cryptosystem is based on the elliptic curve cryptography (ECC). Let the elliptic curve domain parameters discussed in this paper are a sextuple: $T = (p, a, b, G, n, h)$ consisting of an integer p specifying the finite field $\mathbb{F}_p$, two element $a, b \in \mathbb{F}_p$ specifying an elliptic curve $E(\mathbb{F}_p)$ defined by the equation $E : y^2 \equiv x^3 + a \cdot x + b \,(mod\, p)$, a base point $G = (x_G, y_G)$ on $E(\mathbb{F}_p)$, a prime n which is the order of G, and an integer h which is the cofactor $h = E(\mathbb{F}_p)/n$. The ECC key pair (d, Q) associated with T consists of an elliptic curve secret key d which is an integer in the interval $[1, n-1]$, and an elliptic curve public key $Q = (x_Q, y_Q)$ which is the point $Q = dG$. One character of ECC key pair is that the combination of private keys and corresponding public keys in ECC is still a pair of elliptic curve keys. For example, s_1, s_2 are private key in elliptic curve, the corresponding public keys are Q_1, Q_2 that $Q_1 = s_1 \cdot G, Q_2 = s_2 \cdot G$; $s = s_1 + s_2, Q_t = s_t \cdot G = (s_1 + s_2) \cdot G = (s_1 \cdot G) + (s_2 \cdot G) = Q_1 + Q_2$. So the combination of key pairs will generate a new key pairs.

2.2.1 System Setup

In the setup procedure, a trusted authority will generate a master secret in the system and public parameters known to all entities. Every entity needs to authenticate him to authority, and the authority will extract the private key from the master secret according to entity's identity. In IBC, the authority providing private key extraction service is called a Private Key Generator (PKG). The master key in CPK scheme is a matrix in which elements are ECC private keys. The PKG will choose two positive integer w and k as the column count and row count of the matrix. The elements of matrix are randomly generated private keys with ECC domain parameter T. The matrix is denoted with SKM (Secret Key Matrix).

$$SKM = \begin{bmatrix} r_{11} & r_{12} & \cdots & r_{1k} \\ r_{21} & r_{22} & \cdots & r_{2k} \\ \vdots & \vdots & \ddots & \vdots \\ r_{w1} & r_{w2} & \cdots & r_{wk} \end{bmatrix}$$

The public domain parameters in CPK are a Public Key Matrix (PKM) derived from SKM. PKM has the same size with SKM, the corresponding elements in SKM and PKM compose a key pair in ECC domain parameters T.

$$PKM = SKM \cdot G$$

$$= \begin{bmatrix} P_{11} & P_{12} & \cdots & P_{1k} \\ P_{21} & P_{22} & \cdots & P_{2k} \\ \vdots & \vdots & \ddots & \vdots \\ P_{w1} & P_{w2} & \cdots & P_{wk} \end{bmatrix}$$

$$= \begin{bmatrix} r_{11} \cdot G & r_{12} \cdot G & \cdots & r_{1k} \cdot G \\ r_{21} \cdot G & r_{22} \cdot G & \cdots & r_{2k} \cdot G \\ \vdots & \vdots & \ddots & \vdots \\ r_{w1} \cdot G & r_{w2} \cdot G & \cdots & r_{wk} \cdot G \end{bmatrix}$$

The public parameters also include a cryptography hash algorithm denoted by Map, which map an arbitrary string into indexes of the matrix, which can be used to choose a subset of elements in SKM or PKM. The detail of Map will be describe in next subsection. So the master key of CPK scheme is MSK and public parameters are $\{T, w, k, MPK, Map\}$.

In the original scheme, the Map algorithm is implemented though a list of functions satisfy random oracle model. Thus in this paper we simplified the original one to a standard hash algorithm, that satisfy the hash length is equal or larger than the required index total bits. The simplified Map algorithm is defined as follows:

Input: Matrix column count w and row count k, hash function H, lH is hash size of H, make sure that $lH \geqslant \lceil log_2^k \rceil$.

Output: $\{s_0, s_1, \cdots, s_w\}$, for every $s_i, 0 < i < w, 0 < s_i < k$

Operation:

- $h \leftarrow H(ID)$.
- i from 0 to k: $s_i \leftarrow h \,\%\, w$, $h \leftarrow \left\lfloor \frac{h}{w} \right\rfloor$.

2.2.2 Key Pair Extraction

In an IBC system, the PKG (Private Key Generator) act as two roles, the first is a authority. When a user register in the system, it need to provide some credentials that he has own the identity. The PKG will generate a private key according to the identity. The private key should be delivered to the user by a secure channel. This can be approached by any methods. In CPK scheme, given an identity, the corresponding private key can be extracted from the private matrix SKM and the public key can be extracted from the public matrix PKM. Given ID is the identity string, r_{ij} is the element of SKM at (i, j), P_{ij} is the element of PKM at (i, j). The extraction procedures are as follows:

$$\{s_0, s_1, \cdots, s_w\} \leftarrow Map(ID)$$

$$d = r_{i,s_i}$$

$$Q = P_{i,s_i}$$

And d is the private key, Q is the public key, from section 2.1.1 it is proved that (d, Q) is also an ECC key pair [18].

2.2.3 Encrypt and Sign

After PKG is established, the master secret will be generated and kept securely in PKG, the public parameters will be public to every user. A registered user can get his private key from the PKG, the other users' public keys from the public matrix. The key pair is a standard ECC key pair and any standard ECC signature and encryptions schemes include ECDSA, ECDH and Elliptic Curve Integrated Encryption Scheme (ECIES) can be used.

2.3 URI Fragment Identifier

Fragment identifier is an optional component of a Uniform Resource Identifier (URI) [7]. As the name implies, it addresses a fragment of the resource denoted by the fragment-free URI. In a URI, the fragment identifier component is indicated by the presence of the first "#" character and terminated by the end of the URI. For example, a URL with a fragment identifier looks like: `http://domain.com/index.html#frag_id` The fragment identifier in a URL is used by the browser to jump to a given portion

of the HTML document. Because fragment identifier specified portion is only valid within the context of the main resource, location and changing the portion neither need to reload the page from the server, nor need to pass the fragment identifier from the browser to the server. So the content of fragment identifier will never appear over the network. This character means that identifier can act as a container to store private information, such as authentication token or secret key, for client side web applications. Although fragment identifiers have been used in many web applications, it was first BeamAuth [6] to used as a security token. WebIBC is inspired by BeamAuth to provide private key from the fragment identifier to on page javascript IBC system.

The browser will retrieve and display a page with URL `http://domain.com/index.html#frag_id` as follows:

1. Remove the fragment identifier from the URL, use the remaining address `http://domain.com/index.html` to retrieve the page. Domain name ("domain.com") and path inside the server ("index.html") will be send over the network separately to DNS server and the application server.
2. When retrieving the whole page, the browser if there exists a portion named by the fragment identifier. If not exist, the browser will ignore the fragment identifier.
3. The downloaded javascript in the page can read attributes of the Document Object Model (DOM) object, include the original URL from the attribute "window.document.URL", then it can parse the fragment identifier string from the original URL string, and use it for any propose.

2.4 Working Flow

In a scenario that Alice wants to send a secure message throw a web mail enhanced with WebIBC, the working flow is as follows:

1. The authority trusted by Alice and Bob establishes a PKG. And generate the public system parameters include the public matrix PKM.
2. Web application embeds WebIBC into these systems together with the public system parameters released by the PKG.
3. Alice registers to the PKG to get her private key d_A. PKG will create a URL of the web application, encodes Alice's private key into the fragment identifier and append it to the URL.
4. Alice receives the URL with her private key, and adds it to the bookmark of her browser.
5. Now Alice can use this bookmark to log into the web application, the WebIBC javascript files will also be downloaded from the server, include the public matrix of system.
6. Alice uses this web application as normal, entering Bob's email address and message content into the form.
7. When Alice press the send button, javascript program will get the email address from the form, and extract Bob's public key from the matrix. Use this public key with Elliptic Curve Integrated Encryption Scheme (ECIES) to encrypt the message.
8. If Alice wants, she can also sign an ECDSA signature with the private key imported from the bookmark URL fragment identifier component. And then the message will be sent to the server.
9. Bob can also retrieve a URL from the PKG and use it to verify and decrypt Alice's message.

It should be notice that all the cryptography operations are all done within the browser, and the server can only receive the ciphertext. The security and privacy of end users can be protected from attacks both on network and server side. From another point of view, server is also free from the burden of cryptography operations which means WebIBC is a good model for distributed computation based on web browsers.

3 Implementation

The WebIBC include three components: (1) The JavaScript Crypto Library that implements CPK scheme in javascript (2) IBC PKG server that provides private key extraction services, and (3) Web Application Server in which a web mail is provided for the demonstration of concept of WebIBC . In this section, we describe our implementation details of WebIBC. All the system, demonstration and benchmarks are available at `http://infosec.pku.edu.cn/~guanzhi/webibc/`

3.1 JavaScript Crypto Library

The core function of WebIBC is provided by a javascript crpyto library that can be integrated into any web applications. This library includes a set of cryptography algorithms; include SHA-1, AES, big integer operations, ECC and CPK. We get the SHA-1 and AES javascript implementation from the Internet. And Big Integer implementation from a javascript RSA library. The BigInteger is designed for RSA, and not efficient for ECC, we implement efficient big integer reduction for NIST primes.

3.2 PKG Server and Web Server

The PKG server act as a trusted authority in the system. It generates the master key and public system parameters, and provider private key extraction services to users. Before applying WebIBC, user must authenticate himself to the PKG; prove the ownership of the identity string, in our demo system, is an email address. If the authentication succeeds, user can retrieve his private key from the server. For the demonstration, we also provide a sample implement an on page PKG server by javascript. It is not secure, just to show how it runs.

We implement a simple web mail as our application server. After login, user can send a message and view received messages.

4 Performance and Evaluation

In this section computation and bandwidth overheads of WebIBC are evaluated from tests and analysis firstly. The benchmark result shows that the performance of our prove of concept implementation is acceptable for both users and browsers. Optimization methods from algorithm and programming practice introduced later can enhance the efficiency of WebIBC immensely; the estimated delay will be unnoticeable to users. The security of WebIBC under some possible attacks will also be discussed at the end of this section.

4.1 Computation Overhead

Computation overhead is determined by many factors, include the performance of testing environment, how much data to process, how many recipients are involved and the needed security level of cryptography schemes. Before the evaluation, two questions must be answered; the first is what are the average values of these factors for a given application? the other is, what is the maximum overhead can the application sustain without destroying user experiences? For a practical mechanism, answer of the second question must be satisfied, while how to satisfy the above factors is a tradeoff that can be the design choice of the system.

We select distinct machines for the test. Our main benchmark environment is an Apple MacBook laptop with 1.83 GHz Intel Core 2 Duo processor, 2 GB RAM and installed with dual operating systems. The javascript programs are tested in Safari 3.03, Firefox 2.0.0.9, Opera 9.24 on Mac OS X and Internet Explorer 7 (IE7) on Windows XP Professional. We also test the system on a much older Dell desktop computer with 2.6 GHz Pentium 4 processor and 256 MB RAM running Internet Explorer 6 (IE6).

From some tests the second question is sooner found. For all the four browsers there is a timeout mechanism on the JavaScript programs. When a javascript program running over about 5 seconds without any response, the browser will consider the program to be a "Slow Script", so pause it and display an alert dialog to ask the user whether to terminate it or not. The 5 seconds limit is not affected by the speed of the computer; it remains the same in all our test machines. This alert will break the running and confuse the user, so 5 seconds may be considered as the upper bound of computation time overhead.

The computation of WebIBC is mainly coming from three basic cryptography building blocks: AES, SHA-1 and big integer module MULtiply (MUL) operation. The MUL is the main computation component of ECC and CPK. When the unit performance is known, the total running time can be estimated for specific factors. The table below shows the result of test on the unit speed of these functions.

We run a benchmark include SHA-1, AES with 128 bits key and 192 bits big integer MUL on different browsers, so we can estimate the total computation overhead of WebIBC with different data sets and system parameters. The unit of SHA-1 and AES result is bytes per millisecond, the unit of MUL is times per milliseconds.

	IE	Safari	Firefox	Opera
SHA-1	20.5	22.1	39.4	25.1
AES	13. 46	14.6	15.4	8.4
MUL	1.12	2.09	1.91	1.39

Now let us consider the factors will affect the final result:

ECC Key Length Every CPK operation require k elliptic curve point addition and one scalar multiply. The details of these algorithm will not be introduced here, the descriptions will be found in [**?**]. One point addition include 10 MULs, one point doubling include 8 MULs. Given l is the key length in bits, there need l point doubling and $0.5l$ point addition on average. The selected key length is 192 bits, provide security same to 1776 bits RSA. This requires about 2500 MUL operations for a CPK encryption.

Payload Size The average data size is different for various applications. For example the most popular web mail, email statics from UC Berkeley [4] show that the averages email content size is 1863 bytes. So we use 3 different testing data size, 0.5KB, 2KB and 10KB. In our test we did not consider the process of email attachments, for one reason is attachment might be several megabytes which is too large for current JavaScript cryptography functions to process, the other is we did not find a method for JavaScript to get the attachment before it is sent to the server through a HTTP POST.

The result of our test is shown in figure 1.

Even on the slowest browser with the biggest data size, the total time is from 2.1 seconds to 3.6 seconds, less than

	0.5KB	2KB	10KB
Safari	1383.7	1492	2071
Firefox	1523	1661	2401
IE	1459	1698	2791
Opera	2110	2349	3628

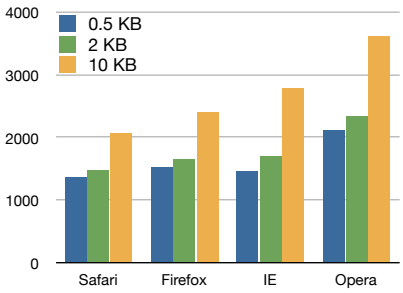

Figure 1. Evaluation result of computation overhead

the 5 seconds limitation. For the average data size - 2KB and the 3 of the most popular browsers IE, Firefox and Safari, the overhead is about the same, from 1.4 seconds to 1.6 seconds.

4.2 Possible Optimization

Our tests on the slower computer will exceed the 5 seconds limits. So WebIBC still need some optimization to fulfill the need of old machines. For the implementation of WebIBC, some possible optimization method is listed here.

1. Reduce ECC key length from 192 bits to 160 bits, which provide the security similar to 1128 bits RSA. The amount of MUL operations will be reduced to 83% of 192 bits key. The time for every MUL will also be reduced.

2. Faster ECC algorithm. In our prove of concept implementation, the slowest point scalar multiply algorithm is used. In [**?**] some optimized algorithms are introduced. And for encrypt and sign operations, 66% of the scalar multiply can pre computed.

3. Technique methods. Although the cryptography operations are very time consuming, compare to user's performance, include thinking and writing the message, the overhead of cryptography operation can be neglected. With Web 2.0 javascript techniques, these operations can be separately finished during the long period when user editing the message. The performance of AES and SHA-1 (about 20KB per seconds) is much faster than people's editing speed.

4.3 Bandwidth Overhead

The bandwidth overhead caused by WebIBC is coming from the downloading of javascript crypto library and system public parameters of IBC scheme. In our prototype, the crypto library include AES, SHA-1, big integer operation, ECC and CPK javascript implementations. Current total WebIBC javascript code size is 61KB, about 1600 lines. The size of public system parameters is determined by the size of the CPK public matrix. Given c is the matrix column count and the r is row count, there need about $96 \times c \times r$ bytes to store the public matrix with uncompressed elliptic curve points, or about $49 \times c \times r$ bytes with compressed elliptic curve points. In a 32×32 matrix with 192 bits ECC, the size is less than 100KB.

The format of elliptic curve points is a tradeoff in WebIBC. Although the compressed format decrease half of the matrix size, there need r extra computations to get the y coordinate of the mapped elliptic curve points. From our last test, it shows that at current state the computation is the bottleneck of our system, because compare to a web page, the bandwidth overhead is acceptable, and if the browser has cache on these javascript files, it can be neglect.

4.4 Security Analysis

In this section, we will define the attack model and analysis the security of the WebIBC system. In this paper we use CPK algorithm as the IBE algorithm implementation in the WebIBC. There are two reasons for us to use this algorithm. At the first, CPK is an Identity Based Cryptography algorithm that can do both IBE and IBS. Do not like other algorithms like BF-IBE or PKI based algorithms like DSA, these algorithms can only provide one mechanism, only encryption or only signature. The other reason is that javascript is not an efficient language, especially for cryptography algorithm implementations. For BF-IBE, (in this place, we will add some simulations on the efficiency of the two algorithms).

4.4.1 Security of Cryptography

The security of WebIBC is depending on the security of cryptography algorithms it applies. We will discuss the security of these cryptography algorithms from theory and practice. In WebIBC we apply CPK as our fundamental IBC scheme. As we have discussed, CPK is a bounded IBC scheme, it can defend a fixed number of collude users.

One of the solutions is to replace CPK with other proved secure IBE schemes, for example, Boneh Franklin IBE have been proved secure in theory. But at current web application environment, these schemes are not efficient enough. For Boneh Franklin IBE [9] is based on Weil pairing, the computation need the computation at least 512 bit elliptic curve. At current we do not have an implementation of BF-IBE on javascript, from a benchmark of ECC performance, the 512 bit elliptic curve is ten times slower than the 160 bit elliptic curve, and the ECC point scalar multiply is only a low level operation. So at current, BF-IBE is not practical. Cocks IBE is also not efficient on bandwidth, for every bit of plaintext, the output ciphertext will be expanded to 2048 bit.

The security of cryptographic hash algorithm should also be considered. The hash algorithm will be used in two places, the CPK map function and by the digital signature procedure. In our implementation we use SHA-1, while SHA-1 has recently been shown to have certain weaknesses. This weakness can be resolved by replacing SHA-1 with stronger SHA-2 family hash algorithms. The later test in next sub section shows that the computation overhead of hash algorithm in WebIBC for average email data size is neglectful.

4.5 Private Key Security

Modern browsers implement the same-origin policy that prohibits a web object from one site from accessing web objects served from a different site. Browsers currently enforce this by checking that the two objects' originating domain names, ports and protocols match. However, if the attacker controls the domain mapping, the same-origin policy will be broken. This attack can be accomplished by Trojan horse to tamper the /etc/hosts config file or through the DNS rebinding attack. Then the javascript downloaded from the attacker server can gain complete control of the session, include the private key in the fragment identifier. Some recent researches [14], [11] have address the attack on same-origin policy but without widespread deployments. So WebIBC is still vulnerable to these attacks.

4.5.1 Phishing

Phishing attacks use both social engineering and technical subterfuge to steal consumers' personal identity data and financial account credentials. Social-engineering schemes use 'spoofed' e-mails to lead consumers to counterfeit websites designed to trick recipients into divulging financial data such as credit card numbers, account usernames, passwords and social security numbers. Hijacking brand names of banks, e-retailers and credit card companies, phishers often convince recipients to respond. Technical subterfuge schemes plant crimeware onto PCs to steal credentials directly, often using Trojan keylogger spyware. Pharming crimeware misdirects users to fraudulent sites or proxy servers, typically through DNS hijacking or poisoning.

4.5.2 Server Cheating

One possible threat is coming from the application server. In WebIBC, the cryptosystem implemented by JavaScript is embedded in the application pages that downloaded from application HTTP servers. The security of WebIBC depended on the reliable of the correctness of the JavaScript program and data. If the server provide fake script, the security will be broken.

We provide some solutions to this problem:

1. WebIBC is a mechanism provided to some honest service provider like Google Mail, to provide a guarantee to customers that they can protect users privacy by mechanism, not just policy. And because WebIBC is totally "open source", users can check if the provider release a valid security implementation.

2. In a web page, JavaScript can be link from other addresses, So the implementation can be downloaded from a trust server, for example, the server belonging to the PKG.

3. Some browsers have generalized plugins that can replace some contents in a web page, such as CSS or scripts. Users can utilize this kind of plugins to insert the trusted WebIBC implementations into the page.

4.6 Limitation

There are still some limitations in our system, include: the javascript is provided by the service provider, so it might be modified. And for an email application, the attachment is hard to be protected by WebIBC.

5 Conclusion and Future Works

In this paper, we present WebIBC to protect the client side security and privacy of web applications. WebIBC integrating identity based cryptosystem into web based applications and is total established by javascript without any browser plugins. The performance of WebIBC prototype is acceptable and has potential to be highly optimized. The security and some possible attacks on WebIBC are also discussed. The future work of WebIBC is to evaluation the feasibility of other IBC schemes on WebIBC, especially Boneh Franklin IBE.

References

[1] CNN news: Yahoo accused of misleading congress. http://www.cnn.com/2007/US/10/16/yahoo.congress/.

[2] Epic online guide to practical privacy tools. http://www.epic.org/privacy/tools.html.

[3] Lynx: a text browser for the world wide web. http://lynx.browser.org/.

[4] UC Berkerly email stats. http://www2.sims.berkeley.edu/research/projects/how-much-info/internet/.

[5] W3m, a text-based browser and pager. http://w3m.sourceforge.net/.

[6] B. Adida. Beamauth: two-factor web authentication with a bookmark. In *CCS '07: Proceedings of the 14th ACM conference on Computer and communications security*, pages 48–57, New York, NY, USA, 2007. ACM.

[7] T. Berners-Lee, R. Fielding, and L. Masinter. Uniform resource identifier (URI): General syntex. 2005.

[8] D. Boneh and X. Boyen. Secure identity based encryption without random oracles. 2004.

[9] D. Boneh and M. Franklin. Identity-based encryption from the Weil pairing. *Lecture Notes in Computer Science*, 2139, 2001.

[10] X. Boyen. General ad hoc encryption from exponent inversion ibe, 2007.

[11] D. W. Chris Karlof, J. D. Tygar and U. Shankar. Dynamic pharming attacks and locked same-orgin policies for web browsers. *ACM Conference on Computer and Communications Security*, 2007.

[12] C. Cocks. An identity based encryption scheme based on quadratic residues. *Lecture Notes In Computer Science*, 2260:360–363, 2001.

[13] F. Hess. Efficient identity based signature schemes based on pairings, 2003. SAC 2002.

[14] C. Jackson, A. Barth, A. Bortz, W. Shao, and D. Boneh. Protecting browsers from dns rebinding attack. *ACM Conference on Computer and Communications Security*, 2007.

[15] K. Paterson. Id-based signatures from pairings on elliptic curves.

[16] B. Ramsdell. RFC 2633 - S/MIME version 3 message specification.

[17] A. Shamir. Identity-based cryptosystems and signature schemes. *Crypto '84*, pages 47– 53, 1985.

[18] W. Tang and X. Nan. CPK Cryptosystem. 2004.

[19] C. Wenz. Javascript und ajax. 2007.

[20] P. Zimmermann. The official PGP user's guide. 1995.