Hash-Based Password Authentication Protocol Against Phishing and Pharming Attacks

IKSU KIM1 AND YONGYUN CHO2,+
1School of Computer Science and Engineering
Soongsil University
Seoul, 156-743 Korea
2Information and Communication Engineering
Sunchon National University
Jeonnam, 540-742 Korea

Until now, although many researchers proposed a variety of authentication protocol to verify the identity of the clients, most of these protocols are inefficient and ineffective. Gouda et al. proposed an anti-phishing single password protocol, but it is vulnerable to pharming attacks. In this paper, we show that the protocol is insecure, and propose a hash-based password authentication protocol against phishing and pharming attacks. In the proposed protocol, the authentication tickets passed between clients and servers are secure because they are hash values which can be verified only by clients and servers. The authentication ticket is used only once, which ensures that the proposed protocol is secure against a variety of attacks such as replay, man-in-the-middle, phishing, and pharming. Because the proposed authentication protocol does not require encryption keys during the authentication phase, it is suitable for wireless and mobile communication systems.

**Keywords:** authentication protocol, phishing attack, pharming attack, web security, hash function

1. INTRODUCTION

In most server-client models, clients log on to a server using an ID and password. However, authentication based on passwords is vulnerable to a variety of attacks, including eavesdropping, man-in-the-middle, and replay. In addition, phishing incidents have been increasing since the eBay phishing incident that occurred in 2003. Phishing is a malicious technique that employs social engineering and technical fraud to steal sensitive information and financial account credentials from clients. More than 5 million U.S. clients lost money to phishing attacks in the 12 months ending September 2008, and phishing victims lost an average of $351 per incident [1]. Table 1 shows the summarized phishing statistics from Phishing Activity Trends Report which Anti-Phishing Working Group (APWG) released in April 2012 [2]. Unique phishing reports rose to an annual high of 32,979 in December 2011, and unique phishing websites detected was 48,410.

In a phishing attack, an attacker first sets up a fake server that mimics a well-known,
Table 1. Statistical highlights for 2nd Half, 2011.

<table>
<thead>
<tr>
<th></th>
<th></th>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>Number of unique phishing email reports (campaigns) received by APWG from consumers</td>
<td>19,606</td>
<td>25,685</td>
<td>32,979</td>
</tr>
<tr>
<td>Number of unique phishing web sites detected</td>
<td>36,733</td>
<td>44,030</td>
<td>48,410</td>
</tr>
<tr>
<td>Number of brands hijacked by phishing campaigns</td>
<td>311</td>
<td>348</td>
<td>362</td>
</tr>
<tr>
<td>Country hosting the most phishing websites</td>
<td>USA</td>
<td>USA</td>
<td>USA</td>
</tr>
<tr>
<td>Contain some form of target name in URL</td>
<td>66.61%</td>
<td>55.72%</td>
<td>56.94%</td>
</tr>
<tr>
<td>No hostname; just IP address</td>
<td>1.37%</td>
<td>1.25%</td>
<td>0.92%</td>
</tr>
<tr>
<td>Percentage of sites not using port 80</td>
<td>0.27%</td>
<td>0.28%</td>
<td>0.64%</td>
</tr>
</tbody>
</table>

legitimate server. The attacker then sends a large number of spoofed e-mails to random clients. If the client receiving the e-mail clicks on the hyperlinks in the message, they are redirected to the fake server instead of the legitimate server. The client submits his or her ID and password in order to log on to the fake server, believing it to be the genuine website. As a result, the attacker is able to steal IDs and passwords, which are then used to conduct online transactions.

Most browsers now use the URL filtering method to protect clients from phishing attacks. This method detects phishing servers by using a blacklist file, which lists the URLs of servers reported as phishing servers. However, the URL filtering method cannot detect new phishing sites because it is based on information collected in advance. Therefore, research has been conducted into anti-phishing protocols and services [3-6]. However, these protocols and services need encryption key operations or an instant messaging service during the authentication phase and are vulnerable to pharming attacks. Gouda et al. proposed an anti-phishing single password protocol [3]. However, this protocol is vulnerable to pharming attacks.

In this paper, we show that the protocol proposed by Gouda et al. is insecure, and propose a hash-based password authentication protocol against phishing and pharming attacks. In the proposed protocol, a client first sends an ID to a server. The server then generates and sends an authentication ticket to the client. Only if the server is successfully verified by the client, the client generates and sends another authentication ticket to the server. The client-server authentication tickets can be verified only by the legitimate client and server. Therefore, the proposed protocol is secure against a variety of attacks such as replay, man-in-the-middle, phishing, and pharming, on ubiquitous networks. Moreover, the proposed authentication protocol is suitable for wireless and mobile communication systems because it does not require encryption keys during the authentication phase.

The rest of this paper is organized as follows. In Section 2, we explore the evolution of phishing and introduce previous work. Section 3 provides an analysis of the vulnerability of the single password protocol (SPP) proposed by Gouda et al., and describes the proposed protocol. Section 4 analyzes the security of the proposed protocol, and Section 5 concludes this paper.

2. RELATED WORKS

2.1 Evolution of Phishing

In phishing attacks, after an attacker sets up a fake server that mimics a well-known
legitimate server, he sends e-mails written in HTML to random clients, as shown in Fig. 1.

If the recipient of the e-mail clicks the hyperlink “http://www.trustedland.com” in the message, they are redirected to a fake server instead of the TrustedLand server. The hyperlink in the e-mail is actually as follows:

\[<a href="http://fakeserver.com">http://www.trustedland.com</a>\]

The e-mail recipient believes that they have been redirected to the TrustedLand server. Therefore, they willingly submit their ID and password to log on to the fake server. Thus the submitted passwords are exposed to the attacker. If the client notices the address of the fake server in the address bar in the browser, they could perceive the phishing attack because the address is not the address of the legitimate server.

Sometimes attackers register a domain name similar to that of a legitimate server. For example, if we assume that the domain name of a legitimate server is “www.trustedland.com”, an attacker might register a domain name such as “www.trusted1and.com”, which uses the number “1” instead of the letter “l”. This would make it much more difficult to notice that the address is fake, and that it is a phishing attacks.

In the case of a browser spoofing attack, it is much more difficult for clients to detect phishing attacks by looking at the address of the website [7]. In this type of attack, the attacker hides the address bar and places a textbox at the top of the webpage that looks like the address bar. The textbox then contains the address of the legitimate server. The client might not notice that the address bar looks slightly different, and will therefore not perceive the browser spoofing attack. For a successful browser spoofing attack, the browser must be able to create a new pop-up window by calling the method window.open(). Clients can prevent browser spoofing attacks by enabling the pop-up blocking feature provided by most reputable web browsers.

Pharming attacks can be conducted by exploiting vulnerability in DNS server software, which is a more advanced type of attack than phishing. Here, attackers alter the IP address information stored in a DNS server so that even when clients input the correct
domain name of a legitimate server in their web browser, they are redirected to a fake server. In this case, the address in the address bar in the browser shows the correct address of the legitimate server, which makes it very difficult to perceive the pharming attack.

2.2 Previous Work

There are many anti-phishing security solutions that use browser extensions and toolbars in order to reduce the risks of phishing attacks [8-13]. Web browsers such as Internet Explorer and Firefox support browser extensions and toolbar functions that developers can use to implement anti-phishing tools.

Microsoft Internet Explorer provides a SmartScreen Filter that helps detect phishing servers. Using a list of reported phishing servers, the SmartScreen Filter checks whether clients are visiting a phishing server [8]. If clients visit a phishing server, the SmartScreen Filter displays a warning message. However, the SmartScreen Filter cannot detect phishing servers other than those that have already been reported. For this reason, Sharifi et al. proposed a blacklist generator that maintains an up-to-date blacklist of phishing servers [14]. The evaluation results of the technique have shown an accuracy of 91% in detecting legitimate pages and 100% in detecting phishing servers.

Miyamoto et al. proposed a filtering algorithm called SPS [15] that protects clients from phishing attacks by removing part of the malicious content that traps clients into entering personal information. In SPS, URLs on HTML documents are checked by a URL filtering component. If a URL is suspicious, the HTTP response sanitizing component removes any input forms from the HTML documents. However, SPS is vulnerable to pharming attacks.

Chou et al. proposed SpoofGuard [16] that examines the domain name, images, and links on HTML documents to detect phishing attacks. SpoofGuard provides traffic lights (green, yellow, or red) in a toolbar indicating whether the HTML documents are spoofed. SpoofGuard displays a warning if client opens a new account with the same ID and password. Ross et al. proposed PwdHash [17], which stores the hash value of a client’s password and server domain name rather than storing the client’s password in a password file. Therefore, although a client might use the same password on several servers, the verification information stored on each server is different. PwdHash makes it more difficult for attackers to find passwords through dictionary and brute-force attacks, but it is also vulnerable to pharming attacks.

Sandirigama et al. proposed a simple and secure password authentication protocol (SAS) [18]. However, Lin et al. showed that SAS was vulnerable to a message replay attack [19]. Huang et al. proposed an anti-phishing authentication service with one-time passwords instead of fixed passwords [4]. This service requires instant messaging services, which means that clients must install at least one instant messaging client. A one-time password authentication scheme proposed by Kuo and Lee is vulnerable to a modification attack and a replay attack [20]. Kim et al. proposed a scheme that resolves the security flaws found in Kuo and Lee’s scheme [21]. However, this scheme requires shared encryption keys. The anti-phishing authentication (APA) protocol and anti-phishing protocol with mutual authentication (APPMA) also require encryption keys in the authentication phase [5, 6].
Gouda et al. proposed the single password protocol (SPP), which is secure against various attacks, including eavesdropping, message replay, and phishing [3]. In this protocol, a client can use the same ID and password on multiple web servers through the use of salted passwords. In SPP, it takes much more time for attackers to crack clients’ passwords because they cannot create pre-computed hash values of passwords. However, SPP is vulnerable to pharming attacks.

3. HASH-BASED PASSWORD AUTHENTICATION PROTOCOL

3.1 Vulnerability of SPP

In this section, we describe SPP and analyze its vulnerability. The notations used in SPP are listed in Table 2 and the protocol is shown in Fig. 2. In this protocol, when C registers with S, C generates a random number \( n_i \) and ticket verification information \( \text{MD}^2(n_i|P|S) \). This information is then sent to S, which it stores in the password file. Thereafter, when C tries to log on to S, S prompts C with the stored value, \( n_i \). C generates ticket \( \text{MD}(n_i|P|S) \) for authentication, a new random number \( n_{i+1} \), and new ticket verification information \( \text{MD}^2(n_{i+1}|P|S) \) and sends everything to S. S then authenticates C using the ticket verification information stored in the password file. If C is successfully authenticated, S replaces \( n_i \) and \( \text{MD}^2(n_i|P|S) \) by \( n_{i+1} \) and \( \text{MD}^2(n_{i+1}|P|S) \), respectively.

<table>
<thead>
<tr>
<th>Notation</th>
<th>Representation</th>
</tr>
</thead>
<tbody>
<tr>
<td>C</td>
<td>Client</td>
</tr>
<tr>
<td>S</td>
<td>Server</td>
</tr>
<tr>
<td>P</td>
<td>Password remembered by client</td>
</tr>
<tr>
<td>( n_i, n_{i+1} )</td>
<td>Random number</td>
</tr>
<tr>
<td>( \text{MD}(\cdot) )</td>
<td>One-way hash function</td>
</tr>
<tr>
<td>( \text{MD}^2(\cdot) )</td>
<td>( \text{MD}(\text{MD}(\cdot)) )</td>
</tr>
<tr>
<td>(</td>
<td>)</td>
</tr>
</tbody>
</table>

![SPP protocol diagram](image)

Fig. 2. SPP protocol.
SPP is vulnerable to pharming attacks when an attacker is able to replace the IP address of a legitimate server with that of the malicious server on a DNS server. As shown in Fig. 3, when a client, C, tries to log on to the legitimate server, the identification information first passes through the malicious server, which forwards it to the legitimate server. The legitimate server responds by sending $n_i$ to the malicious server, which passes this on to the client, C. Now, the client C sends $\text{MD}(n_i|P|S)|n_{i+1}|\text{MD}^2(n_{i+1}|P|S)$ to the malicious server, and the server has access to the login information. The attacker can now log on to the legitimate server using the information.

![Fig. 3. Attack on SPP protocol.](image)

### 3.2 Registration Phase

In this section, we present a hash-based password authentication protocol against phishing and pharming attacks over the Internet. In the proposed protocol, we first assume that clients choose secure passwords against brute force attacks and dictionary attacks. In other words, the passwords consist of at least eight random characters. On a standard computer keyboard, the character types include 10 numeric numbers, 26 lowercase characters, 26 uppercase characters, and 32 special characters. The number of combination for a password length of 8 digits is $94^8$. It would therefore potentially take years for an attacker to try every possible combination of letters, numbers, and special characters before discovering the correct one. Second, we assume that the proposed protocol is used with the Secure Sockets Layer (SSL) that provides communication security over the Internet. The reason for assuming so is because most servers, such as banking servers, e-mail server, and online shopping servers, are using the SSL for security. The SSL can prevent eavesdropping and tampering, but it is still vulnerable to man-in-the-middle attacks.

In the proposed protocol, the SSL protocol runs before the registration phase begins, as follows:

**Step 1:** A client’s machine requests the registration page from a server.

**Step 2:** The server sends its certificate to the client’s machine.

**Step 3:** The client’s machine authenticates the server and generates a session key.
Step 4: The client’s machine encrypts the session key with the server’s public key and sends the encrypted session key to the server.

Step 5: The server decrypts the encrypted session key with its private key.

Now the client’s machine and server can securely send and receive messages, and the registration phase begins. The registration phase is shown in Fig. 4 and the notation used in this section is listed in Table 3.

### Table 3. Notation.

<table>
<thead>
<tr>
<th>Notation</th>
<th>Representation</th>
</tr>
</thead>
<tbody>
<tr>
<td>ID</td>
<td>Identity of client</td>
</tr>
<tr>
<td>P</td>
<td>Original password of client</td>
</tr>
<tr>
<td>P'</td>
<td>Password submitted by client</td>
</tr>
<tr>
<td>D</td>
<td>Domain name of legitimate server</td>
</tr>
<tr>
<td>D'</td>
<td>Domain name of fake server</td>
</tr>
<tr>
<td>IP_{in}</td>
<td>Source IP address in packet header</td>
</tr>
<tr>
<td>IP_{s}</td>
<td>Server IP address</td>
</tr>
<tr>
<td>IP_{c}</td>
<td>Client IP address</td>
</tr>
<tr>
<td>N_s</td>
<td>Random nonce generated by server</td>
</tr>
<tr>
<td>H()</td>
<td>One-way hash function</td>
</tr>
<tr>
<td>M_c</td>
<td>Verification message from client</td>
</tr>
<tr>
<td>M_s</td>
<td>Verification message from server</td>
</tr>
<tr>
<td>|</td>
<td>Concatenation</td>
</tr>
<tr>
<td>+\oplus</td>
<td>Exclusive-or</td>
</tr>
</tbody>
</table>

![Fig. 4. Registration phase.](image)

When a client registers with a server, he determines an ID and password (P), and enters them into the registration form. Before ID and P are sent to the server, P is hashed with the salt value, D. Thereafter, ID and H(P|D) are sent to the server, and the server stores the ID and H(P|D) in the password file.

### 3.3 Authentication Phase

The authentication phase is shown in Fig. 5. When a client tries to log on to the server, he inputs his ID and P' into the input form for authentication. The client’s machine sends only the ID to the server. If the ID exists in the password file on the server, the server generates N_s, and computes N_s + H(P|D) and H(P|D) + H(H(P|D)||IP_c||N_s). The server then sends (H(P|D) + H(H(P|D)||IP_c||N_s))||(N_s + H(P|D)) to the client’s machine.
Because the client’s machine knows $P'$ and $D$, it can compute $H(P'|D)$. After the client’s machine computes $H(P'|D)$, it extracts $N_{c}$ from $N_{c} \oplus H(P|D)$ and computes $H(P'|D) \oplus H(H(P'|D)\oplus IP_{c}\oplus N_{c})$. 

If $H(P'|D) \oplus H(H(P'|D)\oplus IP_{c}\oplus N_{c})$ is the same as $H(P|D) \oplus H(H(P|D)\oplus IP_{c}\oplus N_{c})$ received from the server, the server is verified. This is because the same value for $H(P'|D) \oplus H(H(P'|D)\oplus IP_{c}\oplus N_{c})$ can only be computed by the server that knows $H(P|D)$.

After the client’s machine verifies the server, it computes $H(P'|D) \oplus H(H(P'|D)\oplus IP_{c}\oplus N_{c})$ and sends the computed value to the server. Because the server knows $H(P|D)$, $IP_{bs}$, and $N_{c}$, it can compute $H(P|D) \oplus H(H(P|D)\oplus IP_{bs}\oplus N_{c})$. If the computed value is the same as $H(P'|D) \oplus H(H(P'|D)\oplus IP_{c}\oplus N_{c})$ received from the client’s machine, the client’s machine is verified. The same value for $H(P|D) \oplus H(H(P|D)\oplus IP_{bs}\oplus N_{c})$ can only be computed by the client’s machine that knows both $H(P|D)$ and $N_{c}$. If the client’s machine and the server are not verified by each other during the authentication phase, the authentication fails.

In current situation of the Internet, NAT is widely used in various networks. In the proposed protocol, when a client and/or a server is in private networks, the IPc and/or the IP, have the external IP addresses of the client-side and/or the server-side NAT devices, respectively. Therefore, the proposed protocol can work well in private network environment.

4. SECURITY ANALYSIS

4.1 Eavesdropping Attack

An eavesdropping attack is the act of secretly listening to the private conversation
of others. On the Internet, an attacker listens to network communication and thus tries to steal a client’s password. An eavesdropping attack is impossible when using the proposed protocol. An attacker cannot compute $P$ and $H(P|D)$ from the verification messages $(H(P|D) \oplus H(H(P|D)|IPs|Ns))(Ns \oplus H(P|D))$ and $H(H(P'|D)||IP_c|Ns) \oplus H(P'|D)$ exchanged between the client’s machine and server. Although the attacker will know $D$, $IPs$, and $IP_c$, because the verification messages are created by a one-way hash function and an exclusive-or operation, he cannot find out $P$ and $H(P|D)$. Therefore the proposed protocol is secure against eavesdropping attack without using the SSL.

4.2 Message Modification Attack

Sometimes, an attacker modifies the verification messages between a client and a server to find out the client’s password or to gain unauthorized access. In the proposed protocol, all verification messages are created using a one-way hash function and an exclusive-or operation. Therefore, it is impossible to regenerate valid verification messages by modifying an existing message. Because the attacker cannot log on to the server without using valid verification messages, the proposed protocol is secure against message modification attacks without using the SSL.

4.3 Message Replay Attack

A message replay attack is the act of intercepting the message and retransmitting it. An attacker listens to network communication and intercepts messages, which include verification information. The attacker then tries to log on to the server by replaying the verification messages. Let us assume that an attacker intercepted an ID and verification message, $M_c$. After the attacker sends the ID, he replays the verification message, $M_c$, in an attempt to log on to the server. In order to verify the client, the server computes $H(P|D) \oplus H(H(P|D)|IPhs|Ns)$, and then compares the value with the verification message, $M_c$. Because $M_c$ includes a random nonce, $Ns$, it is a one-time message. Therefore, $M_c$ and $H(P|D) \oplus H(H(P|D)|IPhs|Ns)$ are different to each other when the message replay attack occurs. As a result, the attacker cannot successfully execute a message replay attack. Therefore the proposed protocol is secure against message replay attacks without using the SSL.

4.4 Man-in-the-middle Attack

A man-in-the-middle attack is the act of relaying messages between the victims, making them believe that they are sending messages directly to one another. Before relaying messages, an attacker sometimes forges the messages. When an attacker forwards the verification messages, $M_s$ and $M_c$, exchanged between the client and server in the proposed protocol, because $IP_s$ and $IP_c$ are different from $IPhs$, the attacker cannot succeed with a man-in-the-middle attack. In order to generate valid verification messages by modifying $IP_s$ and $IP_c$, the attacker has to know $P$ or $H(P|D)$. As mentioned in Section 4.1, the attacker cannot establish the values of $P$ and $H(P|D)$ from the verification messages, $M_s$ and $M_c$. Therefore the proposed protocol is secure against man-in-the-middle attacks.
4.5 Malicious Server Attack

In malicious server attacks, an attacker sets up a malicious server and lures clients to register with the server. If clients input their passwords on a fake HTML file, the attacker can read the passwords. In the proposed protocol, when a client registers with a server, he sends the ID and H(P|D) to the server. If the client visits a malicious server impersonating a legitimate server, he sends ID and H(P|D′), where D′ denotes the domain name of the malicious server. An attacker can read the values of ID and H(P|D′), but cannot read H(P|D). Here, H(P|D′) cannot be used in the authentication phase on the legitimate server. Therefore, the proposed protocol is secure against malicious server attacks without using the SSL.

4.6 Phishing Attack

In a phishing attack, an attacker sets up a fake server and sends a large number of spoofed e-mails to random clients. If a recipient of the e-mail attempts to log on to the fake server, their IDs and passwords are exposed to the attacker. In the proposed protocol, when a client visits and tries to log on to a fake server, the client’s machine only sends the ID to the server. The server must then send a verification message to the client. However, the fake server cannot generate the verification message, M_s, because it does not know P or H(P|D). Only if the client’s machine successfully verifies the server using the verification message, M_s, does it send the verification message, M_c, to the server for verification. Therefore, the proposed protocol is secure against phishing attacks without using the SSL.

4.7 Pharming Attack

In a pharming attack, although clients input the correct domain name of the legitimate server in a web browser, they are redirected to a fake server. In the same way as phishing attacks, when they try to log on to the fake server, their IDs and passwords are exposed to the attacker.

In the proposed protocol, when a client tries to log on to the legitimate server, the client’s ID is sent to the malicious server, and the server forwards it to the legitimate server. After that, the legitimate server sends the verification message, M_s, back to the malicious server, and the malicious server forwards it to the client’s machine. The client’s machine computes \( H(P'|D) \oplus H(H(P'|D)||IP_{hs}|Ns) \) and compares it to M_s. Because the value of IP_s that was used to generate M_s is different from IP_{hs}, the authentication phase fails. In order to create valid verification messages by modifying IP_s, the attacker has to know P or H(P|D). As mentioned in Section 4.1, the attacker cannot establish the value of P and H(P|D) from the verification message. Therefore, the proposed protocol is secure against pharming attacks without using the SSL.

5. CONCLUSION

As people spend more social and economic time on the Internet, the importance of protecting their privacy against attacks such as phishing and pharming attacks also in-
creases. Currently, password authentication is the most common means of authentication on the Internet. Until now, although research has been conducted into possible countermeasures to phishing attacks, most of these methods require encryption keys during the authentication phase and are especially vulnerable to pharming attack. The summary of previous work is shown in Table 4.

<table>
<thead>
<tr>
<th>Anti-phishing Requirement</th>
<th>Drawback</th>
</tr>
</thead>
<tbody>
<tr>
<td>Browser extension and tool bar</td>
<td>Black list file</td>
</tr>
<tr>
<td>PwdHash</td>
<td>Hash function</td>
</tr>
<tr>
<td>SPS</td>
<td>Contents filtering</td>
</tr>
<tr>
<td>SpoofGuard</td>
<td>Contents scanning</td>
</tr>
<tr>
<td>SPP</td>
<td>Hash function and exclusive-or operation</td>
</tr>
<tr>
<td>Authentication service by Huang et al.</td>
<td>IM service</td>
</tr>
<tr>
<td>APA</td>
<td>Symmetric key</td>
</tr>
<tr>
<td>APPMA</td>
<td>Asymmetric key</td>
</tr>
</tbody>
</table>

In order to resolve this problem, we proposed a hash-based password authentication protocol. The proposed protocol is secure against a variety of attacks such as replay, man-in-the-middle, phishing, and pharming without using the SSL. Because the proposed authentication protocol does not use encryption keys during the authentication phase, it does not require high computational resources. Therefore the proposed protocol is suitable for wireless and mobile communication systems. The proposed protocol can be broadly adopted to develop secure applications against a variety of attacks in ubiquitous network environments.

REFERENCES


Iksu Kim received the B.S., M.S., and Ph.D. in Computer Science from Soongsil University, South Korea, in 2000, 2002, and 2008, respectively. He worked at SKYCOM as a manager until Jan. 2009. He is a Professor of school of computing at Soongsil University from September 2009. He achieved the CISSP (Certified Information Systems Security Professional) certification in 2008. His research interests include system and network security.
Yongyun Cho received his BE in Computer Science from Inchoen University, Korea, in 1995. And, he received his MS and Ph.D. in Computer Engineering from Soongsil University, Korea, in 1998 and 2006, respectively. Currently, he is a Professor at Information and Communication Engineering, Sunchon University. His research interests include semantic web and web security.