On The Security Of Self-Certified Public Keys *

Cheng-Chi Lee† Min-Shiang Hwang‡ I-En Liao§

Department of Library and Information Science†
Fu Jen Catholic University
510 Jhongjheng Rd., Sinjhuang City,
Taipei County 24205, Taiwan, R.O.C.

Department of Computer Science and Information Engineering‡
Asia University
No. 500, Lioufeng Raod, Wufeng Shiang, Taichung, Taiwan, R.O.C.
Corresponding email: mshwang@asia.edu.tw

Department of Computer Science§
National Chung Hsing University
250 Kuo Kuang Road, 402 Taichung, Taiwan, R.O.C.

May 2, 2011
On The Security Of Self-Certified Public Keys

Abstract

Many cryptosystems have been developed to solve the problem of information security and some of the approaches were based on the self-certified public key proposed by Girault. In Girault’s scheme, the public key is computed cooperatively by both the system authority (SA) and the user. One of the advantages is that the public key is able to implicitly authenticate itself without any additional certificates. Another advantage is that the SA is not able to forge a public key without knowing the user’s secret key. Despite the advantages of Girault’s system, we will prove in this paper that the system still suffers from two weaknesses by an evil user who impersonates the SA and generates a legal signature without knowing the secret key of the SA. Next, we will propose a slight improvement on Girault’s system.

Keyword: Cryptosystem, cryptography, self-certified public key, security, signature.

1 Introduction

Some well-known public key systems have been developed since 1976 [4, 5, 8, 14]. In those systems, each user has two keys, namely, the private key and the public key. The private key is kept secretly by a user, and it is used to provide the legal signature of a message or to decrypt a message sent by another user. The public key is accessible to public through directory lookup, and it is used to verify the validity of a

*This work was supported in part by Taiwan Information Security Center (TWISC), National Science Council under the grant NSC 96-2219-E-009-013 and NSC 99-2221-E-030-022.
signature or to encrypt a message. Since the public key is published to the public key directory, an adversary can modify the public key of a target user from the public key directory. A public-key authentication is an important research’s issue. The purpose of public-key authentication is to verify the public key of a legal user and to prevent public key from being forged.

Three of the most popular schemes for public-key authentication are ID-based scheme [19], certificate-based scheme [10], and self-certified scheme [6]. We briefly review each of them in the following.

In ID-based scheme, a user first chooses his/her own secret key, and then the system authority (SA) generates a public key using the user’s identity and the secret key. Since the public key is derived from the user’s identity, the direct relation between the identity and the public key makes it impossible for an evil user to forge a public key. In addition, there is no need to store the public key in a public directory. However, this scheme has a drawback that the SA can impersonate a user, since SA knows every user’s secret key. In general, public keys are derived from user’s identities and secret keys. For example, the public key is equal to \( s = ID^d \mod n \), where \( ID \) is user’s identity and \( d \) is user’s secret key. This procedure is generated by SA.

In certificate-based scheme, the public key of a user is generated by the SA and is used as the user’s certificate. The process of generating a public key is also known to the public. The difference between ID-based scheme and certificate-based scheme is that the certificate-based scheme has a certificate to verify the public key of a user. The procedure of generating public keys is public. For example, the public is \((y, C)\), where \( y \) is user’s public key and \( C \) is the public key’s certificate. Therefore, one can recalculate a user’s public key and compare it with the one stored in the SA’s system to verify the validity of a public key. These schemes suffers from the same drawback as in the ID-based scheme, namely, the SA is able to impersonate a user by generating
a false certificate. In addition, the certificates have to be stored in SA’s system which may occupy too much storage space.

The self-certified scheme was developed by Girault to overcome the problems of the above two, in which a user first chooses his/her own secret key, and then the public key is computed using both the user’s and SA’s secret keys. That is to say, the public key is generated by both of user and SA. If SA doesn’t know the user’s secret key, SA cannot generate public key. The detail of this procedure can be seen in Section 2. The main feature of this system is that the SA is a trusted parity. The SA is unable to forge a public key. In other words, it makes the SA more trust worthy. Due to such an advantage, this scheme received a lot more attentions than the other schemes did [3, 15, 16, 21, 22]. These schemes also need an SA to help users to sign users’ public keys. The public key is computed by using both of the user’s and SA’s secret keys. Therefore, SA cannot impersonate a user to derive a user’s public key. Using the Girault’s system, these schemes can achieve their proposed requirements.

Despite the advantages of Girault’s system, Saeednia showed that their system is insecure [17]. Saeednia pointed out that the authority SA can know the users’ secret keys if the authority generates modulo $n$ in a special dishonest way. In this paper, we will propose a different cryptanalysis of the system. we assume that the authority is a trusted parity. This paper will show two weaknesses that an evil user who impersonates the SA and generates a legal public key of a user without knowing the secret key of the SA. Next, we will propose a slight improvement on Girault’s system.

The rest of the paper is organized as follows. In the next section, the Girault’s self-certified public key is briefly reviewed. The problem of Girault’s system is described in Section 3. Finally, we give a few concluding remarks and a slight improvement in Section 4.
2 Girault’s Self-Certified Public Key System

Girault proposed a self-certified public key system [6] that is based on the RSA cryptosystem [2, 14] and consists of three phases: the initialization, the registration, and the verification phases.

Initialization Phase:

The SA first generates an RSA key pair \((e, d)\) satisfying \(e \times d \mod (p-1)(q-1) = 1\), where \(p\) and \(q\) are two large primes. Here, \(e\) and \(d\) denote a public key and secret key of the SA, respectively. Then the SA calculates two integers \(n\) and \(g\), where \(n = p \times q\) and \(g\) is a maximal order in the multiplicative group \((\mathbb{Z}/n\mathbb{Z})^*\). After that, the parameters \(p\), \(q\), and \(d\) are kept secret by the SA and \(n\), \(e\), and \(g\) are open to the public.

Registration Phase:

When a user \(U_i\) with identity \(ID_i\), wants to join the system, he/she chooses a secret key \(s_i\) and calculates an integer \(v_i\) in the following:

\[ v_i = g^{-s_i} \mod n. \]  

(1)

Next, \(U_i\) sends \(ID_i\) and \(v_i\) to the SA. Upon receiving these messages, the SA calculates a public key \(p_i\) for the user by

\[ p_i = (v_i - ID_i)^d \mod n, \]  

(2)

and then the SA sends \(p_i\) back to \(U_i\). Upon receiving \(p_i\), \(U_i\) checks the validity of \(p_i\) by

\[ (p_i^e + ID_i) \mod n = v_i. \]  

(3)
If the above equation holds, $U_i$ is certain that $p_i$ is indeed generated by the SA. Note that the public key $p_i$ of a user $U_i$ is generated by SA using both the secret key $s_i$ of $U_i$ and $d$ of SA. However, $s_i$ is unknown to the SA. The registration phase is shown in Figure 1.

$$p_i = (v_i - ID_i)^d \mod n$$

$v_i = g^{-s_i} \mod n$

<table>
<thead>
<tr>
<th>$U_i$</th>
<th>$SA$</th>
</tr>
</thead>
<tbody>
<tr>
<td>chooses a secret key $s_i$</td>
<td>$(v_i, ID_i)$ e: public key, d: secret key</td>
</tr>
<tr>
<td>$v_i = g^{-s_i} \mod n$</td>
<td>$p_i = (v_i - ID_i)^d \mod n$</td>
</tr>
<tr>
<td>checks $p_i$</td>
<td>$p_i$</td>
</tr>
<tr>
<td>$(p_i^e + ID_i) \mod n = v_i$</td>
<td>$p_i$: public key</td>
</tr>
</tbody>
</table>

Figure 1: The Registration Phase of Girault’s Scheme

**Verification Phase:**

When a verifier wants to verify the validity of a user’s $p_i$, he/she can follow identity $ID_i$ [1, 18]:

**Step 1.** $U_i$ sends $ID_i$ and $p_i$ to the verifier, who then calculates $v_i = (p_i^e + ID_i) \mod n$.

**Step 2.** $U_i$ selects a random integer $r_i$, calculates $t_i = g^{r_i} \mod n$, and then sends $t_i$ to the verifier.

**Step 3.** The verifier selects a random integer $r_v$ and sends it to $U_i$.

**Step 4.** $U_i$ calculates $y_i = r_i + s_i \times r_v$, and sends it to the verifier.
Step 5. Upon receiving $y_i$, the verifier checks the following equation $(g^{y_i} \times v_i^{r_v}) \mod n = t_i$. If it holds, the verifier prove that $ID_i$ is valid and $p_i$ was generated by the SA. This phase is shown in Figure 2.

<table>
<thead>
<tr>
<th>$U_i$</th>
<th>Verifier</th>
</tr>
</thead>
<tbody>
<tr>
<td>selects a random number $r_i$</td>
<td>$(p_i, ID_i)$ $v_i = p_i^e + ID_i = g^{-s_i} \mod n$</td>
</tr>
<tr>
<td>$t_i = g^{r_i} \mod n$</td>
<td>$t_i$ selects a random number $r_v$</td>
</tr>
<tr>
<td>$y_i = r_i + s_i r_v$</td>
<td>$y_i$ checks $y_i$ $g^{y_i} \times v_i^{r_v} \mod n = t_i$</td>
</tr>
</tbody>
</table>

Figure 2: The Verification Phase of Girault’s Scheme

Note the name self-certified comes from the fact that no certificate is necessary when verifying $ID_i$ and $p_i$, as the certificate is embedded in the public key itself. An evil user, who has the knowledge of $p_i$, is highly unlikely to obtain $U_i$’s secret key $s_i$ by solving the discrete logarithm problem [6]. Although the SA can impersonate $U_i$ by generating a false public key $p'_i$ with the use of an arbitrary secret key $s'_i$. The existence of two or more public keys linked to $U_i$ can prove that the SA has cheated, as the SA can generate only one valid public key for a particular user.

3 Cryptanalysis of Girault’s System

Despite the advantages of Girault’s self-certified public key system, it is still vulnerable that an evil user who impersonates the SA to sign a legal public key for a user. We propose two weaknesses as follows.
Weakness One:

Step 1. An evil user $U_i$ arbitrarily chooses a random secret key $s'_i$ and calculates $v'_i = g^{-s'_i} \mod n$, which satisfies $v_i + ID_i = v'_i - ID'_i$, where $ID'_i$ is $U_i$’s another identity.

Step 2. $U_i$ registers his/her another $ID'_i$ to SA, and sends $(ID'_i, v'_i)$ to SA.

Step 3. SA calculates a legal public key $p'_i$ for the user $ID'_i$ by $p'_i = (v'_i - ID'_i)^d \mod n$ and sends it to $U_i$.

Step 4. Upon receiving $p'_i$, $U_i$ checks it by Equation (3), who was convinced that $p'_i$ is signed by SA.

Step 5. $U_i$ impersonates SA to sign a legal $p''_i$ for $ID''_i$ as follows, where $p''_i = p_i \times p'_i$ and $ID''_i = ID^2_i$.

\[
\begin{align*}
p''_i &= p_i \times p'_i \mod n \\
&= (v_i - ID_i)^d(v'_i - ID'_i)^d \mod n \\
&= [(v_i - ID_i)(v_i + ID_i)]^d \mod n \\
&= (v^2_i - ID^2_i)^d \mod n.
\end{align*}
\]

The secret key of $ID''_i$ is $2s_i$ which is derived from $v''_i = v^2_i \mod n = g^{-2s_i} \mod n$.

Therefore, $U_i$ can impersonate SA to sign a validity of $p''_i$ and $ID''_i$. But, SA does not know that $ID''_i$ is a forged user.

When a verifier wants to verify the validity of a user’s $p''_i$, he/she can identity $ID''_i$ in the following verification phase:

Step 1. $U_i$ sends $ID''_i$ and $p''_i$ to the verifier, who then calculates $v''_i = (p''_i + ID''_i) \mod n$. 

7
Step 2. $U_i$ selects a random integer $r_i$, calculates $t_i = g^{r_i} \mod n$, and then sends $t_i$ to the verifier.

Step 3. The verifier selects a random integer $r_v$ and sends it to $U_i$.

Step 4. $U_i$ calculates $y_i = r_i + s''_i \times r_v$, where $s''_i = 2s_i$, and sends $y_i$ to the verifier.

Step 5. Upon receiving $y_i$, the verifier checks the following equation: $g^{y_i} \times v''_i r_v \mod n = t_i$. If it holds, the verifier can prove that $ID''_i$ is valid and $p''_i$ was generated by the SA.

It is quite obvious that an evil user $U_i$ can impersonate SA to sign a validity of $p''_i$ and $ID''_i$. It also passes the public-key authentication by the above protocol. That is to say, it is possible that this weakness can be performed if a user $U_i$ can find a correct format $ID'_i$ and $ID''_i$ such that $v_i + ID_i = v'_i - ID'_i$ and $ID''_i = ID'_i^2$.

**Weakness Two:**

Step 1. An evil user $U_i$ arbitrarily chooses a random secret key $s'_i$ and a public key $p'_i$.

Step 2. $U_i$ calculates $v'_i = g^{-s'_i} \mod n$.

Step 3. $U_i$ derives $ID'_i$ by $(p'_i)^e + ID'_i) \mod n = v'_i$.

Step 4. $U_i$ keeps the triplet $(ID'_i, s'_i, p'_i)$, where the $ID'_i$ is $U_i$’s another identity, and $(s'_i, p'_i)$ is his/her another key pair. Therefore, $U_i$ can impersonate SA to sign a validity of $p'_i$ and $ID'_i$, SA is unknown to that $ID'_i$ is a forged user.

When a verifier wants to verify the validity of a user’s $p'_i$, he/she can identity $ID'_i$ in the following verification phase:

Step 1. $U_i$ sends $ID'_i$ and $p'_i$ to the verifier, who then calculates $v'_i = (p'_i)^e + ID'_i) \mod n$. 

8
**Step 2.** \( U_i \) selects a random integer \( r_i \), calculates \( t_i = g^{r_i} \mod n \), and then sends \( t_i \) to the verifier.

**Step 3.** The verifier selects a random integer \( r_v \) and sends it to \( U_i \).

**Step 4.** \( U_i \) calculates \( y_i = r_i + s'_i \times r_v \), and sends \( y_i \) to the verifier.

**Step 5.** Upon receiving \( y_i \), the verifier checks the following equation: \( g^{y_i} \times v'_i^{r_v} \mod n = t_i \). If it holds, the verifier can prove that \( ID'_i \) is valid and \( p'_i \) was generated by the SA.

It can prove that an evil user \( U_i \) can impersonate SA to sign a validity of \( p'_i \) and \( ID'_i \). Therefore, the self-certified public key system is vulnerable. That is to say, it is possible that this weakness can be performed if a user \( U_i \) can find a correct format \( ID'_i \) such that \((p'_i + ID'_i) \mod n = v'_i\).

### 4 Discussions and Conclusions

We have shown that Girault’s self-certified public key system is possible that it has two weaknesses: An evil user can easily impersonate the SA to sign a public key without knowing the secret key of the SA. We can see that an evil user can easily generate the valid pair of \((ID_i, p_i)\) in which they can be only generated by the SA.

To overcome these weaknesses, we proposed a slight improvement on Girault’s system. The proposed slight improvement is shown in Figure 3 and 4. Our proposed improvement uses the concept of one-way hash function. This function, \( h : x \rightarrow y \), has the following properties [7, 9, 11, 12]:

1. The function \( h \) can take a message of arbitrary-length input and produce a message digest of a fixed-length output.
2. The function $h$ is one-way, given $x$, it is easy to compute $h(x) = y$. However, given $y$, it is hard to compute $h^{-1}(y) = x$.

3. The function $h$, given $x$, it is computationally infeasible to find $x' \neq x$ such that $h(x') = h(x)$.

4. The function $h$, it is computationally infeasible to find any two pair $x$ and $x'$ such that $x' \neq x$ and $h(x') = h(x)$.

The difference between the Girault’s system and our improvement is that we only hash the user’s identity ($ID_i$). The proposed improvement does not only achieve their advantages but also enhances their security by withstanding the security weaknesses. Of course, our improvement can apply to other self-certified based public key cryptosystems [13, 20].

<table>
<thead>
<tr>
<th>$U_i$</th>
<th>$SA$</th>
</tr>
</thead>
<tbody>
<tr>
<td>chooses a secret key $s_i$</td>
<td></td>
</tr>
<tr>
<td>$v_i = g^{-s_i} \mod n$</td>
<td></td>
</tr>
<tr>
<td>checks $p_i$</td>
<td></td>
</tr>
<tr>
<td>$(p_i^e + h(ID_i)) \mod n =?v_i$</td>
<td></td>
</tr>
<tr>
<td>$p_i$: public key</td>
<td></td>
</tr>
<tr>
<td>$e$: public key, $d$: secret key</td>
<td></td>
</tr>
<tr>
<td>$p_i = (v_i - h(ID_i))^d \mod n$</td>
<td>$p_i$</td>
</tr>
</tbody>
</table>

Figure 3: Our Improved Registration Phase

References

<table>
<thead>
<tr>
<th>$U_i$</th>
<th>Verifier</th>
</tr>
</thead>
<tbody>
<tr>
<td>selects a random number $r_i$</td>
<td>$(p_i, ID_i)$ $v_i = p_i^e + h(ID_i) = g^{-s_i} \mod n$</td>
</tr>
<tr>
<td>$t_i = g^{r_i} \mod n$</td>
<td>$t_i$ selects a random number $r_v$</td>
</tr>
<tr>
<td>$y_i = r_i + s_i r_v$</td>
<td>$r_v$</td>
</tr>
<tr>
<td></td>
<td>$y_i$ checks $y_i$</td>
</tr>
<tr>
<td></td>
<td>$g^{y_i} \times v_i^{r_v} \mod n = t_i$</td>
</tr>
</tbody>
</table>

Figure 4: Our Improved Verification Phase


