Security Analysis of Practical Anonymous User Authentication Scheme with Security Proof

1Chenglian Liu, 1Changlu Lin and 1Shuliang Sun
1Fuqing Branch of Fujian Normal University, Fuqing, 350300, China
2Fujian Normal University, Fuzhou, 350007, China

Abstract: Chien proposed a practical anonymous user authentication scheme with security proof in 2008. Even he used bitwise exclusive-or operation in cryptographic protocol. But the designer does not mention it may a very dangerous if they misused this function. Chien (2008) proposed a practical anonymous user authentication scheme with security proof in 2008. However, Chien might also misuse mathematical precedence properties in computer programming which logical exclusive-or combine string concatenation operation for their scheme. A sample of paper (Zhang and Wang, 2005; Xu et al., 2010) where it also misused same situation actually.

Key words: Digital signature, bitwise exclusive-or operation, mathematical precedence

INTRODUCTION

Most bitwise exclusive-or operation used in cryptographic protocol. But the designer does not mention it may a very dangerous if they misused this function. Chien (2008) proposed a practical anonymous user authentication scheme with security proof in 2008. However, Chien might also misuse mathematical precedence properties in computer programming which logical exclusive-or combine string concatenation operation for their scheme. A sample of paper (Zhang and Wang, 2005; Xu et al., 2010) where it also misused same situation actually.

Briefly Chien’s Scheme

In Chien (2008) review some hard problems and then propose a practical anonymous user authentication scheme with security. The notation and definitions are same in review of Chien (2008).

In this commonly believed that there is no polynomial-time algorithm to solve FACP, DLP N, CDHP N or DDHP N with non-negligible probability (Girault, 1991). Based on the FAC problem and the CDHP N problem, we propose the two-party key agreement scheme with client anonymity as follows. Our proposed scheme consists of two phases: the key generation phase and the anonymous user identification phase.

Key generation: The SCPC chooses two large safe prime p and q computes:

\[ N = pq \]  \hspace{1cm} (1)

selects e and d computes such that:

\[ ed = 1 \pmod{\phi(N)} \]  \hspace{1cm} (2)

Where:

\[ \phi(N) = (p-1)(q-1) \]  \hspace{1cm} (3)

It chooses a generator g which is a primitive root for both \( z_p \) and \( z_q \), a symmetric-key cryptosystem \( E(\cdot) \) (such as AES), three cryptographic one-way hash functions:

\[ H_1: \{0, 1\}^* \rightarrow Z_p \quad H_2: \{0, 1\}^* \rightarrow \{0, 1\}^* \]

and:

\[ H_3: \{0, 1\}^* \rightarrow \{0, 1\} \]

where, w is a public parameter such that \( w > \phi(N) \) and l is the key length for the symmetric encryption scheme. The encryption function and the decryption function under the secret key K are denoted as \( E_k(\cdot) \) and \( D_k(\cdot) \), respectively. This encryption scheme should satisfy the indistinguishability under chosen plain text attack (IND-CPA) property. The SCPC then publishes parameters \( \{N, e, g, w, E(\cdot), D(\cdot), H_1, H_2, H_3\} \) as public parameters and keeps \( \{d, p\} \) and \( \{q\} \) private. Finally, through a secure channel, the SCPC sends to each registered entity U_i (which could be a client with identity C_i or a server with identity P_i) a secret token:

\[ S_{C_i} = H_3(H_i(C_i)) \]  \hspace{1cm} (4)

Where:

\[ H_3 = H_3(C_i) \]  \hspace{1cm} (5)

Corresponding Author: Chenglian Liu, Department of Mathematics and Computer Science,
Fuqing Branch of Fujian Normal University, Fuqing, 350300, China
If \( U_i \) is a client \( C_i \) or, a secret token:
\[
S_{r_i} = H_{r_i}(\text{mod } N) \quad (6)
\]

Where:
\[
H_{r_i} = H_{r_i}(P_i) \quad (7)
\]

If \( U_i \) is a server \( P_i \), In the following, \( sid \) denotes the unique session identifier of the current session.

**Anonymous user identification:** To request a service from the server \( P_i \), the client \( C_i \) sends the request to \( P_i \). Upon receiving the request, \( P_i \) chooses a random number \( k \), compute:
\[
z = g^k S_{r_i} \text{ (mod } N) \quad (8)
\]
and sends \( z \) to \( C_i \). After receiving \( z \), \( C_i \) chooses a random number \( t \) and computes the following values:
\[
m = ze/H_{r_i} \text{ (mod } N) \quad (9)
\]
\[
r = m^t \text{ (mod } N) \quad (10)
\]
\[
k_{res} = H_j \quad (11)
\]
\[
x = g^s \text{ (mod } N) \quad (12)
\]
\[
s = S_{C_i}^{(t \times x^t \text{ mod } N)} \text{ (mod } N) \quad (13)
\]
\[
y = E_{k_{res}}(C_i \text{ or } s \text{ or } r) \quad (14)
\]
where, \( T \) is the current timestamp. \( C_i \) then sends \( (x, y, T) \) to \( P_i \). Upon receiving the message, \( P_i \) first checks the validity of timestamp \( T \) by checking whether the timestamp is fresh and is within valid time window (Fig. 1). If so, \( P_i \) further computes:
\[
r = x^k \text{ (mod } N) \quad (15)
\]
and:
\[
k_{res} = H_j(r) \quad (16)
\]
Using the key, decrypts to derive:
\[
C_i || s \leftarrow C_i \text{ or } s \text{ or } C_i \text{ or } S_{k_{res}}(y) \quad (17)
\]
computes:
\[
H_{C_i} = H_{C_i}(C_i) \quad (18)
\]

**Fig. 1:** Chien’s Scheme (Chien, 2008)

and verifies whether the equation:
\[
H_{C_i}^{(t \times x^t \text{ mod } N)} \equiv s' \text{ (mod } N) \quad (19)
\]

If all the verifications succeed, then the request is granted; otherwise, the request is rejected. Finally, the client and the server share one common session key and the arrangement of:
\[
E_{k_{res}}(C_i \text{ or } s \text{ or } r) \quad (20)
\]
is to ensure the indistinguishability in the formal model. It is easy to show the correctness of the verification equation:
\[
H_{C_i}^{(t \times x^t \text{ mod } N)} \equiv s' \text{ (mod } N) \quad (21)
\]
as follows:
\[
s' = S_{C_i}^{(t \times x^t \text{ mod } N)}
= H_{C_i}^{(t \times x^t \text{ mod } N)}
= H_{C_i}^{(t \times x^t \text{ mod } N)} \text{ (mod } N) \quad (22)
\]

**OUR SECURITY ANALYSIS**

Here, we introduce two point views, one is logical bitwise exclusive-or operation, the other one is order of operations in computer system. We introduce the precedence properties in some programming languages.

**Order of operation:** For example, the bitwise exclusive-or is in level 9. However, there is no string concatenation operator in C/C++ directly. Although, the C/C++ provides some libraries such as strcat(), sprintf() and so on functions to connect string but it still can not combine strings with numbers (Table 1).

For Java programming language, the string concatenation is in level 5 and the bitwise exclusive-or is in level 10 (Table 2). The string concatenation therefore is
higher precedence than bitwise exclusive-or as known. The Chien’s scheme does not appear to be true, as pointed out below. According to Sedgewick and Wayne (2008) and Kruse and Ryba (1999), the string concatenate has higher precedence than bitwise exclusive-or. Therefore the string concatenate should be applied first and then to process bitwise exclusive-or in most parts of computer programming such as Java, Visual Basic (Microsoft, 2011) and so on, this rule is known as a precedence rule or order of operation.

<table>
<thead>
<tr>
<th>Level</th>
<th>Operator</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>[ ]</td>
<td>Access array element</td>
</tr>
<tr>
<td>2</td>
<td>.</td>
<td>Access object number</td>
</tr>
<tr>
<td>3</td>
<td>?</td>
<td>Invoke a method</td>
</tr>
<tr>
<td>4</td>
<td>++</td>
<td>Post-increment</td>
</tr>
<tr>
<td>5</td>
<td>-</td>
<td>Post-decrement</td>
</tr>
<tr>
<td>6</td>
<td>++</td>
<td>Pre-increment</td>
</tr>
<tr>
<td>7</td>
<td>+</td>
<td>Pre-decrement</td>
</tr>
<tr>
<td>8</td>
<td>+</td>
<td>Unary plus</td>
</tr>
<tr>
<td>9</td>
<td>-</td>
<td>Unary minus</td>
</tr>
<tr>
<td>10</td>
<td>!</td>
<td>Logical NOT</td>
</tr>
<tr>
<td>11</td>
<td>~</td>
<td>Bitwise NOT</td>
</tr>
<tr>
<td>12</td>
<td>New</td>
<td>Cast</td>
</tr>
<tr>
<td>13</td>
<td>*</td>
<td>Object creation</td>
</tr>
<tr>
<td>14</td>
<td>/</td>
<td>Multiplicative</td>
</tr>
<tr>
<td>15</td>
<td>%</td>
<td>Additive</td>
</tr>
<tr>
<td>16</td>
<td>&gt;&gt;</td>
<td>String concatenation</td>
</tr>
<tr>
<td>17</td>
<td>&gt;&gt;&gt;</td>
<td>Shift</td>
</tr>
<tr>
<td>18</td>
<td>&lt;=</td>
<td>Relational type comparison</td>
</tr>
<tr>
<td>19</td>
<td>==</td>
<td>Equality</td>
</tr>
<tr>
<td>20</td>
<td>&amp;</td>
<td>Bitwise AND</td>
</tr>
<tr>
<td>21</td>
<td>^</td>
<td>Bitwise XOR</td>
</tr>
<tr>
<td>22</td>
<td></td>
<td></td>
</tr>
<tr>
<td>23</td>
<td>&amp;&amp;</td>
<td>Conditional AND</td>
</tr>
<tr>
<td>24</td>
<td></td>
<td></td>
</tr>
<tr>
<td>25</td>
<td>=</td>
<td>Assignment</td>
</tr>
</tbody>
</table>

Our attack: The attack (Fig. 2) can forge a valid parameter (r, s) where \( r \oplus s = \langle r \oplus s \rangle \) (Liu et al., 2012). He does follow stops:

- **Step 1**: Attack sets \( r' = r \)
- **Step 2**: Attack sets \( s' = -s \)
- **Step 3**: Attack sets \( C_i' = -C_i \)

CONCLUSION

In general, the string concatenate operator is always higher than bitwise logical exclusive-or operation and if a designer or developer misused or misunderstood this situation, it may cause a dangerous problem. We clearly described an example of this case in the paper. On the other hand, user used XOR operation in two’s complement number system, it may cause others dangerous problem. Thus, the Chien’s scheme is insecure. From previous section, we confirm our assumption.
ACKNOWLEDGMENTS

This study is partially supported by the National Natural Science Foundation of China under Grant No. 61103247 and the Natural Science Foundation of Fujian Province under Grant No. 2011J05147.

REFERENCES