A Note on “A Time-Sensitive Token-Based Anonymous Authentication and Dynamic Group Key Agreement Scheme for Industry 5.0”

Zhengjun Cao\textsuperscript{1}, Lihua Liu\textsuperscript{2}

\textbf{Abstract.} We show that the Xu et al.’s authentication and key agreement scheme [IEEE Trans. Ind. Informatics, 18(10), 7118-7127, 2022] is flawed. (1) It confused some operations for bilinear maps and presented some inconsistent computations. (2) It failed to keep anonymity, not as claimed. The adversary can use any device’s public key stored in the blockchain to test some verification equations so as to reveal the identity of a target device.

\textbf{Keywords:} Key agreement, anonymity, authentication, blockchain.

\section{Introduction}

Recently, Xu et al. [1] have presented an anonymous authentication and dynamic group key agreement scheme for industry 5.0. It is designed to meet many security requirements, such as anonymity and untraceability, session key establishment, forward and backward secrecy, resistance to replay attack, impersonation attack, etc. In this note, we show that the scheme has some inconsistent computations and fails to keep anonymity, not as claimed.

\section{Review of the Xu et al.’s scheme}

In the proposed scenario, there are two main kinds of entities, device (\textit{DE}) and private key generator (\textit{PKG}). The \textit{DE}s are general nodes, and have mobile capabilities. Each \textit{PKG} is similar to a group controller responsible for key generation, distribution, management, and group communication tasks. Each group is dynamic, which means that \textit{DE} may join or leave a group at any time. The scheme consists of seven phases: initialization, registration, authentication without token, authentication with token, group key generation, \textit{DE} join, and \textit{DE} leave.

\textit{Initialization.} The system administrator picks a cyclic additive group $\mathcal{G}_1$ with a generator $Q$ and a cyclic multiplicative group $\mathcal{G}_2$. Both are of the prime order $p$. Select a bilinear map $\hat{e}: \mathcal{G}_1 \times \mathcal{G}_1 \rightarrow \mathcal{G}_2$ and a private key $s$, and set the public key as $P_{pub} = sQ$. Pick two random numbers $n_1, n_2$, and a unique identity $IDP_j$ for each $PKG_j$. Store $\{s, n_1, n_2, IDP_j\}$ in the memory of $PKG_j$. Publish $\{p, \mathcal{G}_1, \mathcal{G}_2, Q, e, P_{pub}, h(\cdot), E_k, D_k\}$. See Table I for descriptions of involved notations. For

\textsuperscript{1}Department of Mathematics, Shanghai University, Shanghai, 200444, China
\textsuperscript{2}Department of Mathematics, Shanghai Maritime University, Shanghai, 201306, China. liuhl@shmtu.edu.cn
Table I: Notations

<table>
<thead>
<tr>
<th>Symbol</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>TID&lt;sub&gt;i&lt;/sub&gt;</td>
<td>Temporary identity of the DE&lt;sub&gt;i&lt;/sub&gt;</td>
</tr>
<tr>
<td>IDD&lt;sub&gt;i&lt;/sub&gt;</td>
<td>The identity of DE&lt;sub&gt;i&lt;/sub&gt;</td>
</tr>
<tr>
<td>IDP&lt;sub&gt;j&lt;/sub&gt;</td>
<td>The identity of PKG&lt;sub&gt;j&lt;/sub&gt;</td>
</tr>
<tr>
<td>GID&lt;sub&gt;k&lt;/sub&gt;</td>
<td>The identity of kth group</td>
</tr>
<tr>
<td>s, P&lt;sub&gt;pub&lt;/sub&gt;</td>
<td>Private key and public key of all PKGs</td>
</tr>
<tr>
<td>S&lt;sub&gt;i&lt;/sub&gt;, a&lt;sub&gt;i&lt;/sub&gt;, b&lt;sub&gt;i&lt;/sub&gt;</td>
<td>The DE&lt;sub&gt;i&lt;/sub&gt;’s private key</td>
</tr>
<tr>
<td>W&lt;sub&gt;i&lt;/sub&gt;, A&lt;sub&gt;i&lt;/sub&gt;, B&lt;sub&gt;i&lt;/sub&gt;</td>
<td>The DE&lt;sub&gt;i&lt;/sub&gt;’s public key</td>
</tr>
<tr>
<td>ST&lt;sub&gt;i&lt;/sub&gt;, ET&lt;sub&gt;i&lt;/sub&gt;</td>
<td>The authorized time slot range [ST&lt;sub&gt;i&lt;/sub&gt;, ET&lt;sub&gt;i&lt;/sub&gt;]</td>
</tr>
<tr>
<td>E&lt;sub&gt;k&lt;/sub&gt;, D&lt;sub&gt;k&lt;/sub&gt;</td>
<td>Symmetric encryption/decryption with key k</td>
</tr>
<tr>
<td>⊕</td>
<td>Bitwise XOR operation</td>
</tr>
<tr>
<td>(a, b)</td>
<td>Concatenation of data a and data b</td>
</tr>
<tr>
<td>h(·)</td>
<td>A hash function h(·) : {0, 1}* → {0, 1}</td>
</tr>
<tr>
<td>h&lt;sup&gt;a&lt;/sup&gt;(b)</td>
<td>Perform a + 1 hash operations on b</td>
</tr>
</tbody>
</table>

convenience, we now only depict the registration phase and authentication without token phase as follows (see Table II).

3 Inconsistent computations

Pairings in elliptic curve cryptography are functions which map a pair of elliptic curve points to an element of the multiplicative group of a finite field. Let n be a positive integer. Let G<sub>1</sub> and G<sub>2</sub> be Abelian groups written in additive notation. Suppose that G<sub>1</sub> and G<sub>2</sub> have exponent n (i.e., [n]P = 0 for all P ∈ G<sub>1</sub>, G<sub>2</sub>). Suppose G<sub>3</sub> is a cyclic group of order n written in multiplicative notation. A pairing is a function \( \hat{e} : G_1 \times G_2 \rightarrow G_3 \) satisfying:

**Bilinearity.** For all P, P′ ∈ G<sub>1</sub> and all Q, Q′ ∈ G<sub>2</sub> we have \( \hat{e}(P + P′, Q) = \hat{e}(P, Q)\hat{e}(P′, Q) \) and \( \hat{e}(P, Q + Q′) = \hat{e}(P, Q)\hat{e}(P, Q′) \).

**Non-degeneracy.** For all P ∈ G<sub>1</sub>, with P ≠ 0, there is some Q ∈ G<sub>2</sub> such that \( \hat{e}(P, Q) \neq 1 \). For all Q ∈ G<sub>2</sub>, with Q ≠ 0, there is some P ∈ G<sub>1</sub> such that \( \hat{e}(P, Q) \neq 1 \).

To this day, the two practical examples of pairings are the Weil and Tate pairings on elliptic curves over finite fields [2]. Both use a non-rational homomorphism \( \phi : G_2 \rightarrow G_1 \) to construct the so-called self-pairing \( e : G_1 \times G_1 \rightarrow G_3 \). In view of this fact, we find the Xu et al.’s scheme have confused the basic operations for bilinear maps and presented some inconsistent computations. It wrongly specifies that

For points \((x, y)\) belonging to G<sub>1</sub> or G<sub>2</sub>, we only focus on x. For example, for Q(x<sub>Q</sub>, y<sub>Q</sub>) and a private key s′, we can obtain \((x_{s'Q}, y_{s'Q})\) by point multiplication operation s′Q, and the corresponding public key P<sub>pub</sub> is x<sub>s'Q</sub>.
Table II: The Xu et al.’s authentication and key agreement scheme

<table>
<thead>
<tr>
<th>(DE_i)</th>
<th>Registration</th>
<th>(PKG_j: {s, n_1, n_2, IDP_j})</th>
</tr>
</thead>
<tbody>
<tr>
<td>Send the join request.</td>
<td>(\xrightarrow{\text{request}} IDD_i, W_i, S_i).</td>
<td>Pick a unique identity (IDD_i).</td>
</tr>
<tr>
<td>Store ({IDD_i, W_i, S_i}).</td>
<td>(\xleftrightarrow{\text{secure channel}} IDD_i, W_i, S_i)</td>
<td>Compute (W_i = h(\text{IDD}_i), S_i = sW_i).</td>
</tr>
<tr>
<td>(DE_i: {IDD_i, W_i, S_i})</td>
<td>Authentication</td>
<td>Create a new block containing ({IDD_i, W_i}), and link it to the Blockchain.</td>
</tr>
</tbody>
</table>

Pick random \(a_i, b_i\) and group identity \(GID_k\).  
Set a timestamp \(T_1\) and time-slot \(ST_i, ET_i\).  
Compute \(A_i = a_iQ, B_i = b_iQ, TK = b_iP_{pub}\),  
\(DNT_1 \leftarrow E_{TK}(IDD_i, ST_i, ET_i, A_i)\),  
\(DNT_2 = h(DNT_1, B_i, T_i, GID_k)S_i\).  
\(\xrightarrow{\text{open channel}} DNT_1, DNT_2, B_i, GID_k, T_i\)  
\(\xleftarrow{\text{open channel}} DNT_2, DNT_3, T_2\).  
Check the timestamp \(T_2\).  
Then check \(e(Q, DNT_2) = e(P_{pub}, h(DNT_2, T_2)W_i)\).  
If so, \((TID, SA_i, TS_{SA}, TS_b) \leftarrow E_{TK}(DNT_3)\).  
Store \((TID, SA_i, TS_{SA}, TS_b, A_i)\).  

It also wrongly formulates that:

\[
W_i = h(\text{IDD}_i), \quad S_i = sW_i, \quad DNT_4 = h(DNT_3, T_2)S_i
\]

\[
DNT_2 = h(DNT_1, B_i, T_i, GID_k)S_i;
\]

\[
e(Q, DNT_2) = e(P_{pub}, h(DNT_1, B_i, T_1, GID_k)W_i);
\]

\[
e(Q, DNT_4) = e(P_{pub}, h(DNT_3, T_2)W_i).
\]

Clearly, \(W_i = h(\text{IDD}_i)\) is not a point over the underlying elliptic curve. So do \(DNT_2, DNT_4\). Thus, the computations

\[
e(Q, DNT_2) = e(P_{pub}, h(DNT_1, B_i, T_1, GID_k)W_i),
\]

\[
e(Q, DNT_3) = e(P_{pub}, h(DNT_3, T_2)W_i)
\]

make no sense. Likewise, the following computations

\[
e(xQ, DNT_2) = e(xP_{pub}, h(DNT_1, B_i, T_1, GID_k)W_i),
\]

\[
e(xQ, DNT_4) = e(xP_{pub}, h(DNT_3, T_2)W_i)
\]

make no sense, too.
One should remove the above wrong specification and formulate that \( W_i = h(IDD_i)Q \) i.e.,
converting \( W_i \) into a point over the underlying elliptic curve. In this case, all

\[
DNT_2, h(DNT_1, B_i, T_1, GID_k)W_i, DNT_4, h(DNT_3, T_2)W_i
\]

are compatible with the bilinear map.

4 The loss of anonymity

Anonymity is a security requirement adopted by many protocols. As for this property, it argues that
(page 7124, [1]):

Among the messages sent during the authentication without token phase and authenti-
cation with token phase, only \( DNT_1, DWT_3, \) and \( HDE_i \) contain the \( IDD_i \) information.
However, \( IDD_i \) in \( DNT_1, DWT_3 \) and \( HDE_i \) is protected by \( h() \). In addition, if an adversary
wants to get \( IDD_i \) from \( DNT_1 \), he/she must get the \( TK \) key. However, according to
the computational Diffie-Hellman (CDH) problem, the adversary cannot obtain \( TK \) from
\( P_{pub}, B_i, \) or \( Q \) in polynomial time.

The argument is not sound. In fact, the legitimate \( PKG_j \) needs to decrypt \( DNT_1 \) to re-
trieve the identity \( IDD_i \), and then uses it to get the target public key \( W_i \) from the blockchain.
Though an adversary cannot decrypt the ciphertext, he can access the set \( \mathcal{Y} = \{(IDD_i, W_i)\}_{1 \leq i \leq n} \),
which is stored in the blockchain. The adversary who has captured \( \{DNT_1, DNT_2, B_i, T_1, GID_k\} \) or
\( \{DNT_3, DNT_4, T_2\} \) via open channels, can test the equation

\[
e(Q, DNT_2) = e(P_{pub}, h(DNT_1, B_i, T_1, GID_k)\chi),
\]

or

\[
e(Q, DNT_4) = e(P_{pub}, h(DNT_3, T_2)\chi), \quad (\rho, \chi) \in \mathcal{Y}
\]

Practically, the size of \( \mathcal{Y} \) is moderate and the success probability of such testings is not negligible.
Once such a public key \( \chi \) is searched out, the adversary can reveal the target identity. To achieve
true anonymity, we think, one should adopt other techniques.

5 Conclusion

We show that the Xu et al.’s key agreement scheme is flawed. We hope the findings in this note
could be helpful for the future work on designing such schemes.

References

Key Agreement Scheme for Industry 5.0, IEEE Trans. Ind. Informatics, 18(10): 7118-7127
(2022)