Insecurity of a provably secure and lightweight certificateless signature scheme for IIoT environments

Lunzhi Deng[a]

[a]School of Mathematical Sciences, Guizhou Normal University, Guiyang 550001, China.

Abstract

Recently, Karati et al. presented a lightweight certificateless signature scheme for industrial Internet of Things (IIoT) environments, and claimed the scheme was provably secure in the standard model. In this paper, it is indicated that the scheme is not secure by showing two concrete attacks.

Keywords: Certificateless cryptography, Signature, Bilinear pairing, Standard model, Internet of Things

1. Introduction

In this paper, we first review Karati et al's scheme [1], then give some comments on it. Through presenting two concrete attacks, it is demonstrated that the scheme [1] cannot provide unforgeability.

2. Review Karati et al.'s scheme

Karati et al.'s scheme [1] is specified by the following algorithms:

- Setup: KGC generates two groups G_1 and G_2 with prime order p, a bilinear pairing $e : G_1 \times G_1 \to G_2$. then picks a generators g_1 of G_1, and a secure hash functions $H_1 : \{0,1\}^* \to Z_p^*$, and a value $y \in Z_p^*$ as master private key. Next, KGC computes $g_2 = e(g_1, g_1)^y$ and $Y_{KGC} = g_1^y$. KGC publish the public parameters: $params = \{G_1, G_2, p, e, g_1, g_2, Y_{KGC}, H\}$ and keeps $msk = \{y\}$ safety.
- Set-Partial-Private-Key: For a user with identity ID_i, KGC picks at random $r_i \in Z_p^*$, computes $R_i = g_1^{r_i}$, $h_i = H(ID_i)$ and $y_i = g_1^{\frac{yh_i}{h_i + r_i + y}}$, then sends $D_i = (y_i, R_i)$ to the user.
- Set-Secret-Value: The user ID_i randomly chooses $x_i, c_i \in Z_p^*$ and sets secret value $SK_i = (c_i, x_i, R_i)$.
- Sets-Public-Key: The user ID_i set the public key $Y_i = (Y_{i1} = y_i^{\frac{1}{x_i}}, Y_{i2} = g_2^{c_i})$.
- CLS-Sign: Given a message $m_i \in Z_p^*$, the user ID_s performs the following steps:
  1. Chooses at random $t_i \in Z_q^*$, computes $h_s = H(ID_s)$, $\sigma_1 = g_2^t$, $\sigma_2 = (g_1^{h_s} \cdot R_s \cdot Y_{KGC})^{(\frac{c_s}{m} - t)x_s}$.

2. Outputs $\sigma = (\sigma_1, \sigma_2)$ as the signature.

- Verify: To verify a signature $(params, ID_s, Y_s, m, \sigma = (\sigma_1, \sigma_2))$, the verifier performs the following steps:
  1. Computes $h_s = H_1(ID_s)$.
  2. Checks whether $\left(\frac{Y_{s2}^{\frac{1}{m}}}{\sigma_1}\right)^{h_s} = e(Y_{s1}, \sigma_2)$. If the equation holds, outputs VALID. Otherwise, outputs INVALID.

3. Forgery attack

Karati et al. claimed that their scheme is unforgeable. However, we show two concrete attacks to refute their claim in this section.

Case 1. A Type adversary $\mathscr{A}_1$ can forge a new signature σ' on the same message£ after he obtains a valid signature σ for the message m.

- Setup. $\mathscr{C}$ sets the msk and $params = \{G_1, G_2, p, e, g_1, g_2, Y_{KGC}, H\}$, then sends the $params$ to $\mathscr{A}_1$ and keeps the msk secret.
- Query. $\mathscr{A}_1$ issues three queries as follows.
  1. $\mathscr{A}_1$ issues a public key query for a user ID_s, and gets the value $Y_s = (Y_{s1} = y_s^{\frac{1}{x_s}}, Y_{s2} = g_2^{c_s})$.
  2. $\mathscr{A}_1$ issues a signature query for the tuple (m, ID_s, Y_s), and gets a signature $\sigma = (\sigma_1, \sigma_2)$, where $\sigma_1 = g_2^t$, $\sigma_2 = (g_1^{h_s} \cdot R_s \cdot Y_{KGC})^{(\frac{c_s}{m} - t)x_s}$.
  3. $\mathscr{A}_1$ picks a value $u \in Z_p^*$, computes $Y'_{s1} = Y_{s1}^{\frac{1}{u}}$, sets $Y'_{s2} = Y_{s2}$, then issues a public key replacement request for ID_s with a new value $Y'_s = (Y'_{s1}, Y'_{s2})$,
- Forge. $\mathscr{A}_1$ sets $\sigma'_1 = \sigma_1$, computes $\sigma'_2 = \sigma_2^u$, and outputs a forged signature $\sigma' = (\sigma'_1, \sigma'_2)$ on a tuple (ID_s, Y'_s, m).

The tuple $(params, ID_s, Y'_s, \sigma' = (\sigma'_1, \sigma'_2), m)$ is a valid signature due to the verification equation is satisfied.

$$
\begin{aligned}
e(Y'_{s1}, \sigma'_2) &= e\left(g_1^{\frac{yh_s}{(h_s + r_s + y)ux_s}}, (g_1^{h_s} \cdot R_s \cdot Y_{KGC})^{(\frac{c_s}{m} - t)ux_s}\right) \\
&= e\left(g_1^{\frac{yh_s}{(h_s + r_s + y)ux_s}}, g_1^{(h_s + r_s + y)}\right)^{(\frac{c_s}{m} - t)ux_s} \\
&= e(g_1, g_1)^{y(\frac{c_s}{m} - t)h_s} \\
&= \left(\frac{g_2^{\frac{c_s}{m}}}{g_2^t}\right)^{h_s} \\
&= \left(\frac{Y'^{\frac{1}{m}}_{s2}}{\sigma'_1}\right)^{h_s}
\end{aligned}
$$

In this attack, it is required for $\mathscr{A}_1$ to replace the public key of the user ID_s. However, $\mathscr{A}_1$ does not know the new secret value of the user ID_s. $\mathscr{A}_1$ uses the difference between the new public key and the old public key to forge a signature.

Case 2. An ordinary adversary $\mathscr{A}$ can give a forged signature on a new message m' even if he has no knowledge for the partial private key and secret value of any user.

- Setup. $\mathscr{C}$ sets the msk and $params = \{G_1, G_2, p, e, g_1, g_2, Y_{KGC}, H\}$, then sends the $params$ to $\mathscr{A}$ and keeps the msk secret.
- Query. $\mathscr{A}$ issues two queries as follows.
  1. $\mathscr{A}$ issues a public key query for a user ID_s, and gets the value $Y_s = (Y_{s1} = y_s^{\frac{1}{x_s}}, Y_{s2} = g_2^{c_s})$.
  2. $\mathscr{A}$ issues a signature query for the tuple (m, ID_s, Y_s), and gets a signature $\sigma = (\sigma_1, \sigma_2)$, where $\sigma_1 = g_2^t$, $\sigma_2 = (g_1^{h_s} \cdot R_s \cdot Y_{KGC})^{(\frac{c_s}{m} - t)x_s}$.
- Forge. $\mathscr{A}$ chooses at random a message $m' \in Z_p^*$, computes $\sigma'_1 = \sigma_1^{\frac{m}{m'}}$, $\sigma'_2 = \sigma_2^{\frac{m}{m'}}$, and outputs a forged signature $\sigma' = (\sigma'_1, \sigma'_2)$ on a tuple (ID_s, Y_s, m').

The tuple $(params, ID_s, Y_s, \sigma' = (\sigma'_1, \sigma'_2), m')$ is a valid signature due to the verification equation is satisfied.

$$
\begin{aligned}
e(Y'_{s1}, \sigma'_2) &= e\left(g_1^{\frac{yh_s}{(h_s+r_s+y)x_s}}, (g_1^{h_s} \cdot R_s \cdot Y_{KGC})^{(\frac{c_s}{m} - t)\cdot\frac{mx_s}{m'}}\right) \\
&= e\left(g_1^{\frac{yh_s}{(h_s+r_s+y)x_s}}, g_1^{(h_s+r_s+y)}\right)^{(\frac{c_s}{m'} - \frac{mt}{m'})x_s} \\
&= e(g_1, g_1)^{y(\frac{c_s}{m'} - \frac{mt}{m'})h_s} \\
&= \left(\frac{g_2^{\frac{c_s}{m'}}}{g_2^{\frac{mt}{m'}}}\right)^{h_s} \\
&= \left(\frac{Y'^{\frac{1}{m}}_{s2}}{\sigma'_1}\right)^{h_s}
\end{aligned}
$$

In this attack, an ordinary adversary $\mathscr{A}$ needs to get a signature σ on a message m under a tuple (ID_s, Y_s). $\mathscr{A}$ then can generate a forged signature σ' on a new message m'.

4. Problems analysis

In this section, it is analyzed for why Karati et al.'s scheme cannot withstand the above two attacks. In the first attack, the adversary $\mathscr{A}_1$ changes σ_2 to σ_2^u and turns Y_{s1} into $Y_{s1}^{\frac{1}{u}}$ for any $u \in Z_p^*$, he then can obtain a new signature for the message m. In the second attack, the adversary $\mathscr{A}_1$ changes σ_1 to $\sigma_1^{\frac{m}{m'}}$ and turns σ_2 to $\sigma_2^{\frac{m}{m'}}$, he then can obtain a signature for a new message m'.

References

[1] A. Karati, SK. Islam, and M. Karuppiah, "Provably secure and lightweight certificateless signature scheme for IIoT environments," *IEEE T Ind Inform.*, vol.14, no.8, pp.3701-3711, 2018.