Security Analysis of Two Lightweight Certificateless Signature Schemes

Nasrollah Pakniat

Information Science Research Center, Iranian Research Institute for Information Science and Technology (IRANDOC), Tehran, Iran.

Abstract

Certificateless cryptography can be considered as an intermediate solution to overcome the issues in traditional public key infrastructure (PKI) and identity-based public key cryptography (ID-PKC). There are a vast number of certificateless signature (CLS) schemes in the literature; however, most of them are not efficient enough to be utilized in limited resources environments such as Internet of Things (IoT) or Healthcare Wireless Sensor Networks (HWSN). Recently, two lightweight CLS schemes have been proposed by Karati et al. and Kumar et al. to be employed in IoT and HWSNs, respectively. While both schemes are claimed to be existentially unforgeable, in this paper, we show that both these signatures can easily be forged. More specifically, it is shown that 1) in Karati et al.’s scheme, a type 1 adversary, considered in certificateless cryptography, can generate a valid partial private key corresponding to any user of its choice and as a consequence, it can forge any users’ signature on any message of its choice, and 2) in Kumar et al.’s scheme, both types of adversaries which are considered in certificateless cryptography are able to forge any signer’s signature on an arbitrary message.

Introduction

Certificateless cryptography, introduced in 2003 by Al-riyami and Paterson [1], can be considered as an intermediate solution to overcome the issues in traditional public key infrastructure (PKI) and identity-based public key cryptography (ID-PKC) [2]. Whereas a trusted authority is needed in traditional PKI to bind the identity of an entity to his public key, ID-PKC requires a trusted private key generator to generate the private keys of users based on their identities. Therefore, the certificate management problem in the public-key setting is actually replaced by the key escrow problem. In certificateless cryptography, the users’ private keys are still generated with the help of a third party, called the key generation center (KGC). However, the KGC doesn’t have access to the final private keys generated by the users themselves (based on some private information obtained from the KGC and some secret values chosen by the users). The public key of a user is computed from the KGC’s public parameters and some information, private to the user, and is published by the user himself.

Regarding the security of a certificateless cryptographic scheme, two types of adversaries are considered in the literature; a Type 1 adversary $A_1$ who simulates malicious ordinary users and a Type 2 Adversary $A_2$ who simulates a malicious KGC in a cer-
tificateless cryptographic scheme. To perform these simulations, $A_1$ is allowed to replace the public key of entities with other values of its choice and $A_2$ is allowed to get access to the master secret key.

The first certificateless signature (CLS) scheme was proposed in [1] by Al-Riyami and Paterson. After this seminal work, a vast number of certificateless signature schemes were proposed such as ordinary CLS schemes [1][3][10], certificateless proxy signature schemes [11][14], certificateless aggregate signature schemes [1][15][20], certificateless signature schemes with designated tester [21][22], certificateless threshold signature schemes [23][24], certificateless ring signature schemes [26][27], and etc. However, due to their heavy computational costs, most of these schemes can not be applied in limited resource environments such as Internet of Things (IoT) and Healthcare Wireless Sensor Networks (HWSN). As a consequence, new efforts have been put forth to construct lightweight cryptographic schemes in certificateless setting in order to be applicable in limited resource environments. In this regard, recently, two lightweight certificateless signature schemes have been proposed by Karati et al. [3] and Kumar et al. [4]. The authors of both papers claimed that their proposed CLS schemes are existentially unforgeable. However, in this paper, we disprove their claims and show that the CLS schemes of [3] and [4] are both insecure. This is done by showing that:

- In Karati et al.’s CLS scheme, a type 1 adversary of certificateless cryptography is able to generate a valid partial private key corresponding to any identity of its choice and then uses this generated partial private key to forge the signature of the corresponding user on any message of its choice.
- In Kumar et al.’s CLS scheme, both types of adversaries, considered in certificateless cryptography, are able to violate the unforgeability of the scheme. More precisely, 1) a type 1 adversary is able to forge any signer’s signature on any message in this scheme as soon as it gets access to a pair of message and its corresponding signature of that signer, and 2) a type 2 adversary is able to forge each user’s signature on any message in this scheme (without even requiring to see a signature of that signer).

The rest of this paper is organized as follows. In Section 2, we provide the framework and the security definition of CLS schemes. In Section 3 after reviewing the CLS scheme of [3], we provide the proof of its insecurity. Then, the CLS scheme of Kumar et al. [4] and analysis of its security are reviewed in Section 4. Finally, the conclusions are provided in Section 5.

2 Certificateless Signature Schemes

In this section, we provide the framework and the security definition of Certificateless signature schemes.

2.1 The Framework

There exist three entities in a CLS scheme: a key generation center (KGC) which helps users to generate their private keys, a signer, and a verifier. A CLS scheme consists of six algorithms: Setup, Set-Partial-Private-Key, Set-Secret-Value, Set-Public-Key, CLSSign and CLS-Verify. The details of these algorithms are described in the following:

**Setup:** Performed by KGC.
- **Input:** The security parameter $k$.
- **Process:**
  - Generates the master secret key $MSK$, and the public parameters $params$.
- **Output:** The master secret key $MSK$ which will be secured by KGC and the public parameters $params$ which are published.

**Set-Partial-Private-Key:** Performed by KGC.
- **Input:** $params$, $MSK$ and a user’s identity $ID_S$.
- **Process:**
  - Computes a partial private key $D_S$ corresponding to this user.
- **Output:** Partial private key $D_S$ which will be sent securely to the user with identity $ID_S$.

**Set-Private-Key:** Performed by a user $S$.
- **Input:** $params$ and $S$’s partial private key $D_S$.
- **Process:**
  - Computes the public key $PK_S$ by using it and $D_S$.
- **Output:** $PK_S$ which will be published.

**Set-Public-Key:** Performed by a user $S$.
- **Input:** $params$ and $S$’s private key $SK_S$.
- **Process:**
  - Computes the public key $PK_S$.
- **Output:** $PK_S$ which will be published.

**CLS-Sign:** Performed by the user $S$.
- **Input:** $params$, the user’s identity $ID_S$ and his private key $SK_S$, and a message $m$.
- **Process:**
  - Generates a signature $\sigma$ on the message $m$.
- **Output:** $\sigma$ as the signature on $m$.

**CLS-Verify:** Performed by the verifier.
- **Input:** $params$, signer’s identity $ID_S$ and his public key $PK_S$, message $m$ and a signature $\sigma$.
- **Process:**
  - Checks the validity of $\sigma$. 

2.2 Security Model

To call a CLS scheme secure, it should provide existentially unforgeability against adaptive chosen-message and -identity attacks in the adversarial model of certificateless cryptography which consists of the following two types of adversaries:

- A type-1 adversary \( A_1 \), that has not access to the master secret key but can replace any signer’s public key with any value of its choice.
- A type-2 Adversary \( A_2 \), that has access to the master secret key but cannot replace public keys.

The security of a CLS scheme is modeled through the following two games played between a challenger \( C \) and adversaries \( A_1 \) or \( A_2 \).

**Game 1:** This game, played between \( C \) and \( A_1 \), consists of the following phases:

- **Setup:** In this phase, \( C \) generates the master secret key \( MSK \) and the public parameters \( params \). It keeps \( MSK \) secure and sends \( params \) to \( A_1 \).
- **Queries:** In this phase, \( A_1 \) can perform a polynomially bounded number of the following queries and \( C \)'s answers to these queries are as follows:
  - Request-Partial-Private-Key \( (ID_S) \): inputting \( ID_S \) to this query, \( A_1 \) will get \( S \)'s partial private key \( D_S \) as the output.
  - Request-Secret-Value \( (ID_S) \): inputting \( ID_S \) to this query, \( A_1 \) will get \( S \)'s secret value \( x_S \) as the output.
  - Request-Public-Key \( (ID_S) \): inputting \( ID_S \) to this query, \( A_1 \) will get \( S \)'s public key \( PK_S \) as the output.
  - Replace-Public-Key \( (ID_S, PK'_S) \): inputting \( ID_S \) and \( PK'_S \) to this query, \( PK'_S \) will be set as the public key corresponding to the user \( S \).
  - CL-Sign \( (ID_S, m) \): inputting \( ID_S \) and \( m \) to this query, \( A_1 \) will get \( \sigma \) as the output which is a valid signature of \( S \) on \( m \).
- **Output:** Finally, when \( A_1 \) decides to end the queries phase, it outputs a signature \( \sigma \) on a message \( m \) on behalf of a targeted user with identity \( ID \). It wins the game if the following conditions are fulfilled:
  - The algorithm CLS-Verify outputs VALID on inputs \( params, m, \sigma, ID, \) and \( PK \) where, \( PK \) is the public key corresponding to the user with identity \( ID \).
  - The queries Request-Partial-Private-Key(\( ID \)) and CL-Sign(\( ID, m \)) weren’t queried in the queries phase.

**Definition 1.** A CLS scheme is Type-1 secure against the adaptively chosen-message and -identity attack if the advantage of any polynomially bounded adversary \( A_1 \) in winning Game 1 be negligible.

**Game 2:** This game, played between \( C \) and \( A_2 \), consists of the following phases:

- **Setup:** In this phase, \( C \) generates the master secret key \( MSK \) and the public parameters \( params \) and sends them to \( A_2 \).
- **Queries:** In this phase, \( A_2 \) can perform a polynomially bounded number of queries as in Game 1 and \( C \) answers them in the same way. The only constraint here is that \( A_2 \) is not allowed to replace any public keys. Note that \( A_2 \) knows \( MSK \) and can compute the partial private key of any identity by itself.
- **Output:** Finally, when \( A_2 \) decides to end the queries phase, it outputs a signature \( \sigma \) on a message \( m \) on behalf of a targeted user with identity \( ID \). It wins the game if the following conditions are fulfilled:
  - The algorithm CLS-Verify outputs VALID on inputs \( params, m, \sigma, ID, \) and \( PK \) where, \( PK \) is the public key corresponding to the user with identity \( ID \).
  - The queries Request-Secret-Value(\( ID \)) and CL-Sign(\( ID, m \)) weren’t queried in the queries phase.

**Definition 2.** A CLS scheme is Type-2 secure against the adaptively chosen-message and -identity attack if the advantage of any polynomially bounded adversary \( A_2 \) in winning Game 2 be negligible.

3 Karati et al.’s CLS Scheme

In this section, we first review Karati et al.’s CLS scheme and then prove that it is completely insecure.

3.1 Review of the Scheme

The CLS scheme of Karati et al. consists of the following algorithms:

**Setup:** Performed by \( KGC \).

- **Input:** The security parameter \( k \).
- **Process:**
  - Generates two groups \( G_1 \) and \( G_2 \) with the same prime order \( p \) and an efficient bilinear pairing \( e : G_1 \times G_1 \rightarrow G_2 \).
  - Chooses a generator \( g_1 \in G_1 \).
  - Chooses a cryptographic hash function \( H : \{0,1\}^* \rightarrow Z_p^* \).
  - Chooses a random \( y \in Z_p^* \) as his master secret.
key.
- Computes $g_2 = e(g_1, g_1)^h$ and $Y_{KGC} = g_1^y$.
- Output: The master secret key $y$ which will be secured by KGC and the public parameters $params = (G_1, G_2, p, c, g_1, g_2, Y_{KGC}, H)$ which will be published.

**Set-Partial-Private-Key:** Performed by KGC.

- Input: $params$, master secret key $y$ and a user’s identity $ID_i \in \{0, 1\}^*$. 
- Process:
  - Computes $h_i = H(ID_i)$. 
  - Chooses $r_i \in Z_p^*$ randomly and computes $R_i = g_1^{y_i}$ and $y_i = (g_1)^{y_i \cdot r_i \cdot (h_i \cdot R_i \cdot Y_{KGC})}$. 
- Output: Partial private key $D_i = (y_i, R_i)$ which will be sent securely to the user with identity $ID_i$. After receiving $D_i$ from KGC, the user considers $D_i$ genuine if:
  
  $$e(g_1, Y_{KGC})^h_i = e(y_i, (g_1^h \cdot R_i \cdot Y_{KGC})) \cdot (1)$$

**Set-Private-Key:** Performed by a user $i$.

- Input: $params$ and $i$’s partial private key $D_i = (y_i, R_i)$. 
- Process:
  - Chooses $x_i, c_i \in Z_p^*$ randomly and sets $SK_i = (c_i, x_i, R_i)$.
  - Output: $SK_i$ which will be secured by the user $i$.

**Set-Public-Key:** Performed by a user $i$.

- Input: $params$, $i$’s partial private key $D_i = (y_i, R_i)$ and his private key $SK_i = (c_i, x_i, R_i)$.
- Process:
  - Computes $Y_i = (Y_{i1} = (y_i)^{x_i} \cdot Y_{i2} = g_2^{y_i})$ as the user’s public key.
  - Output: $Y_i$ which will be published.

**CLS-Sign:** Performed by a user $S$.

- Input: $params$, the user’s identity $ID_S$ and his private key $SK_S = (c_S, x_S, R_S)$ and a message $m$.
- Process:
  - Computes $h_S = H(ID_S)$.
  - Chooses a random value $t \in Z_p^*$ and computes
    
    $$\sigma_1 = g_1^t,$$
    
    $$\sigma_2 = (g_1^{h_S} \cdot R_S \cdot Y_{KGC})^{(\frac{c_S}{x_S} - t)} \cdot x_S . \quad (2)$$
  - Output: $\sigma = (\sigma_1, \sigma_2)$ as the signature on $m$.

**CLS-Verify:** Performed by the verifier.

- Input: $params$, $S$’s identity $ID_S$ and his public key $Y_S = (Y_{S1}, Y_{S2})$, message $m$ and a signature $\sigma = (\sigma_1, \sigma_2)$.
- Process:
  - Computes $h_S = H(ID_S)$.
  - Checks whether $(\frac{Y_{S1}}{\sigma_1})^{h_S} = e(Y_{S1}, \sigma_2)$.
  - Output: VALID if the above equation holds and INVALID otherwise.

### 3.2 Cryptanalysis of the Scheme

The authors of [9] claimed that their proposed scheme is a secure certificateless signature scheme. However, in this section, we disprove their claim. More specifically, we show that by accessing to a valid partial private key corresponding to any user, a valid partial private key corresponding to any other user can be generated. Thereupon, each user of this scheme can forge the signature of other users on any arbitrary message of his choice. This is formally stated and proved in the following theorem.

**Theorem 1.** Let $S$ with identity $ID_S$ be an arbitrary user of Karati et al.’s scheme. Suppose that $A_1$ has access to a valid partial private key corresponding to $S$. Then, $A_1$ is able to generate a valid partial private key corresponding to any other user $S'$ with arbitrary identity $ID_{S'}$ and as a consequence, he is able to forge $S'$’s signature on any message of his choice.

**Proof.** According to Set-Partial-Private-Key algorithm of Karati et al.’s CLS scheme, the partial private key corresponding to $S$ with identity $ID_S$ is a pair $(y_S, R_S)$ where, $R_S = g_1^{y_S}$ and $y_S = (g_1)^{y_S \cdot r_S \cdot (h_S \cdot R_S \cdot Y_{KGC})}$ in which $r_S \in Z_p^*$ is an unknown randomly chosen value, $y$ is the master secret key and $h_S = H(ID_S)$. In the following, we show how $A_1$ is able to use $S$’s partial private key to generate a valid partial private key corresponding to any other user $S'$ with arbitrary identity $ID_{S'}$. To this end, $A_1$:

1. Computes $h_{S'} = H(ID_{S'}) \in Z_p^*$.
2. Computes $\alpha = \frac{h_{S'}}{h_S} \in Z_p^*$. Note that the output of $H(\cdot)$ is a member of $Z_p^*$ and therefore, $h_{S'}$ exists in $Z_p^*$.
3. Computes $y_{S'} = y_3^3$ and $R_{S'} = \frac{R_S}{g_1^{h_{S'} \cdot c_S \cdot Y_{KGC}}}.$
4. Sets $(y_{S'}, R_{S'})$ as the partial private key corresponding to the user $S'$ with identity $ID_{S'}$.

Using the following relation, it can easily be verified that $(y_{S'}, R_{S'})$ is a valid partial private key corresponding to the user $S'$ with identity $ID_{S'}$:

\[ (\frac{Y_{S1}}{\sigma_1})^{h_{S'}} = e(Y_{S1}, \sigma_2). \]
After computing $S$'s partial private key, $A_1$ can perform Set-Private-Key and Set-Public-Key (as explained in Katari et al.'s CLS scheme) instead of $S'$ to compute a valid pair of private and public keys corresponding to $S'$. Now, using the private key of $S'$, $A_1$ can forge $S'$’s signature through CLS-Sign algorithm on any message of its choice.

\[
\begin{align*}
\text{CLS-Sign: Performed by a user } S.&&\\
\text{Input: } &\text{ signer’s identity } ID_S, \text{ his public key } Y_S, \text{ his private key } SK_S = (x_S, D_S), \text{ some state information } \Delta \text{ and a message } m.&&\\
\text{Process: }&\text{ chooses a random value } r \in Z_q^*\text{ and computes } R = r \cdot P \in G_1.&&\\
&\text{ computes } W = H_2(\Delta)\text{ and } h = H_3(m, ID_S, Y_S, R).&&\\
&\text{ computes } V = D_S + r \cdot W + h \cdot x_S \cdot P_{Pub}.&&\\
\text{Output: } &\text{ }\sigma = (R, V) \text{ as the signature on } m \text{ under the state information } \Delta.&&
\end{align*}
\]

4 Kumar et al.’s CLS Scheme

In this section, we first review Kumar et al.’s CLS scheme and then prove that their scheme is forgeable.

4.1 Review of the Scheme

The CLS scheme of Kumar et al. [4] consists of the following algorithms:

**Setup:** Performed by $KGC$.
- Input: The security parameter $k$.
- Process:
  - Chooses two groups $G_1$ and $G_2$ with the same prime order $q$ and a generator $P$ in $G_1$.
  - Chooses a bilinear map $e : G_1 \times G_1 \rightarrow G_2$.
  - Chooses a random $\alpha \in Z_q^*$ as the master secret key and sets $P_{Pub} = \alpha \cdot P$.
  - Chooses cryptographic hash functions $H_1, H_2 : \{0, 1\}^* \rightarrow G_1$ and $H_3 : \{0, 1\}^* \rightarrow Z_q^*$.
- Output: The master secret key $\alpha$ which will be secured by $KGC$ and the system parameters $\text{params} = (q, G_1, G_2, e, P, P_{Pub}, H_1, H_2, H_3)$ which will be published.

**Set-Partial-Private-Key:** Performed by $KGC$.
- Input: $\text{params}$, master secret key $\alpha$ and a user’s identity $ID_i \in \{0, 1\}^*$.
- Process:
  - Computes $Q_{ID_i} = H_1(ID_i)$.
  - Computes $D_i = \alpha \cdot Q_{ID_i}$.
- Output: Partial private key $D_i$ which will be sent securely to the user with identity $ID_i$.

**Set-Private-Key:** Performed by a user $i$.
- Input: $\text{params}$ and $i$’s identity $ID_i$.
- Process:
  - Chooses a random value $x_i \in Z_q^*$ as the $i$’s secret key.
  - Sets $SK_i = (x_i, D_i)$.
- Output: $SK_i$ which will be published.

**Set-Public-Key:** Performed by a user $i$.
- Input: $\text{params}$ and $i$’s private key $SK_i = (x_i, D_i)$.
- Process:
  - Computes $Y_i = x_i \cdot P$ as $i$’s public key.
  - Output: $Y_i$ which will be published.

CLS-Verify: Performed by the verifier.
- Input: $\text{params}$, signer’s identity $ID_S$, his public key $Y_S$, his private key $SK_S = (x_S, D_S)$, some state information $\Delta$ and a signature $\sigma = (R, V)$.
- Process:
  - Computes $Q_{ID_S} = H_1(ID_S)$, $W = H_2(\Delta)$ and $h = H_3(m, ID_S, Y_S, R)$.
  - Verifies $e(V, P) \overset{?}{=} e(Q_{ID_S} + h \cdot Y_S, P_{Pub})e(R, W)$.
- Output: $\text{VALID}$ if the above equation holds and INVALID otherwise.

4.2 Cryptanalysis of the Scheme

Kumar et al. claimed that their scheme is existentially unforgeable against adaptive chosen message attacks. However, in this section, we disprove their claim. We prove the insecurity of Kumar et al.’s CLS scheme by the following theorems:

**Theorem 2.** Let $S$ be a signer with identity $ID_S$ who uses Kumar et al.’s CLS scheme. Suppose that a type 1 adversary $A_1$ has access to a tuple $(m, \sigma = (R, V), \Delta)$, where $\sigma$ is $S$’s signature on message $m$ under the state information $\Delta$. Then, $A_1$ is able to forge $S$’s signature on any new message $m’$ under the same state information $\Delta$.

**Proof.** According to Kumar et al.’s CLS-Sign algorithm, the signature $\sigma$ is as follows:

\[
R = r \cdot P, \quad V = D_S + r \cdot H_2(\Delta) + x_S \cdot h \cdot P_{Pub}.\]
where \( h = H_3(m, ID_S, Y_S, R) \) and \( r \in Z_q^* \) is a random value that is unknown to \( A_1 \). Now, in order to forge \( S \)'s signature on a new message \( m' \), \( A_1 \):

1. Issues a Request-Secret-Value query on the input of \( ID_S \) and obtains \( x_S \) as the result.
2. Computes \( D_{S, \Delta} = V - x_S \cdot h \cdot P_{Pub} = D_S + r \cdot H_2(\Delta) \).
3. Uses \( D_{S, \Delta}, x_S \) and \( R \) to forge \( S \)'s signature on \( m' \) as follows:
   1. Computes \( h' = H_3(m', ID_S, Y_S, R) \) and \( V' = D_{S, \Delta} + h' \cdot x_S \cdot P_{Pub} \).
   2. Outputs \( \sigma' = (R, V') \) as \( S \)'s signature on message \( m' \).

It can be easily verified that the forged signature \( \sigma' \) is valid.

\[ \square \]

**Theorem 3.** Let \( S \) be a signer with identity \( ID_S \) who uses Kumar et al.'s CLS scheme. Then, a type 2 adversary \( A_2 \) is able to forge \( S \)'s signature on any message \( m \) of its choice under any arbitrary state message information \( \Delta \).

**Proof.** To forge \( S \)'s signature on any arbitrary message \( m, A_2 \):

1. Chooses a random value \( r \in Z_q^* \) and computes \( R = r \cdot P \).
2. Computes \( h = H_4(m, ID_S, Y_S, R) \) and \( V = D_S + h \cdot \alpha \cdot Y_S \).
3. Outputs \( \sigma = (R, V) \) as \( S \)'s signature on message \( m' \).

Note that \( A_2 \) acts as the malicious key generation center and has access to partial private keys. It can be easily verified that the forged signature \( \sigma \) is valid. \[ \square \]

5 Conclusion

In this paper, the security of two recently proposed lightweight certificateless signature schemes is considered. We prove that in one of them, a type 1 adversary of certificateless cryptography can forge the signature of any user on any arbitrary message of his choice and in the other one, both considered types of adversaries in certificateless cryptography can forge valid signatures on behalf of any user on any message of their choices.

**References**


Nasrollah Pakniat has a PhD in Mathematics, graduated in 2015 from Shahid Beheshti University. He received his MSc degree in Computer Science from Shahid Beheshti University in 2011. He holds a BSc degree in Computer Science from Shahid Bahonar University of Kerman in 2008. He began his scientific experience in 2016 as a faculty member in Iranian Research Institute for Information Science and Technology (IranDoc). His research interests include cryptography, network security and text mining.