Insecurity of An Anonymous Authentication For Privacy-preserving IoT Target-driven Applications

Xi-Jun Lin *and Lin Sun †

November 28, 2013

Abstract: The Internet of Things (IoT) will be formed by smart objects and services interacting autonomously and in real-time. Recently, Alcaide et al. proposed a fully decentralized anonymous authentication protocol for privacy-preserving IoT target-driven applications. Their system is set up by an ad-hoc community of decentralized founding nodes. Nodes can interact, being participants of cyberphysical systems, preserving full anonymity. In this study, we point out that their protocol is insecure. The adversary can cheat the data collectors by impersonating a legitimate user.

Key words: Anonymous credential system; Fully decentralized protocol; Threshold cryptography; Zero-knowledge proof of knowledge; Smart community

1 Introduction

The Internet of Things (IoT) will be formed by smart objects and services interacting autonomously and in real-time. Recently, Alcaide et al. [1] proposed a fully decentralized anonymous authentication protocol for privacy-preserving IoT target-driven applications. Their system is set up by an ad-hoc community of decentralized founding nodes. Nodes can interact, being participants of cyberphysical systems, preserving full anonymity.

In Alcaide et al.'s protocol, two roles are defined for participant nodes: *Users* and *Data Collectors*. Users are the nodes originating the data. They can anonymously and unlinkably authenticate themselves in front of data collectors proving possession of a valid Anonymous Access Credential (AAC) encoding a particular set of attributes. Data collectors are entities responsible for the collection of data from authorized users. Therefore, before collecting the data, the data collector must verify that the user holds a valid AAC encoding a particular set of attributes.

The main characteristics of their protocol are:

- The protocol does not rely on any central organization.
- The users of the system jointly generate and distribute a private key that is used in a (t, n)-threshold fashion. This allows up to t nodes to be compromised without the key being compromised.

*X.J.Lin is with the Department of Computer Sciences and Technology, Ocean University of China. Qingdao 266100, P.R.China. email: linxj77@163.com

†L. Sun is with the College of Liberal Arts, Qingdao University. Qingdao 266071, P.R.China. email: sunlin9@126.com

- Users of the system can obtain an AAC encoding a set of certified attributes and use such an AAC to anonymously and unlinkably authenticate themselves to data collector entities.
- Data collectors can anonymously authenticate users by means of attribute-based boolean formulas, which can be defined and modified by the data collector itself at any time.

Without loss of generality, their protocol is described assuming that users are authenticated based on only two attributes (x_1, x_2), which can be derived from a standard certificate $\tilde{x}$ signed by some trusted external entity (i.e., a certification authority). In the protocol, once verified, the attributes (x_1, x_2) will be encoded into an AAC so that the user can prove possession of those attributes without disclosing their actual value.

Any linear boolean formula (or a conjunction of formulas) can be used to anonymously authenticate users as legitimate participants. Such a formula can be represented as the function $\Phi_R(x_1, x_2)$. As mentioned before, users can prove that the attributes x_1 and x_2, encoded in a AAC, satisfy $\Phi_R(x_1, x_2) = 1$ without revealing the actual attribute values.

In this study, we point out that their protocol is insecure. The adversary can cheat the data collectors by impersonating a legitimate user.

2 Alcaide et al.'s Protocol

Their protocol consists of three phases: *Set-Up* (where all parameters are generated), *User Registration* (users obtain AAC) and *Credential Proving* (users prove possession of a valid AAC to a data collector).

2.1 Set-Up

In this phase, a community of n users, $\{\mathcal{U}_i : i = 1, \cdots, n\}$, known as founders, comes together to generate all protocol parameters. As a result of this phase, each founder $\mathcal{U}_i$ holds two tuples:

- a public tuple:

$$Pub_i = (N, \varepsilon, V_1, V_2, V_3, e, g, s, c, g_1, g_2, g_3, q, h_1, h_2, h_3, h_4, h_5, h_6)$$

  equal for all founders.

- a private tuple:

$$Priv_i = (p_{1_i}, p_{2_i}, \phi_i(N), d_i, v_{1_i}, v_{2_i}, v_{3_i})$$

  different for each founder $\mathcal{U}_i$.

$(N, \varepsilon, V_1, V_2, V_3, d, v_1, v_2, v_3)$ are generated by the founders in a joint and decentralized manner, where $d = \varepsilon^{-1}$, $v_1 = V_1^{-1}$, $v_2 = V_2^{-1}$, $v_3 = V_3^{-1}$ (mod $\phi(N)$) and $N = 4p_1p_2 + 2p_1 + 2p_2 + 1$ such that (p_1, p_2) are *Sophie Germain* primes. q is a prime picked by a

super node such that $N|(q-1)$. $(e, g, c, h_1, \cdots, h_6)$ are picked by the super node randomly and broadcasted to the founders. $s = g^d, g_1 = g_1^v, g_2 = g_2^v, g_3 = g_3^v \pmod N$ are also broadcasted by the super node so that every founder holds them.

$Priv_i = (p_{1_i}, p_{2_i}, \phi_i(N), d_i, v_{1_i}, v_{2_i}, v_{3_i})$ are shares that the founder $\mathcal{U}_i$ holds of the secret values $(p_1, p_2, \phi(N), d, v_1, v_2, v_3)$ respectively.

At the end of this phase, the founders have computed the necessary cryptographic material to create AACs for valid data holders later on.

The detail of this phase which has nothing to do with our attack is omitted here.

2.2 User Registration

In this phase, a new user $\mathcal{U}_{new}$, holding a valid certificate $\tilde{x}$, obtains the necessary material from $(t+1)$ founders to construct its own Anonymous Access Credential AAC_{new}, a token that encodes $\mathcal{U}_{new}$'s attributes $(x_1, x_2) \in \mathbb{Z}_e$ and allows $\mathcal{U}_{new}$ to anonymously and unlinkably prove to the data collector to be a legitimate participant. Before $\mathcal{U}_{new}$ can participate in the system, it needs to obtain the protocol public parameters. In order to do that, $\mathcal{U}_{new}$ simply requests the tuple Pub_i from some founder $\mathcal{U}_i$ and makes $Pub_{new} = Pub_i$.

The process that $\mathcal{U}_{new}$ must follow to construct AAC_{new} is described as follows.

- $\mathcal{U}_{new}$ picks $x_3 \in_R \mathbb{Z}_e^*$ such that $gcd(x_3, \varepsilon) = 1$ and $x \in_R \mathbb{Z}_N^*$, and then computes $a' = g_3^{x_3} x^e \pmod N$. Then, it picks $b \equiv c \pmod N$ randomly, and then sends $(\tilde{x}, a', b)$ to the founders.
- After receiving $(\tilde{x}, a', b)$, each founder $\mathcal{U}_i$ verifies $\tilde{x}$ and extracts the attributes (x_1, x_2) from $\tilde{x}$. Then, it computes $a = g_1^{x_1} g_2^{x_2} a' \pmod N$ and $m = (a + b) \pmod N$. Furthermore, it signs m with its private tuple $Priv_i$ and sends $\sigma_i(m)$ to $\mathcal{U}_{new}$.
- After receiving $(t+1)$ partial signatures $\sigma_i(m)$ from the founders, $\mathcal{U}_{new}$ generates the signature $v = \sigma(m)$. Then, it computes $a = g_1^{x_1} g_2^{x_2} a' \pmod N$ and $m = (a + b) \pmod N$, and then checks whether the equation $v^\varepsilon = m \pmod N$ holds. If the equation holds, $\mathcal{U}_{new}$ obtains its own Anonymous Access Credential $AAC_{new} = (x_1, x_2, x_3, x, v)$.

AAC_{new} must be kept secretly at all times, only randomized versions of it will be used in the *Credential Proving* phase. It should be noted that $\mathcal{U}_{new}$ must show its certificate to the founders.

2.3 Credential Proving

When a user $\mathcal{U}$ interacts with a data collector $\mathcal{R}$, $\mathcal{U}$ must anonymously and unlinkably prove to $\mathcal{R}$ that it holds the appropriate attributes to satisfy the boolean formula Φ_R that $\mathcal{R}$ uses to authenticate legitimate participants.

As the process that $\mathcal{U}$ must follow to accomplish such a proof is the key part of the anonymous credential system. Each of the sub-processes is given as follows.

1. **Session values and commitments**: every time that $\mathcal{U}$ wants to authenticate to the data collector $\mathcal{R}$, $\mathcal{U}$ must generate and commit a set of session values, which are secret random values used during a particular authentication operation. This process is given as follows.

(a) $\mathcal{U}$ picks $y \in_R \mathbb{Z}_e$, and then computes

$$\begin{aligned} \hat{v} &= s^y v \pmod N \\ \hat{m} &= g^y m \pmod N \\ \hat{a} &= g^y a \pmod N \\ \hat{b} &= g^y b \pmod N \end{aligned}$$

Then, it picks $w_1, w_2, w_3 \in \mathbb{Z}_N$ and computes

$$\begin{aligned} \bar{m} &= h_1^{\hat{m}} h_2^{w_1} \pmod q \\ \bar{a} &= h_3^{\hat{a}} h_4^{w_2} \pmod q \\ \bar{b} &= h_5^{\hat{b}} h_6^{w_3} \pmod q \\ \bar{c} &= \bar{m}\bar{a}\bar{b} \pmod q \end{aligned}$$

(In the original paper, $\bar{a} = h_2^{\hat{a}} h_3^{w_2} \pmod q$. According to the Appendix A.2-Theorem 2 [1], we correct this typo by replacing it with $\bar{a} = h_3^{\hat{a}} h_4^{w_2} \pmod q$.)

(b) $\mathcal{U}$ sends and commits the session values $(\hat{a}, \bar{m}, \bar{b}, \bar{c})$ to $\mathcal{R}$.

2. **Zero-Knowledge Proofs of Knowledge (ZKPoKs)**: once $\mathcal{U}$ has committed some set of values, $\mathcal{U}$ and $\mathcal{R}$ engage in a series of four ZKPoKs:

   In order to authenticate $\mathcal{U}$, every operation that $\mathcal{R}$ performs at the end of each ZKPoK must succeed. Whatever data $\mathcal{R}$ collects from $\mathcal{U}$ must follow these authentication subprocess.

   - **ZKPoK1**: The objective of this ZKPoK is to prove that the private attributes (x_1, x_2) that $\mathcal{U}$ holds in its AAC satisfy the boolean formula $\Phi_R(x_1, x_2)$. Note that Φ_R can be any linear boolean formula; however, for simplicity reasons, as an example, the authors use:

     $$\Phi_R(x_1, x_2) = 1 \Leftrightarrow \alpha x_1 + \beta = x_2 : (\alpha, \beta) \in \mathbb{Z}$$

     α, β being chosen by data collector $\mathcal{R}$.

     Consequently, $\mathcal{U}$ must prove, in a zero-knowledge manner, possession of AAC encoding attributes (x_1, x_2) such that $\alpha x_1 + \beta = x_2$. This ZKPoK is given as follows.

     – $\mathcal{U}$ picks $r'_y, r'_1, r'_3 \in_R \mathbb{Z}_e$ and $r' \in_R \mathbb{Z}_N^*$, then computes and sends $t' = g^{r'_y}(g_1 g_2^{\alpha})^{r'_1} g_3^{r'_3} r'^e \pmod N$ to $\mathcal{R}$.

     – $\mathcal{R}$ picks and responds $\gamma \in_R \mathbb{Z}_{2^t}$ to $\mathcal{U}$.

     – $\mathcal{U}$ computes

     $$\begin{aligned} s'_y &= \gamma y + r'_y \pmod e \\ s'_1 &= \gamma x_1 + r'_1 \pmod e \\ s'_3 &= \gamma x_3 + r'_3 \pmod e \\ s' &= g^{(\gamma y + r'_y) dive} (g_1 g_2^{\alpha})^{(\gamma x_1 + r'_1) dive} g_3^{(\gamma x_3 + r'_3) dive} x^{\gamma} r' \pmod N \end{aligned}$$

     It sends (s'_y, s'_1, s'_3, s') to $\mathcal{R}$.

     – $\mathcal{R}$ checks whether the equation $(\hat{a} g_2^{-\beta})^{\gamma} t' = g^{s'_y} (g_1 g_2^{\alpha})^{s'_1} g_3^{s'_3} s'^e \pmod N$ holds.

- **ZKPoK2**: The goal of this ZKPoK is to prove that all session value commitments generated by $\mathcal{U}$ are valid. For that matter, $\mathcal{U}$ must prove, in a zero-knowledge manner, knowledge of the secret tuple $(y, \hat{a}, \hat{b}, \hat{m})$. This ZKPoK is given as follows.
  - $\mathcal{U}$ picks $r'_{\hat{b}}, r'_1, r'_2, r'_3 \in_R \mathbb{Z}_q$. Then, it computes and sends $t' = (h_1 h_5)^{r'_{\hat{b}}} h_2^{r'_1} h_4^{r'_2} h_6^{r'_3} \pmod{q}$ to $\mathcal{R}$.
  - $\mathcal{R}$ picks and responds $\gamma \in_R \mathbb{Z}_q$ to $\mathcal{U}$.
  - $\mathcal{U}$ computes
$$\begin{aligned} s'_{\hat{b}} &= \gamma\hat{b} + r'_{\hat{b}} \\ s'_1 &= \gamma w_1 + r'_1 \\ s'_2 &= \gamma w_2 + r'_2 \\ s'_3 &= \gamma w_3 + r'_3 \end{aligned}$$
    It sends $(s'_{\hat{b}}, s'_1, s'_2, s'_3)$ to $\mathcal{R}$. (In the original paper, (s'_y, s'_1, s'_3, s') is sent to $\mathcal{R}$, we correct this typo by replacing it with $(s'_{\hat{b}}, s'_1, s'_2, s'_3)$.)
  - $\mathcal{R}$ checks whether the equation $(\bar{c}(h_1 h_3)^{-\hat{a}})^{\gamma} t' = (h_1 h_5)^{s'_{\hat{b}}} h_2^{s'_1} h_4^{s'_2} h_6^{s'_3} \pmod{q}$ holds.
- **ZKPoK3**: The objective of this ZKPoK is to prove that the signature v encoded in $\mathcal{U}$'s AAC is valid. In order to do that, $\mathcal{U}$ proves, in a zero-knowledge manner, knowledge of the ε-th root of the h_1-part, which is equivalent to proving that $\hat{v}$ is the ε-th root of $\hat{m}$. This ZKPoK given below must be executed L times.
  - $\mathcal{U}$ picks $r'_1, r'_2 \in_R \mathbb{Z}_N$. Then, it computes and sends $t' = h_1^{r_1'^{\varepsilon}} h_2^{r'_2} \pmod{q}$ to $\mathcal{R}$.
  - $\mathcal{R}$ picks and responds $\gamma \in_R \{0, 1\}$ to $\mathcal{U}$.
  - $\mathcal{U}$ preforms as follows.
    If $\gamma = 0$, let $s'_1 = r'_1$ and $s'_2 = r'_2$.
    If $\gamma = 1$, it computes $s'_1 = r'_1 \hat{v}^{-1} \pmod{N}$ and $s'_2 = r'_2 - w_1 (r'_1 \hat{v}^{-1})^{\varepsilon} \pmod{N}$.
    $\mathcal{U}$ sends (s'_1, s'_2) to $\mathcal{R}$.
  - $\mathcal{R}$ performs as follows.
    If $\gamma = 0$, it checks whether the equation $t' = h_1^{s_1'^{\varepsilon}} h_2^{s'_2} \pmod{q}$ holds.
    If $\gamma = 1$, it checks whether the equation $t' = \bar{m}_1^{s_1'^{\varepsilon}} h_2^{s'_2} \pmod{q}$ holds.
- **ZKPoK4**: The goal of this ZKPoK is to prove that $\mathcal{U}$ knows the session value y involved in ZKPoK1 - 3. For that matter, $\mathcal{U}$ must prove, in a zero-knowledge manner, knowledge of the discrete logarithm with base g of the h_5-part. This ZKPoK given below must be executed L times.
  - $\mathcal{U}$ picks $r'_1, r'_2 \in_R \mathbb{Z}_N$. Then, it computes and sends $t' = h_5^{g^{r'_1}} h_6^{r'_2} \pmod{q}$ to $\mathcal{R}$. (In the original paper, g' is used to compute t', we correct this typo by replacing it with g.)
  - $\mathcal{R}$ picks and responds $\gamma \in_R \{0, 1\}$ to $\mathcal{U}$.
  - $\mathcal{U}$ preforms as follows.
    If $\gamma = 0$, let $s'_1 = r'_1$ and $s'_2 = r'_2$.
    If $\gamma = 1$, it computes $s'_1 = r'_1 - y$ and $s'_2 = r'_2 - w_3 c^{-1} g^{s'_1}$.
    $\mathcal{U}$ sends (s'_1, s'_2) to $\mathcal{R}$.

– $\mathcal{R}$ performs as follows.

If $\gamma = 0$, it checks whether the equation $t' = h_5^{g^{s'_1}} h_6^{s'_2} \pmod q$ holds. (In the original paper, g' is used in the equation, we correct this typo by replacing it with g.)

If $\gamma = 1$, it checks whether the equation $t' = (\bar{b}^{c^{-1}})^{g^{s'_1}} h_6^{s'_2} \pmod q$ holds.

Note that, if and only if the authentication of user $\mathcal{U}$ succeeds (i.e., all ZKPoK1 - 4 are successful), the data collector $\mathcal{R}$ collects data from $\mathcal{U}$.

3 Cryptanalysis

In this section, we present an attack to point out that an adversary $\mathcal{A}$ can cheat the data collector by impersonating a legitimate user. In our attack, $\mathcal{A}$ needs not compromise user nodes.

It should be noted that if and only if $\mathcal{A}$ processes all ZKPoK1-4 with a data collector $\mathcal{R}$ successfully, $\mathcal{A}$ can cheat $\mathcal{R}$ into collecting data from it.

Suppose that the public parameters are $(N, \varepsilon, V_1, V_2, V_3, e, g, s, c, g_1, g_2, g_3, q, h_1, h_2, h_3, h_4, h_5, h_6)$, and $\mathcal{R}$'s data collection policy is the equation $\Phi_R(x_1, x_2) = 1 \Leftrightarrow \alpha x_1 + \beta = x_2 : (\alpha, \beta) \in \mathbb{Z}$.

In the *Credential Proving* phase, the adversary $\mathcal{A}$ interacts with $\mathcal{R}$ anonymously and unlinkably as follows.

1. **Session values and commitments**: $\mathcal{A}$ picks $y \in_R \mathbb{Z}_e, \hat{v}, a \in_R \mathbb{Z}_N^*$ and $w, w_1, w_3 \in_R \mathbb{Z}_N$. Then, it computes

   - $\hat{a} = g_2^{-\beta} a^e \pmod N$
   - $\bar{m} = h_1^{\hat{m}} h_2^{w_1} \pmod q$, where $\hat{m} = \hat{v}^{\varepsilon} \pmod N$
   - $\bar{b} = h_5^{\hat{b}} h_6^{w_3} \pmod q$, where $\hat{b} = g^y c \pmod N$
   - $\bar{c} = (h_1 h_3)^{\hat{a}} (h_1 h_5)^w \pmod q$

   $\mathcal{A}$ sends and commits the session values $(\hat{a}, \bar{m}, \bar{b}, \bar{c})$ to $\mathcal{R}$.

2. **Zero-Knowledge Proofs of Knowledge(ZKPoKs)**: once $\mathcal{A}$ has committed its session values, $\mathcal{A}$ and $\mathcal{R}$ engage in ZKPoK1-4 as follows.

   - **ZKPoK1**: $\mathcal{A}$ picks $s'_y, s'_1, s'_3 \in_R \mathbb{Z}_e^*$, and computes $t' = g^{s'_y} (g_1 g_2^{\alpha})^{s'_1} g_3^{s'_3} \pmod N$. Then, it sends t' to $\mathcal{R}$. After receiving γ from $\mathcal{R}$, $\mathcal{A}$ computes $s' = a^{\gamma} \pmod N$, and then sends (s'_y, s'_1, s'_3, s') to $\mathcal{R}$.

     Correctness: Since $\hat{a} = g_2^{-\beta} a^e \pmod N$, $t' = g^{s'_y} (g_1 g_2^{\alpha})^{s'_1} g_3^{s'_3} \pmod N$ and $s' = a^{\gamma} \pmod N$, we have that

$$\begin{aligned} (\hat{a} g_2^{-\beta})^{\gamma} t' &= ((g_2^{-\beta} a^e) g_2^{-\beta})^{\gamma} t' \\ &= a^{e\gamma} t' \\ &= a^{e\gamma} g^{s'_y} (g_1 g_2^{\alpha})^{s'_1} g_3^{s'_3} \\ &= g^{s'_y} (g_1 g_2^{\alpha})^{s'_1} g_3^{s'_3} s'^{e} \pmod N \end{aligned}$$

     Hence, ZKPoK1 is preformed successfully.

- **ZKPoK2**: $\mathcal{A}$ picks $s_1', s_2', s_3' \in_R \mathbb{Z}$, and computes $t' = h_2^{s_1'} h_4^{s_2'} h_6^{s_3'} \pmod q$. Then, it sends t' to $\mathcal{R}$. After receiving γ from $\mathcal{R}$, $\mathcal{A}$ computes $s_{\hat{b}}' = w\gamma$, and then sends $(s_{\hat{b}}', s_1', s_2', s_3')$ to $\mathcal{R}$.

  Correctness: Since $\bar{c} = (h_1 h_3)^{\hat{a}} (h_1 h_5)^w \pmod q$, $t' = h_2^{s_1'} h_4^{s_2'} h_6^{s_3'} \pmod q$ and $s_{\hat{b}}' = w\gamma$, we have that

$$\begin{aligned}
(\bar{c}(h_1 h_3)^{-\hat{a}})^\gamma t' &= ((h_1 h_3)^{\hat{a}} (h_1 h_5)^w (h_1 h_3)^{-\hat{a}})^\gamma t' \\
&= (h_1 h_5)^{w\gamma} t' \\
&= (h_1 h_5)^{w\gamma} h_2^{s_1'} h_4^{s_2'} h_6^{s_3'} \\
&= (h_1 h_5)^{s_{\hat{b}}'} h_2^{s_1'} h_4^{s_2'} h_6^{s_3'} \pmod q
\end{aligned}$$

  Hence, ZKPoK2 is preformed successfully.

- **ZKPoK3**: $\mathcal{A}$ picks $r_1', r_2' \in_R \mathbb{Z}_N$ and computes $t' = h_1^{r_1'^{\varepsilon}} h_2^{r_2'} \pmod q$. Then, it sends t' to $\mathcal{R}$. After receiving $\gamma \in \{0, 1\}$ from $\mathcal{R}$, $\mathcal{A}$ performs as follows.

  (a) if $\gamma = 0$, let $s_1' = r_1'$ and $s_2' = r_2'$.
  (b) if $\gamma = 1$, let $s_1' = r_1' \hat{v}^{-1} \pmod N$ and $s_2' = r_2' - w_1 (r_1' \hat{v}^{-1})^\varepsilon \pmod N$

  $\mathcal{A}$ sends (s_1', s_2') to $\mathcal{R}$.

  Correctness: Since $t' = h_1^{r_1'^{\varepsilon}} h_2^{r_2'} \pmod q$ and $\bar{m} = h_1^{\hat{m}} h_2^{w_1} \pmod q$, where $\hat{m} = \hat{v}^\varepsilon \pmod N$, we have that

  (a) if $\gamma = 0$, $t' = h_1^{r_1'^{\varepsilon}} h_2^{r_2'} = h_1^{s_1'^{\varepsilon}} h_2^{s_2'} \pmod q$
  (b) if $\gamma = 1$, we have that

$$\begin{aligned}
\bar{m}^{s_1'^{\varepsilon}} h_2^{s_2'} &= (h_1^{\hat{m}} h_2^{w_1})^{s_1'^{\varepsilon}} h_2^{s_2'} \\
&= (h_1^{\hat{m}} h_2^{w_1})^{(r_1' \hat{v}^{-1})^\varepsilon} h_2^{r_2' - w_1 (r_1' \hat{v}^{-1})^\varepsilon} \\
&= (h_1^{\hat{m}})^{(r_1' \hat{v}^{-1})^\varepsilon} h_2^{r_2'} \\
&= (h_1^{\hat{v}^\varepsilon})^{(r_1' \hat{v}^{-1})^\varepsilon} h_2^{r_2'} \\
&= h_1^{r_1'^{\varepsilon}} h_2^{r_2'} \\
&= t' \pmod q
\end{aligned}$$

  Hence, ZKPoK3 is preformed successfully.

- **ZKPoK4**: $\mathcal{A}$ picks $r_1', r_2' \in_R \mathbb{Z}_N$ and computes $t' = h_5^{g^{r_1'}} h_6^{r_2'} \pmod q$. Then, it sends t' to $\mathcal{R}$. After receiving $\gamma \in \{0, 1\}$ from $\mathcal{R}$, $\mathcal{A}$ performs as follows.

  (a) if $\gamma = 0$, let $s_1' = r_1'$ and $s_2' = r_2'$.
  (b) if $\gamma = 1$, let $s_1' = r_1' - y$ and $s_2' = r_2' - w_3 c^{-1} g^{s_1'}$

  $\mathcal{A}$ sends (s_1', s_2') to $\mathcal{R}$.

  Correctness: Since $t' = h_5^{g^{r_1'}} h_6^{r_2'} \pmod q$ and $\bar{b} = h_5^{\hat{b}} h_6^{w_3} \pmod q$, where $\hat{b} = g^y c \pmod N$, we have that

  (a) if $\gamma = 0$, $t' = h_5^{g^{r_1'}} h_6^{r_2'} = h_5^{g^{s_1'}} h_6^{s_2'} \pmod q$
  (b) if $\gamma = 1$, we have that

$$
\begin{aligned}
(\bar{b}^{c^{-1}})^{g^{s_1'}} h_6^{s_2'} &= ((h_5^{\hat{b}} h_6^{w_3})^{c^{-1}})^{g^{s_1'}} h_6^{s_2'} \\
&= ((h_5^{\hat{b}} h_6^{w_3})^{c^{-1}})^{g^{s_1'}} h_6^{r_2' - w_3 c^{-1} g^{s_1'}} \\
&= h_5^{\hat{b} c^{-1} g^{s_1'}} h_6^{r_2'} \\
&= h_5^{(g^y c) c^{-1} g^{s_1'}} h_6^{r_2'} \\
&= h_5^{g^y g^{s_1'}} h_6^{r_2'} \\
&= h_5^{g^{y+(r_1'-y)}} h_6^{r_2'} \\
&= h_5^{g^{r_1'}} h_6^{r_2'} \\
&= t' \pmod{q}
\end{aligned}
$$

Hence, ZKPoK4 is preformed successfully.

Since all ZKPoK1-4 are successful, $\mathcal{R}$ authenticates the adversary $\mathcal{A}$ as a legitimate user, and then collects data from it. Hence, the protocol is broken. □

Another main flaw of Alcaide et al.'s protocol is given below.

In *User Registration* phase, since the new user $\mathcal{U}_{new}$ sends its certificate $\tilde{x}$ in plaintext to the founders, $\tilde{x}$ can be captured by $\mathcal{A}$. Then, $\mathcal{A}$ sends $(\tilde{x}, a', b)$ to the founders by impersonating $\mathcal{U}_{new}$, where a' and b are computed by $\mathcal{A}$ itself. Since the founders are user nodes, it is reasonable to assume that they are stateless. Hence, they do not know whether $\tilde{x}$ belongs to a user who has already received its AAC. So $\mathcal{A}$ can obtain $(t+1)$ partial signatures from the founders, then it can construct a valid AAC. That is, once a user's certificate is sent in plaintext in *User Registration* phase, the adversary $\mathcal{A}$ can obtain this certificate and forge a valid AAC by impersonating this user. With this AAC, $\mathcal{A}$ can cheat the data collectors in *Credentail Proving* phase. □

4 Conclusion

Recently, Alcaide et al. proposed a fully decentralized anonymous authentication protocol for privacy-preserving IoT target-driven applications. Their system is set up by an ad-hoc community of decentralized founding nodes. Nodes can interact, being participants of cyberphysical systems, preserving full anonymity. In this study, we point out that their protocol is insecure. The adversary can cheat the data collectors by impersonating a legitimate user.

References

[1] A.Alcaide, E.Palomar, J.Montero-Castillo and A.Ribagorda. Anonymous authentication for privacy-preserving IoT target-driven applications. Computers & Security. 37 (2013). pp:111-123.