Modeling and Verifying AUPS Using CSP

Hongqin Zhang, Huibiao Zhu*, Jiaqi Yin, Ningning Chen
Shanghai Key Laboratory of Trustworthy Computing,
East China Normal University, Shanghai, China

***Abstract*—The Internet of Things (IoT) is an important technology in IT industries. The wide adoption of IoT raises concerns about security and privacy. The Authenticated Publish/Subscribe (AUPS) model is an IoT system which aims to address the security and privacy issues in the IoT environment. AUPS is attracting more and more attention from industries. Hence, the reliability of AUPS is worth investigating.**

In this paper, we model AUPS using Communicating Sequential Processes (CSP). Five properties (Deadlock Freedom, Data Availability, Data Leakage, Device Faking and User Privacy Leakage) of the model are verified by utilizing the model checker Process Analysis Toolkit (PAT). The verification results demonstrate that AUPS cannot ensure the security of critical data. To solve the problem, we improve the model by using a digital certificate. The verification results of the improved model indicate that our study can enhance the security and reliability of the AUPS model.

***Index Terms*—AUPS, CSP, PAT, Modeling, Verifying**

I. INTRODUCTION

As an important paradigm in IT industries, the Internet of Things (IoT) connects heterogeneous devices to provide users with required services [1]. Communication efficiency, data security and user privacy are the three major issues of IoT [2]. In order to cope with these issues, several solutions have been proposed [3]–[5]. Shi et al. came up with an IoT system [3] which used a machine learning method to improve the efficiency of data processing. Jung et al. proposed a distributed IoT scheme where two or more servers created a group and a client viewed the group as a powerful server [4]. This scheme reduced network latency. However, it cannot protect user privacy. Shang et al. presented a publish/subscribe IoT framework [5]. It adopted data authentication to improve the communication security without considering user privacy issues. These solutions improved communication efficiency. However, they did not fully explore security and privacy issues.

Thus, a publish/subscribe IoT system called Authenticated Publish/Subscribe (AUPS) [6] was proposed. AUPS adopted Attribute-based Access Control (ABAC) [7] to improve the security and privacy of the system. ABAC controls access to data by evaluating rules against the attributes of users. According to the experimental results [6], AUPS was more efficient than the other existing secure solutions. As AUPS is attracting more and more attention from industries, we believe that it is valuable to analyse the functional and security properties of AUPS using formal methods.

In this paper, AUPS is formally modeled using the process algebra CSP [8]. The model checking tool PAT [9] is adopted to verify its functional and security properties. According to the verification results, AUPS may cause data leakage, device faking and user privacy leakage once intruders appear. Thus, we improve the model by using a digital certificate. Then we verify the improved model using PAT. The verification results show that our work can guarantee the security of AUPS.

The rest of this paper is organized as follows. Section II briefly introduces AUPS and CSP. Section III is devoted to the modeling of AUPS. In Section IV, we analyse the verification results and give the improvement that can address the vulnerabilities of the model. Finally, conclusion and future work are given in Section V.

II. BACKGROUND

In this section, we give a brief description of AUPS. After that, we introduce the process algebra CSP.

A. AUPS

As shown in Fig. 1, the Authenticated Publish/Subscribe (AUPS) contains the following entities:

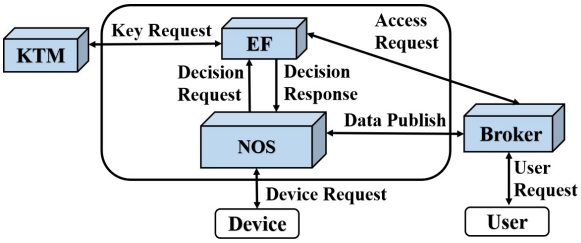

Fig. 1: AUPS schema

- **Device:** It collects data from the environment and sends them to the Networked Smart Object (NOS).
- **Networked Smart Object (NOS):** The NOS deals with the collected data and publishes them to the broker.
- **User:** The user subscribes to services to get the required data by interacting with the broker.
- **Broker:** It handles the subscription requests from users, and then forwards data to the legal subscribers.
- **Enforcement Framework (EF):** The EF is a policy enforcement framework. It performs access control and manages the data encryption/decryption keys using a key table. The structure of key table is shown in TABLE I.
- **Keys Topics Manager (KTM):** It generates the data encryption/decryption keys according to the services.

The core functions of AUPS are data publishing and service subscription. Data publishing represents that the devices publish collected data to the Internet. Service subscription means

*Corresponding author: hbzhu@sei.ecnu.edu.cn (H. Zhu).

DOI reference number: 10.18293/SEKE2022-088

TABLE I: Key table

Field	Description
id	The identifier of the corresponding key
key	The actual key
val	The expiration date of the key
atb	The attribute(s) owned by users allowed to get the data

that users subscribe to services to obtain the required data. We introduce the actions of the two functions as follows. The related notations and descriptions are listed in TABLE II.

TABLE II: Notations and descriptions

Notation	Description
$puk_x/prk_x/sk_x$	Public/Private/Symmetric key of the user/device/ broker/NOS/intruder, $x \in \{u, d, b, n, i\}$
id_x/sid_x	Identity information/Session identifier of the user/ device, $x \in \{u, d\}$

Before publishing data, the device must finish device registration. The related actions are shown in Fig. 2.

- $a1$: A device sends a registration request req_d to the NOS.
- $a2$: When receiving the request, the NOS sends its public key puk_n to the device.
- $a3$: The device sends its identity information id_d and symmetric key sk_d encrypted with puk_n to the NOS.
- $a4$: The NOS decrypts the message to get id_d and sk_d using its private key prk_n, and then distributes a session identifier sid_d encrypted with sk_d to the device.
- $a5$: The device sends the collected data d and session identifier sid_d encrypted with sk_d to the NOS.
- $a6$: The NOS decrypts the message to get the data d using sk_d, and then asks the EF for a data encryption key.
- $a7$: The EF sends a key kT to the NOS. After that, the NOS can publish data encrypted with kT to the broker.

Before getting data, the user needs to register and subscribe to relevant services. The related actions are given in Fig. 3.

- $b1$: A user sends a registration request req_u to the broker.
- $b2$: The broker sends its public key puk_b to the user.
- $b3$: The user sends its private information id_u and symmetric key sk_u encrypted with puk_b to the broker.
- $b4$: The broker decrypts the message to get id_u and sk_u using prk_b, and then sends the session identifier sid_u and attribute at to the user. Notice that sid_u is used to identify the user without revealing its sensitive information.
- $b5$: The user sends a subscription request $reqT$ encrypted with sk_u to the broker.
- $b6$: The broker verifies the identity of the user, and then sends an access control request to the EF.
- $b7$: The EF checks whether the user can access the data of service T. If the result is positive, the EF forwards the data decryption key kT to the broker.
- $b8$: The broker sends kT encrypted with sk_u to the user. Finally, the user can obtain the data using kT.

B. CSP

Communicating Sequential Processes (CSP) is a process algebra proposed by C. A. R. Hoare [8]. Here we briefly introduce the syntax of CSP used in this paper.

$$P, Q ::= a \to P \mid c?x \to P \mid c!v \to P \mid P; Q \mid P||Q \mid P \square Q \mid P \lhd b \rhd Q \mid P[[a \leftarrow b]]$$

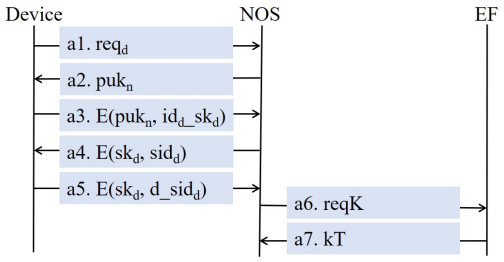

Fig. 2: Data publishing

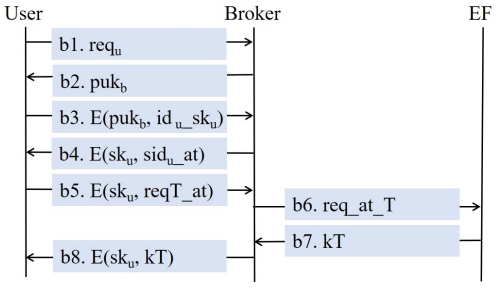

Fig. 3: Service subscription

- $a \to P$ indicates that a process performs action a first, and then acts like process P.
- $c?x \to P$ represents that a process receives a message via channel c and assigns the received message to x, and then behaves like process P.
- $c!v \to P$ denotes that message v is sent through channel c, and then process P is executed.
- $P; Q$ is the sequential execution of processes P and Q.
- $P||Q$ describes that processes P and Q run in parallel.
- $P \square Q$ stands for the general choice of processes P and Q, and the selection is made by the environment.
- $P \lhd b \rhd Q$ shows that if the condition b is true, process P is executed, otherwise process Q is executed.
- $P[[a \leftarrow b]]$ means renaming action. Event a in process P is replaced by event b.

III. MODELING

In this section, we focus on the modeling of AUPS. First, we introduce some preparatory notations for the modeling including sets, messages and channels. Based on these notations, we give the formal model of AUPS.

A. Sets, Messages and Channels

First, we explain the sets defined in our model. **Entity** is a set of entities described in Section II. **Req** set contains requests of entities. **Key** set is composed of all the keys. **Data** set contains the data collected by devices. **Inf** set denotes other message contents including identifier set ID, feedback message set Ack, attribute set Atb and service set $Service$.

Besides, we define the encryption function E and decryption function D to model the messages:

E(k, m); D(k, E(k, m)); D(k^{-1}, E(k,m))

Function $E(k, m)$ means that we encrypt the message m using k. $D(k, E(k, m))$ denotes that we use a symmetric key

k to decrypt the message which is encrypted by k. D(k^{-1}, E(k,m)) indicates that we use the corresponding decryption key k^{-1} to decrypt the message encrypted by k.

Based on the sets and functions defined above, we abstract and classify the messages as follows:

$$MSG_{req} = \{msg_{req}.a.b.req, msg_{req}.a.b.E(k, req) \mid a, b \in Entity, k \in Key, req \in Req\}$$
$$MSG_{key} = \{msg_{key}.a.b.E(k_1, k) \mid a, b \in Entity, k_1, k \in Key\}$$
$$MSG_{inf} = \{msg_{inf}.a.b.inf \mid a, b \in Entity, inf \in Inf\}$$
$$MSG_{data} = \{msg_{data}.a.b.d, msg_{data}.a.b.E(k, d) \mid a, b \in Entity, d \in Data, k \in Key\}$$
$$MSG_{in} = \{msg_{req1}.t, msg_{key1}.k \mid t \in Service, k \in Key\}$$
$$MSG_{out} = MSG_{req} \cup MSG_{inf} \cup MSG_{key} \cup MSG_{data}$$
$$MSG = MSG_{out} \cup MSG_{in}$$

MSG_{req} represents the set of request messages. MSG_{key} denotes the set of messages containing the keys. MSG_{inf} involves messages containing identifiers, feedback messages, attributes and services. MSG_{data} consists of messages containing the data collected by devices. MSG_{out} means the set of messages transmitted between entities. MSG_{in} consists of the internal processing messages of entities. MSG is composed of all the messages in the model.

Then we give the definitions of communication channels:

- Channels between honest entities shown by ***COM_PATH***: $ComDN,\ ComNB,\ ComUB,\ ComBE,\ ComNE,\ GetE,\ ComEK$
- Channels for intruders to intercept or fake the transmitted messages denoted by ***INTRUDER_ PATH***: $FakeDN,\ FakeND,\ FakeUB,\ FakeBU$

The declaration of the channels is given as follows:

$$Channel\ COM\_PATH, INTRUDER\_PATH : MSG$$

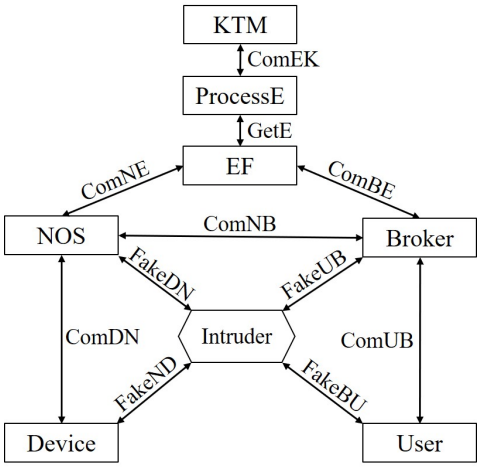

Fig. 4: Channels of AUPS model

B. Overall Modeling

In this subsection, we give the whole model of AUPS. $System_0$ only contains legal entities which are running in parallel. Based on the model $System_0$, we construct the model of $System$ by introducing attacks from intruders.

$$System_0 =_{df} Broker \| User \| EF \| ProcessE \| KTM \| Device \| NOS \| Clock$$
$$System =_{df} System_0 [| INTRUDER\_PATH |] Intruder$$

$Broker$, $User$, EF, $Device$, NOS and KTM are processes describing the behavior of the broker, user, EF, device, NOS and KTM respectively. $ProcessE$ denotes the internal processing procedures of the EF. The process $Clock$ is used to realize the synchronization of time. In addition, the process $Intruder$ simulates the actions of intruders. The channels between all processes are shown in Fig. 4.

C. Clock Modeling

AUPS adopts a temporary key to encrypt the published data. Before using the temporary key, the entity needs to check the expiration date of the key. Hence, we define the process $Clock$ to realize the synchronization of all entities. The process $Clock$ serves to record the time and return the current time whenever the entities want it.

$$Clock(t) =_{df} tick \rightarrow Clock(t+1) \ \Box \ Time!t \rightarrow Clock(t)$$

D. User Modeling

We formalize the process $User_0$ to describe the behavior of the user without intruders as follows:

$$\begin{aligned} User_0 =_{df}\ & ComUB!msg_{req}.U.B.req_u \rightarrow \\ & ComUB?msg_{key}.B.U.puk_b \rightarrow \\ & ComUB!msg_{req}.U.B.E(puk_b, id_u.sk_u) \rightarrow \\ & ComUB?msg_{inf}.B.U.E(sk_u, sid_u.at) \rightarrow \\ & \left(\begin{array}{l} ComUB!msg_{req}.U.B.E(sk_u, reqT.at) \rightarrow \\ ComUB?msg_{key}.B.U.E(sk_u, kT) \rightarrow \\ \left(\begin{array}{l} ComUB?msg_{data}.B.U.E(kT, d) \rightarrow \\ User_0 \lhd D(kT, E(kT, d)) \rhd (fail \rightarrow User_0) \end{array} \right) \\ \lhd D(sk_u, E(sk_u, kT)) \rhd (fail \rightarrow User_0) \end{array} \right) \\ & \lhd D(sk_u, E(sk_u, sid_u.at)) \rhd (fail \rightarrow User_0) \end{aligned}$$

The above actions correspond to $b1 - b5$ and $b8$ in Fig. 3. First, the user sends a registration request req_u to the broker and receives the broker's public key puk_b. Then the user requests an attribute by sending its identity information id_u and symmetric key sk_u encrypted with puk_b to the broker. Once getting the attribute at and session identifier sid_u, the user sends a subscription request $reqT$ encrypted with sk_u to the broker. If the request is accepted, the user receives the data decryption key kT and encrypted data. Finally, the user can obtain the required data d using kT. Then we consider attacks from intruders.

Based on the achieved model $User_0$, we formalize the process $User$ with intruders via renaming as follows:

$$\begin{aligned} User =_{df} & User_0[[\\ & ComUB!\{|ComUB|\} \leftarrow ComUB!\{|ComUB|\}, \\ & ComUB!\{|ComUB|\} \leftarrow FakeUB!\{|ComUB|\}, \\ & ComUB?\{|ComUB|\} \leftarrow ComUB?\{|ComUB|\}, \\ & ComUB?\{|ComUB|\} \leftarrow FakeBU?\{|ComUB|\}]] \end{aligned}$$

$\{|ComUB|\}$ represents the set of all communications over the channel $ComUB$. The first two lines mean that whenever $User_0$ transmits a message on the channel $ComUB$, $User$

can transmit the same message on the channel $ComUB$ or $FakeUB$. The last two lines are similar.

E. Broker Modeling

We give the model of process $Broker_0$ to describe the behavior of the broker without intruders as follows:

$$
\begin{array}{l}
Broker_0 =_{df} ComUB?msg_{req}.U.B.req_u \rightarrow \\
\quad ComUB!msg_{key}.B.U.puk_b \rightarrow \\
\quad ComUB?msg_{req}.U.B.E(puk_b, id_u.sk_u) \rightarrow \\
\left(\begin{array}{l}
ComUB!msg_{inf}.B.U.E(sk_u, sid_u.at) \rightarrow \\
ComUB?msg_{req}.U.B.E(sk_u, reqT.at) \rightarrow \\
\left(\begin{array}{l}
ComBE!msg_{req}.B.E.req.at.T \rightarrow \\
ComBE?msg_{inf}.E.B.ack.kT \rightarrow \\
\left(\begin{array}{l}
ComUB!msg_{key}.B.U.E(sk_u, kT) \rightarrow \\
ComNB?msg_{data}.N.B.E(kT, d) \rightarrow \\
ComUB!msg_{data}.B.U.E(kT, d) \rightarrow \\
Broker_0 \lhd (ack == true) \rhd \\
(fail \rightarrow Broker_0)
\end{array} \right) \\
\lhd D(sk_u, E(sk_u, reqT.at)) \rhd (fail \rightarrow Broker_0)
\end{array} \right) \\
\lhd D(prk_b, E(puk_b, id_u.sk_u)) \rhd (fail \rightarrow Broker_0)
\end{array} \right)
\end{array}
$$

The above actions correspond to $b1 - b8$ in Fig. 3. During user registration, the broker sends its public key puk_b to the user, and then obtains the user's identity information id_u and symmetric key sk_u through decryption. After that, the broker distributes the attribute at and session identifier sid_u encrypted with sk_u to the user. When receiving the subscription request $reqT$, the broker verifies the user's identity by sending the attribute at and service T to the EF. If the feedback message ack from the EF is true, the broker sends the data decryption key kT encrypted with sk_u to the user. Otherwise, the subscription request is rejected.

Based on $Broker_0$, the model of the process $Broker$ with intruders can be acquired via renaming similar to the process $User$. We leave out the details.

F. EF Modeling

The EF interacts with the broker and NOS to perform access control. We model the process EF using general choice $\Box$.

$$
\begin{array}{l}
EF =_{df} ComBE?msg_{req}.E.B.req.at.T \rightarrow \\
\quad ack := check(at, T) \rightarrow \\
\left(\begin{array}{l}
GetE!msg_{req1}.T \rightarrow GetE?msg_{key1}.kT \rightarrow \\
ComBE!msg_{key}.E.B.ack.kT \rightarrow EF_0 \\
\lhd (ack == true) \rhd (fail \rightarrow EF_0)
\end{array} \right) \\
\Box ComNE?msg_{req}.N.E.reqK \rightarrow GetE!msg_{req1}.T \rightarrow \\
\quad GetE?msg_{key1}.kT \rightarrow ComNE!msg_{key}.E.N.kT \rightarrow EF_0
\end{array}
$$

The model before $\Box$ describes the communication between the EF and broker. $check(at, T)$ is a function used to verify whether the user with attribute at can access service T. After receiving the broker's request req, the EF adopts $check(at, T)$ to verify the access authority of the user. If the result ack is true, the EF sends a key request to its internal process $ProcessE$. Once receiving the data decryption key kT from $ProcessE$, the EF forwards kT to the broker. These actions correspond to $b6$ and $b7$ in Fig. 3. The model after $\Box$ describes the communication between the EF and NOS. When receiving the NOS's request $reqK$, the EF requests a data encryption key from $ProcessE$, and then forwards the key to the NOS. The related actions are illustrated by $a6$ and $a7$ in Fig. 2.

G. ProcessE Modeling

In order to simulate the internal process of EF, we model $ProcessE$. It mainly deals with the key requests of entities.

$$
\begin{array}{l}
ProcessE =_{df} GetE?msg_{req1}.T \rightarrow kT := findKey(T) \rightarrow \\
\left(\begin{array}{l}
GetE!msg_{key1}.kT \rightarrow ProcessE \\
\lhd (\exists e \in table_i \bullet e.key == kT \wedge Time?t \rightarrow e.val > t) \rhd \\
ComEK!msg_{req}.E.K.reqEK.T \rightarrow ComEK?msg_{key}. \\
K.E.kT \rightarrow GetE!msg_{key1}.kT \rightarrow ProcessE
\end{array} \right)
\end{array}
$$

$findKey(T)$ is a function designed to find the symmetric key that can encrypt or decrypt the data of service T. After receiving the key request from the EF, $ProcessE$ adopts $findKey(T)$ to find a symmetric key kT. Then $ProcessE$ verifies whether kT is valid by checking the expiration date val. If val is later than the current system time, it means that kT has not expired. Then $ProcessE$ sends kT to the EF. If kT has expired, $ProcessE$ requests a new key from the KTM, and then forwards the new key to the EF.

Similarly, we can define CSP processes representing the device, NOS and KTM. The actions of them are introduced in Section II. We omit the details of these processes here.

H. Intruder Modeling

In order to simulate the attacks from the real environment, we model the $Intruder$ process. It can intercept and fake the messages on channel $ComDN$, $ComNB$ and $ComUB$.

First, we define the set of facts that the intruder can learn.

$$Fact =_{df} Entity \cup MSG_{out} \cup \{sk_i, puk_i, prk_i\}$$

Through the known facts, the intruder can deduce new facts. The symbol $F \mapsto f$ means that the intruder can deduce a fact f from the fact set F.

$$
\begin{array}{l}
\{k, c\} \mapsto E(k, c) \\
\{k^{-1}, E(k, c)\} \mapsto c, \quad \{sk, E(sk, c)\} \mapsto c \\
F \mapsto f \wedge F \subseteq F' \Longrightarrow F' \mapsto f
\end{array}
$$

The first rule means encryption. The second and third rules denote the decryption in asymmetric and symmetric encryption forms respectively. The last rule shows that if the fact f can be derived from a fact set F, and F is a subset of F', then the intruder can also deduce f from the larger set F'.

Moreover, we use a function $Info(m)$ to imply the facts that the intruder can learn through intercepted messages.

$$
\begin{array}{l}
Info(msg_{key}.a.b.E(k_1, k)) =_{df} \{a, b, E(k_1, k)\} \\
Info(msg_{data}.a.b.E(k, d)) =_{df} \{a, b, E(k, d)\}
\end{array}
$$

Besides, we introduce a channel $DEDUCE$ for the intruder to deduce new facts. Its definition is given as below:

$$\text{Channel } DEDUCE : Fact.P(Fact)$$

Then the process $Intruder_0$ can be modeled as follows:

$$
\begin{array}{l}
Intruder_0(F) \\
=_{df} \Box_{m \in MSG_{out}} Fake.m \rightarrow Intruder_0(F \cup Info(m)) \\
\quad \Box\Box_{f \in Fact, f \notin F, F \mapsto f} Init\{dl = false\} \rightarrow Deduce.f.F \\
\quad \rightarrow \left(\begin{array}{l}
(\ dl := true \rightarrow Intruder_0(F \cup \{f\})\ ) \\
\lhd (f == d) \rhd \\
(\ dl := false \rightarrow Intruder_0(F \cup \{f\})\ )
\end{array} \right)
\end{array}
$$

When intercepting a message m, the intruder adds $Info(m)$ to its knowledge. If the intruder can decrypt m, it can falsify m and send the modified message to the original receiver.

If the receiver does not recognize that the message has been modified, it means that the intruder successfully fakes as the original sender. Furthermore, the intruder can deduce new facts from its knowledge via the channel $DEDUCE$ and add them to its knowledge. Once the intruder deduces the published data successfully, data leakage occurs. id_u represents the identity information of the user, such as name and address. If the intruder deduces the user's sensitive information id_u, user privacy leakage happens. Now we give the model of $Intruder$. The parameter IK is the initial knowledge of the intruder.

$$Intruder =_{df} Intruder_0(IK)$$
$$where,\ IK =_{df} Entity \cup \{sk_i, puk_i, prk_i\}$$

IV. VERIFICATION AND IMPROVEMENT

In this section, we verify several functional and security properties of AUPS. Based on the verification results and analysis of attacks, we improve the original model and give the new verification results of the improved model.

A. Properties Verification

We use Linear Temporal Logic (LTL) formulas to describe five properties of AUPS. $System()$ denotes the model with intruders. By using the assertion $\#assert\ System() \mid = F$ in PAT, we verify whether the model satisfies the formula F.

Property 1: Deadlock Freedom

The system should not run into a deadlock state. We verify this property by means of a primitive in PAT.

$$\#assert\ System()\ deadlockfree;$$

Property 2: Data Availability

The property means that legal users should be able to obtain the required data. We define a Boolean variable $data\_suc$ to verify this property. When the subscriber gets the required data, we set the value of $data\_suc$ to $true$.

$$\#define\ \ Data\_Available\ \ data\_suc == true;$$
$$\#assert\ \ System()\ \ reaches\ \ Data\_Available;$$

Property 3: Data Leakage

Data leakage can cause a bad effect to the system. We use a Boolean variable dl to verify the property. If the intruder obtains the data, we set the value of dl to $true$.

$$\#define\ \ Data\_Leak\_Success\ \ dl == true;$$
$$\#assert\ \ System()\ \ \mid = []!\ \ Data\_Leak\_Success;$$

Property 4: Device Faking

The property means that the intruder can pretend to be a legal device without being recognized. We adopt a Boolean variable df to verify the property. If the intruder fakes as a legal device successfully, we set the value of df to $true$.

$$\#define\ \ Device\_Fake\_Success\ \ df == true;$$
$$\#assert\ \ System()\ \ \mid = []!\ \ Device\_Fake\_Success;$$

Property 5: User Privacy Leakage

User privacy leakage may bring great security risks to users. Hence, we check whether the intruder can obtain the sensitive information of the user using the following assertion.

$$\#define\ \ User\_Privacy\_Leak\ \ pl == true;$$
$$\#assert\ \ System()\ \ \mid = []!\ \ User\_Privacy\_Leak;$$

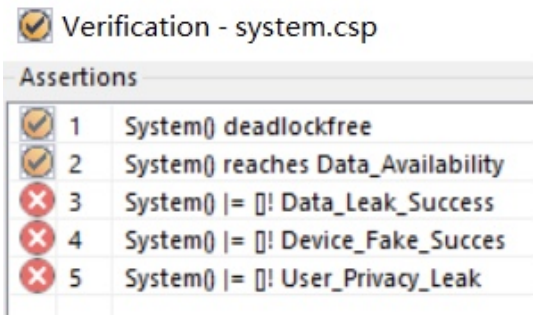

Fig. 5: Verification results of the original model

B. Verification Results

The verification results are shown in Fig. 5:

- $Property\ 1$ is valid. It represents that the model will never run into a deadlock state.
- $Property\ 2$ is valid. It shows that the data can be transmitted to the legal subscribers.
- $Property\ 3$ is invalid. It indicates that the intruder can obtain the data illegally.
- $Property\ 4$ is invalid. It means that the intruder can pretend to be a legal device to publish fake data.
- $Property\ 5$ is invalid. It indicates that the model cannot protect the user privacy once intruders appear.

C. Attack Analysis

According to the verification results, although AUPS adopts the access control and temporary keys, the system is still unreliable. Now we discuss the reasons for the insecure results. When the broker sends puk_b to the user, the intruder can intercept the message, and then replace puk_b with its public key puk_i. Since the user cannot detect that the key has been changed, the user sends sk_u and id_u encrypted with puk_i to the broker. Then the intruder can decrypt the message with prk_i to obtain sk_u and the user's sensitive information id_u, which leads to user privacy leakage. After obtaining sk_u, the intruder can get the data decryption key. Finally, the intruder can acquire the data, which results in data leakage. We give an example of the related attacks as follows:

$$A1.\ U \longrightarrow I : U.B.req_u$$
$$A2.\ I \longrightarrow B : U.B.req_u$$
$$A3.\ B \longrightarrow I : B.U.puk_b$$
$$A4.\ I \longrightarrow U : B.U.puk_i$$
$$A5.\ U \longrightarrow I : U.B.E(puk_i, sk_u.id_u)$$
$$A6.\ I \longrightarrow B : U.B.E(puk_b, sk_u.id_u)$$
$$A7.\ B \longrightarrow I : B.U.E(sk_u, sid_u.at)$$
$$A8.\ I \longrightarrow U : B.U.E(sk_u, sid_u.at)$$
$$A9.\ U \longrightarrow I : U.B.E(sk_u, at.T))$$
$$A10.\ I \longrightarrow B : U.B.E(sk_u, at.T))$$
$$A11.\ B \longrightarrow I : B.U.E(sk_u, kT)$$

where U, I and B mean user, intruder and broker respectively.

- $A1$: The user sends a request req_u to the broker.
- $A2$: The intruder intercepts the request.
- $A3$: The broker sends its public key puk_b to the user.

- $A4$: The intruder intercepts the message, and then replaces puk_b with its own public key puk_i.
- $A5$: The user sends its symmetric key sk_u and private information id_u encrypted with puk_i to the broker.
- $A6$: The intruder intercepts the message, and then decrypts the message to obtain sk_u and id_u using prk_i. At this point, user privacy leakage occurs.
- $A7$: The broker distributes the session identifier sid_u and attribute at encrypted with sk_u to the user.
- $A8$: The intruder acquires sid_u and at using sk_u.
- $A9$: The user requests to subscribe to service T.
- $A10$: The intruder eavesdrops on the message.
- $A11$: The broker distributes the data decryption key kT encrypted with sk_u to the user. The intruder intercepts the message and gets kT using sk_u. Then, the intruder can obtain the data using kT, which results in data leakage.

Similarly, the intruder can obtain the session identifier sid_d and symmetric key sk_d, and then fake as the device to publish data, which leads to device faking. We omit the details here.

D. Improved Model and Verification

In order to address the above issues, we improve the model by adding a digital certificate. Before sending the public key to other entities, the entity needs to send its public key to the Certification Authority (CA) to apply for a certificate. Fig. 6 depicts the flows of the digital certificate. First, the sender applies for a certificate. Second, CA generates a certificate based on the information of the sender, and then transmits the certificate to the receiver. Finally, the receiver verifies the validity of the certificate.

As the certificate is encrypted by CA's private key, the intruder cannot fake the certificate. It means that the intruder cannot replace the public keys of the honest entities with its own public key. Thus, the intruder can neither get the data nor violate user privacy. We modify the message definitions of the model. MSG_{key} is replaced by the following MSG_{key2}.

$$MSG_{key2} = \{msg_{key2}.a.b.E(k_1, k.inf) \mid a, b \in Entity, k_1, k \in Key, inf \in Inf\}$$

Then we formalize the improved processes of $Broker_1$, $User_1$, $Device_1$ and NOS_1 using the new message definitions. The improved model is given as follows.

$$System_1 =_{df} Broker_1 \| User_1 \| EF \| ProcessE \| KTM \| Device_1 \| NOS_1 \| CA \| Clock$$

$$System =_{df} System_1 [| INTRUDER\_PATH |] Intruder$$

The verification results are shown in Fig. 7. $Property\ 3-5$ are valid. It means that $Data\ Leakage$, $Device\ Faking$ and $User\ Privacy\ Leakage$ problems are solved now.

V. CONCLUSION AND FUTURE WORK

AUPS is an IoT system based on the publish/subscribe paradigm. In this paper, we formalized AUPS using the process algebra CSP. Feeding the model into PAT, we verified several functional and security properties of the model including deadlock freedom, data availability, data leakage, device faking and user privacy leakage. According to the verification results, data leakage, device faking and user privacy leakage may occur once intruders appear. Hence, we improved the model by using a digital certificate. Then we verified the improved model with PAT. The verification results show that the improved model can prevent intruders from invading the system. In the future, we will study more security properties of AUPS using formal methods and improve our model to handle more attacks.

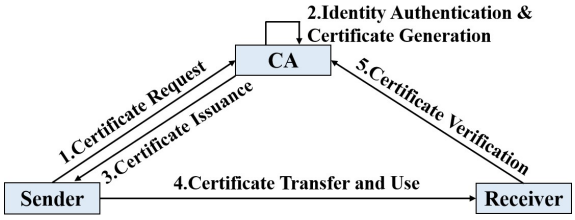

Fig. 6: Flows of digital certificate

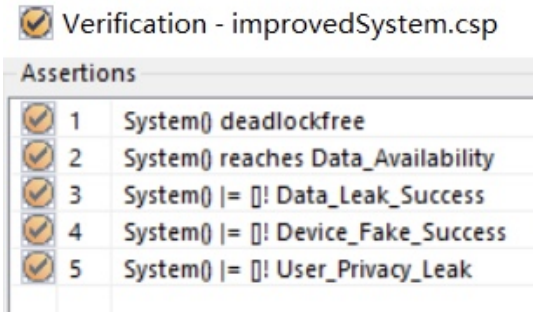

Fig. 7: Verification results of the original model

Acknowledgements. This work was partly supported by the National Key Research and Development Program of China (Grant No. 2018YFB2101300), the National Natural Science Foundation of China (Grant Nos. 61872145, 62032024), Shanghai Trusted Industry Internet Software Collaborative Innovation Center, and the Dean's Fund of Shanghai Key Laboratory of Trustworthy Computing (East China Normal University).

REFERENCES

[1] Tewari A, Gupta B B. Security, privacy and trust of different layers in Internet-of-Things (IoTs) framework. Future Gener. Comput. Syst. 2020, 108: 909-920.

[2] Khan F I, Hameed S. Understanding Security Requirements and Challenges in Internet of Things (IoTs): A Review. ArXiv abs/1808.10529 2019.

[3] Shi Y, Zhang Y, et al. Using machine learning to provide reliable differentiated services for IoT in SDN-like Publish/Subscribe middleware. Sensors, 2019, 19(6): 1449.

[4] Jung J, Choi Dong, et al. Distributed pub/sub model in CoAP-based Internet-of-Things networks. *Proc.* of the International Conference on Information Networking (ICOIN), 2018: 657-662.

[5] Shang W, Gawande A, et al. Publish-Subscribe Communication in Building Management Systems over Named Data Networking. *Proc.* of the 28th International Conference on Computer Communication and Networks, 2019: 1-10.

[6] Rizzardi A, Sicari S, et al. AUPS: An Open Source AUthenticated Publish/Subscribe system for the Internet of Things. Inf. Syst. 2016, 62: 29-41.

[7] Hu V C, Kuhn D R, et al. Attribute-based access control. Computer, 2015, 48(2): 85-88.

[8] Hoare C A R. Communicating sequential processes. Communications of the ACM, 1978, 21(8): 666-677.

[9] PAT, PAT: Process Analysis Toolkit. 2019. http://pat.comp.nus.edu.sg.