Formal Modeling and Verification of ICN-IoT Middleware Architecture

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

***Abstract*—As a key technology of the Internet of Things (IoT), middleware plays an important role in managing virtualized resources and services. However, traditional Internet architectures cannot ensure adequate data security and efficient data delivery for IoT middlewares. Therefore, Information-Centric Networking (ICN), a paradigm of the future network, is introduced into IoT middlewares. Since ICN-IoT middleware is attracting more and more attentions, its security is worth discussing.**

In this paper, we adopt Communicating Sequential Processes (CSP) to model the ICN-IoT middleware architecture. Five properties (deadlock freedom, data availability, action keys leakage, device faking and user faking) of the model are verified by utilizing the model checker Process Analysis Toolkit (PAT). According to the verification results, the model cannot guarantee the security of data. To solve the problems, we encrypt messages with the receiver's public key, and improve the model by introducing a method similar to the digital signature. The new verification results demonstrate that our study can assure the security of the ICN-IoT middleware architecture.

***Index Terms*—ICN, IoT Middleware, CSP, PAT, Modeling, Verification**

I. INTRODUCTION

The Internet of Things (IoT) [1] is an emerging paradigm, which connects heterogeneous devices with the Internet. As a crucial technology of IoT, IoT middleware [2] manages the integration of devices and provide interested users IoT services. The effectiveness of information retrieval and security of the transmitted information are two key challenges of IoT middlewares. To cope with the challenges, several IoT middleware solutions have been proposed [3]–[5]. Park et al. put forward a cloud-based middleware for self-adaptive IoT collaboration services, which improved the feasibility and performance of IoT systems [3]. Sicari et al. proposed a quality-aware IoT architecture aiming to deal with the data security and quality [4]. Shi et al. came up with an SDN-like publish/subscribe middleware architecture [5]. It used a machine learning method based on the eXtreme Gradient Boosting (XGBoost) model to improve the efficiency of IoT systems [5]. However, the above solutions cannot support the effectiveness and security of IoT systems at the same time. Hence, a solution called ICN-IoT middleware architecture [6] was proposed by introducing Information-Centric Networking (ICN) [7] into IoT. ICN identifies a network object by the name instead of the IP address, which supports content-oriented security and effectiveness [8]. Whereas, there are few works on the verification of the ICN-IoT middleware architecture.

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

In this paper, the ICN-IoT middleware architecture is formally modeled using the process algebra CSP [9]. Model checking tool PAT [10] is adopted to verify its functional and security properties. The verification results demonstrate that the architecture may cause action keys leakage and device faking. Therefore, we improve the original architecture by encrypting the messages with the receiver's public key to protect action keys, and introduce a method similar to digital signature to avoid device faking. Then we verify the improved architecture using PAT. The new verification results show that our work can enhance the security of the architecture.

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

II. BACKGROUND

In this section, we give a brief description of the ICN-IoT middleware architecture. After that, we introduce the syntax of the process algebra CSP.

A. *ICN-IoT Middleware Architecture*

ICN-IoT middleware architecture is designed to build a unified IoT platform using ICN. The schema of the architecture is illustrated in Fig. 1.

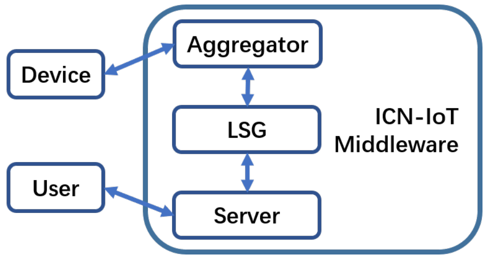


Fig. 1: ICN-IoT middleware architecture (simplified from [7])

The architecture involves five entities:

- **Device:** It collects data from the environment and publishes them to the aggregator.
- **Aggregator:** It deals with the data received from devices. For simplicity, we use Agg to represent the aggregator.

DOI reference number: 10.18293/SEKE2021-003

- **Local Service Gateway (LSG):** It serves to connect the local IoT system to the global one and handle the local name assignment.
- **ICN-IoT Server:** It manages the subscriptions within the IoT system, provides subscribers services and enforces data access policies.
- **User:** The user interacts with the ICN-IoT server to get the data for subscribed services.

The core functions supported by the architecture are: (i) device discovery aiming to connect a new device with the system and establish relationships between nodes; (ii) service discovery meaning to subscribe to IoT services; (iii) naming service denoting assigning persistent names to devices; (iv) content delivery representing forwarding data to subscribers. The related notations and descriptions are listed in TABLE I.

TABLE I: Notations and descriptions

Notation	Description
puk_x/prk_x	Public/Private key of the device/user/intruder, $x \in \{d, u, i\}$
ak_x	Action key of the device/user/intruder, $x \in \{d, u, i\}$
cer_x	Certificate of the device's name/public key, $x \in \{n, k\}$

Before publishing data, the device must finish device discovery and naming service phases. Fig. 2 shows the actions.

- $a1$: A device sends an encrypted discovery request along with cer_k to the aggregator.
- $a2$: When receiving the request, the aggregator decrypts it using puk_d acquired from cer_k, and then verifies the device's identity. If the device is legal, the aggregator sends ak_d encrypted with prk_d to the device.
- $a3$: The device obtains ak_d through decryption, and then requests a name from the aggregator.
- $a4$: The aggregator sends a name request to the LSG via a secure channel which can prevent intruders from obtaining the request.
- $a5$: The LSG sends cer_n to the aggregator.
- $a6$: The aggregator encrypts cer_n using ak_d, and then provides it to the device.
- $a7$: The device gets cer_n through decryption, and then publishes the data encrypted with ak_d to the aggregator. In order to improve the security, the device also sends cer_n encrypted with prk_d to the aggregator.

Before getting data, the user needs to pass user registration and service discovery phases. The actions are given in Fig. 3.

- $b1$: A user initiates a registration request to the server.
- $b2$: When the request is received, the server sends a temporary password to the user via a secure channel.
- $b3$: Once receiving the password, the user changes it first, and then sends an action key request to the server.
- $b4$: The server assigns ak_u encrypted by prk_u to the user.
- $b5$: The user decrypts the message to get ak_u using puk_u, and then sends the server a service request encrypted with ak_u and prk_u.

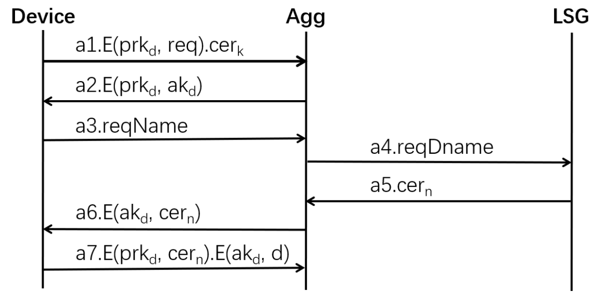


Fig. 2: Overview of publishing data

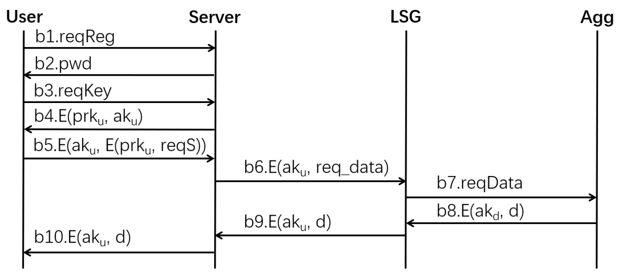


Fig. 3: Overview of getting data

- $b6$: The server decrypts the service request through two layers of decryption to verify the user's identity. If the user is honest, the server requests the corresponding data from the LSG. If not, the service request is rejected.
- $b7$: The LSG requests the data from the aggregator.
- $b8$: The aggregator provides the data to the LSG.
- $b9$: The LSG forwards the received data to the server.
- $b10$: The server sends a message to the user who can obtain the data by decrypting the message with ak_u.

B. CSP

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

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

- $Skip$ means that a process terminates successfully.
- $a \rightarrow P$ indicates that a process performs action a first, and then acts like process P.
- $c?x \rightarrow P$ represents that a process receives a message via channel c and assigns the received message to x, and then behaves like process P.
- $c!v \rightarrow P$ 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.
- $P\Box Q$ 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 formal modeling of the ICN-IoT middleware architecture.

A. Sets, Messages and Channels

Before we investigate the formal model, we introduce some preparatory notations including sets, messages and channels.

First, we describe the related sets in this model. **Entity** set denotes entities including devices, aggregators, LSGs, ICN-IoT servers and users. **Req** set involves the request messages of entities. **Key** set represents all the keys including public key set Puk, private key set Prk and symmetric key set Smk. **Data** set contains the data published by devices. **Con** set means other message contents involving certificate set Cer, feedback message set Ack and password set Pwd.

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

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

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:

$$\begin{aligned}
MSG_{req} = \{&msg_{req}.a.b.req, msg_{req}.a.b.E(k_1, req).cer,\\
&msg_{req}.a.b.E(k_2, E(k_3, req)) \mid a, b \in Entity,\\
&k_1, k_2, k_3 \in Key, req \in Req, cer \in Cer\}\\
MSG_{key} = \{&msg_{key}.a.b.E(k_1, k_2) \mid a, b \in Entity,\\
&k_1 \in Prk, k_2 \in Smk\}\\
MSG_{con} = \{&msg_{con}.a.b.p, msg_{con}.a.b.E(k, c) \mid\\
&a, b \in Entity, p, c \in Con, k \in Key\}\\
MSG_{data} = \{&msg_{data}.a.b.E(k_1, d),\\
&msg_{data}.a.b.E(k_2, c).E(k_3, d) \mid a, b \in Entity,\\
&d \in Data, k_1, k_2, k_3 \in Key, c \in Cer\}\\
MSG = MSG&_{req} \cup MSG_{key} \cup MSG_{con} \cup MSG_{data}
\end{aligned}$$

MSG_{req} represents the set of request messages. MSG_{key} means the set of messages containing action keys encrypted by private keys. MSG_{con} set involves messages containing name certificates, feedback messages and passwords. MSG_{data} is composed of messages containing the data published by devices. MSG consists of all the messages in the model.

Then we give the definitions of communication channels:

- Channels between devices, aggregators, LSGs, ICN-IoT servers and users described by *COM_PATH*:

  $$ComDA, ComAL, ComSL, ComUS$$

- Channels for intruders to intercept or fake the transmitted messages denoted by *INTRUDER_ PATH*:

  $$FakeAD, FakeDA, FakeSU, FakeUS$$

The declaration of channels is shown as follows:

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

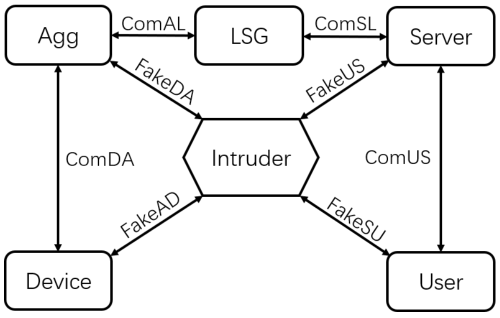


Fig. 4: Communication in the Model

B. Overall Modeling

In this section, we give the whole model of the ICN-IoT middleware architecture. $System_0$ represents the system consisting only of legal entities. In order to simulate the real environment, we consider behavior of intruders. $System$ denotes the system which introduces the attacks from intruders.

$$\begin{aligned}
&System_0 =_{df} Device \| Agg \| LSG \| Server \| User\\
&System =_{df} System_0[|INTRUDER\_PATH|]Intruder
\end{aligned}$$

$Device$, Agg, LSG, $Server$ and $User$ are processes describing the behavior of devices, aggregators, LSGs, ICN-IoT servers and users respectively. Besides, $Intruder$ process represents the actions of intruders such as intercepting and faking the messages transmitted among legal entities. The channels between processes are shown in Fig. 4.

C. Device Modeling

We formalize the process $Device_0$ to describe the behavior of the device without intruders as below:

$$\begin{aligned}
&Device_0\\
=_{df}\ &ComDA!msg_{req}.D.A.E(prk_d, req).cer_k \rightarrow\\
&ComDA?msg_{key}.A.D.E(prk_d, ak_d) \rightarrow\\
&\left(\begin{array}{l}
ComDA!msg_{req}.D.A.reqName \rightarrow\\
ComDA?msg_{con}.A.D.E(ak_d, cer_n) \rightarrow\\
\left(\begin{array}{l}
\left(\begin{array}{l}
ComDA!msg_{data}.D.A.E(prk_d, cer_n).\\
E(ak_d, d) \rightarrow ComDA?msg_{con}.A.D.suc\\
\rightarrow Device_0
\end{array}\right)\\
\lhd(D(ak_d, E(ak_d, cer_n))) \rhd (fail \rightarrow Device_0)
\end{array}\right)\\
\lhd(D(puk_d, E(prk_d, ak_d))) \rhd (fail \rightarrow Device_0)
\end{array}\right)
\end{aligned}$$

$reqName$ denotes the device's name request. d means the collected data. First, the device initiates a discovery request req signed with prk_d to the aggregator and then receives an action key ak_d. Then, the device sends the request $reqName$ to the aggregator, and obtains the name certificate cer_n. After acquiring ak_d and cer_n, the device publishes the data d encrypted with ak_d to the aggregator. In order to improve the confidentiality and integrity of the system, the message for publishing data also contains cer_n encrypted with prk_d. The above actions correspond to $a1 - a7$ in Fig. 2.

Now we consider the attacks from intruders. The process $Device$ with intruders is formalized via renaming as follows:

$$\begin{aligned} Device =_{df} & Device_0[[\\ & ComDA!\{|ComDA|\} \leftarrow ComDA!\{|ComDA|\}, \\ & ComDA!\{|ComDA|\} \leftarrow FakeDA!\{|ComDA|\}, \\ & ComDA?\{|ComDA|\} \leftarrow ComDA?\{|ComDA|\}, \\ & ComDA?\{|ComDA|\} \leftarrow FakeAD?\{|ComDA|\}]] \end{aligned}$$

$\{|ComDA|\}$ means the set of all communication over the channel $ComDA$. The first two lines mean that whenever $Device_0$ transmits a message on the channel $ComDA$, $Device$ can transmit the same message on channel $FakeDA$ or $ComDA$. The same is true for the last two lines.

D. Aggregator Modeling

The formal model of the aggregator abbreviated as Agg_0 without intruders is shown below:

$$\begin{aligned} & Agg_0 \\ =_{df} & \; ComDA?msg_{req}.D.A.E(prk_d, req).cer_k \rightarrow \\ & \left(\begin{array}{l} ComDA!msg_{key}.A.D.E(prk_d, ak_d) \rightarrow \\ ComDA?msg_{req}.D.A.reqName \rightarrow \\ ComAL!msg_{req}.A.L.reqDname \rightarrow \\ ComAL?msg_{con}.L.A.cer_n \rightarrow \\ ComDA!msg_{con}.A.D.E(ak_d, cer_n) \rightarrow \\ ComDA?msg_{data}.D.A.E(prk_d, cer_n).E(ak_d, d) \rightarrow \\ \left(\begin{array}{l} \left(\begin{array}{l} ComDA!msg_{con}.A.D.suc \rightarrow \\ ComAL?msg_{req}.L.A.reqData \rightarrow \\ ComAL!msg_{data}.A.L.E(ak_d, d) \rightarrow Agg_0 \end{array} \right) \\ \lhd (D(puk_d, E(prk_d, cer_n))) \wedge D(ak_d, E(ak_d, d)) \rhd \\ (fail \rightarrow Agg_0) \end{array} \right) \\ \lhd (D(puk_d, E(prk_d, req))) \rhd (fail \rightarrow Agg_0) \end{array} \right) \end{aligned}$$

When receiving the device's discovery request, the aggregator verifies its identity using the signature signed by prk_d. If the device is legal, the aggregator allows the device to join the system and assigns ak_d to it. Once the collected data are received from the device, the aggregator first verifies the authenticity of the device using the name certificate cer_n signed by prk_d. If the device is honest, the aggregator sends a positive feedback message suc to the device. The actions on the channel $ComDA$ mean $a1 - a3$ and $a6 - a7$ in Fig. 2. When receiving the LSG's data request, the aggregator sends the requested data to the LSG via the channel $ComAL$. The actions on the channel $ComAL$ represent $a4 - a5$ in Fig. 2 and $b7 - b8$ in Fig. 3.

The model of Agg with intruders can be drawn via renaming similar to the process $Device$, we omit the details here.

E. User Modeling

The model of $User_0$ without intruders is given as below:

$$\begin{aligned} & User_0 \\ =_{df} & \; ComUS!msg_{req}.U.S.reqReg \rightarrow \\ & \; ComUS?msg_{con}.S.U.pwd \rightarrow modifyPwd \rightarrow \\ & \; ComUS!msg_{req}.U.S.reqKey \rightarrow \\ & \; ComUS?msg_{key}.S.U.E(prk_u, ak_u) \rightarrow \\ & \left(\begin{array}{l} ComUS!msg_{req}.U.S.E(ak_u, E(prk_u, reqS)) \rightarrow \\ \left(\begin{array}{l} ComUS?msg_{data}.S.U.E(ak_u, d) \rightarrow User_0 \\ \lhd (D(ak_u, E(ak_u, d))) \rhd (fail \rightarrow User_0) \end{array} \right) \\ \lhd (D(puk_u, E(prk_u, ak_u))) \rhd (fail \rightarrow User_0) \end{array} \right) \end{aligned}$$

$modifyPwd$ is a function to change the password. First, the user sends a registration request $reqReg$ to the server and receives a temporary password. After the user changes the password using $modifyPwd$, it sends an action key request $reqKey$ to the server. When receiving ak_u, the user sends the service request $reqS$ encrypted by ak_u and signed with prk_u to subscribe to interested services. Finally, the user obtains the data for subscribed services through decryption. These actions denote $b1 - b5$ and $b10$ in Fig. 3.

The model of $User$ with intruders can also be acquired by renaming, we leave out the details here.

F. Server Modeling

We give the model of process $Server_0$ to describe the behavior of the ICN-IoT server without intruders as follows:

$$\begin{aligned} & Server_0 \\ =_{df} & \; ComUS?msg_{req}.U.S.reqReg \rightarrow \\ & \; ComUS!msg_{con}.S.U.pwd \rightarrow \\ & \; ComUS?msg_{req}.U.S.reqKey \rightarrow \\ & \; ComUS!msg_{key}.S.U.E(prk_u, ak_u) \rightarrow \\ & \; ComUS?msg_{req}.U.S.E(ak_u, E(prk_u, reqS)) \rightarrow \\ & \left(\begin{array}{l} ComSL!msg_{req}.S.L.E(ak_u, req\_data) \rightarrow \\ ComSL?msg_{data}.L.S.E(ak_u, d) \rightarrow \\ \left(\begin{array}{l} ComUS!msg_{data}.S.U.E(ak_u, d) \rightarrow Server_0 \\ \lhd (D(ak_u, E(ak_u, d))) \rhd (fail \rightarrow Server_0) \end{array} \right) \\ \lhd (D(ak_u, puk_u, E(ak_u, E(prk_u, reqS)))) \rhd \\ (fail \rightarrow Server_0) \end{array} \right) \end{aligned}$$

pwd is the temporary password sent to the user. req_data is the data request sent to the LSG. When the server receives the encrypted service request $reqS$, it checks if the user is legal using the signature signed by prk_u. If the user is legal, the server allows its service request, and then requests the corresponding data from the LSG. Otherwise, the service request is rejected by the server. After receiving the message containing requested data, the server first decrypts the message, and then sends the data encrypted with ak_u to the user. The actions on channel $ComUS$ correspond to $b1 - b5$ and $b10$ in Fig. 3. The actions on channel $ComSL$ denote $b6$ and $b9$ in Fig. 3. The model of $Server$ considering intruders can be formalized via renaming as well, the details are omitted here. Similarly, we can define the model of process LSG.

G. Intruder Modeling

In order to better simulate the ICN-IoT middleware architecture in the real environment, we model the $Intruder$ process which can intercept and fake messages among honest entities.

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

$$\begin{aligned} Fact =_{df} & \; Entity \cup Puk \cup Cer \cup \{E(k, d) \mid k \in Key, \\ & \; d \in Data\} \cup MSG \cup \{puk_i, prk_i\} \end{aligned}$$

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.

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

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.

$$Info(msg_{req}.a.b.E(k_1, req).c) =_{df} \{a, b, E(k_1, req), c\}$$
$$Info(msg_{key}.a.b.E(k_1, k_2)) =_{df} \{a, b, E(k_1, k_2)\}$$
$$Info(msg_{con}.a.b.E(k_1, con)) =_{df} \{a, b, E(k_1, con)\}$$
$$Info(msg_{data}.a.b.E(k_1, d)) =_{df} \{a, b, E(k_1, d)\}$$

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:

$$\begin{aligned}
&Intruder_0(F)\\
&=_{df} \Box_{m \in MSG} Fake.m \rightarrow Intruder_0(F \cup Info(m))\\
&\quad \Box\Box_{f \in Fact, f \notin F, F \mapsto f} Init\{kl = false\} \rightarrow Deduce.f.F\\
&\quad \rightarrow \left(\begin{array}{l} (\ kl = true \rightarrow Intruder_0(F \cup \{f\})\ )\\ \lhd (f == ak_d) \vee (f == ak_u) \rhd\\ (\ kl = false \rightarrow Intruder_0(F \cup \{f\})\ ) \end{array} \right)
\end{aligned}$$

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 m 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. If the intruder successfully deduces action keys of the entities, action keys leakage occurs. Now we give the formal model of $Intruder$, including its initial knowledge IK.

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

IV. VERIFICATION AND IMPROVEMENT

In this section, we analyse the verification results of the architecture. Based on the verification results and the analysis of attacks, we improve the original model and give the new verification results of the improved model.

A. Properties Verification

$System()$ means the model with intruders. We use Linear Temporal Logic (LTL) formulas to verify its properties.

Property 1: Deadlock Freedom

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

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

Property 2: Data Availability

$$\#define\ Data\_Availability\ data\_success == true;$$
$$\#assert\ System()\ reaches\ Data\_Availability;$$

The property means that a legal user should get the required data. The assertion is used to check the property.

Property 3: Action Keys Leakage

$$\#define\ ActionKeys\_Leak\_Success\ kl == true;$$
$$\#assert\ System()\ \ |= []!\ ActionKeys\_Leak\_Success;$$

As a vital part of the architecture, the leakage of action keys will cause a bad effect. We define a Boolean variable kl to check if the intruder can get the action keys, using the "always" operator [] in LTL.

Property 4: Device Faking

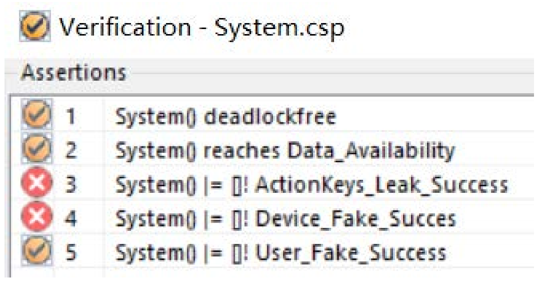
$$\#define\ Device\_Fake\_Success\ device\_fake == true;$$
$$\#assert\ System()\ |= []!\ Device\_Fake\_Success;$$

The property means that the intruder can pretend to be a legal device without being recognized. We adopt a Boolean variable $device\_fake$ for the verification in PAT.

Property 5: User Faking

$$\#define\ User\_Fake\_Success\ user\_fake == true;$$
$$\#assert\ System()\ |= []!\ User\_Fake\_Success;$$

The architecture should prevent intruders from subscribing to the services. If the intruder can fake as a legal user to obtain the data, many security issues may appear. We define a Boolean variable $user\_fake$ to verify the property.


Verification - System.csp

Assertions

✓	1	System() deadlockfree
✓	2	System() reaches Data_Availability
✗	3	System() |= []! ActionKeys_Leak_Success
✗	4	System() |= []! Device_Fake_Succes
✓	5	System() |= []! User_Fake_Success

Fig. 5: Verification results of the model

B. Verification Results

The verification results are shown in Fig. 5:

- $Property$ 1 is valid. It indicates that the proposed architecture will never get stuck in a deadlock situation.
- $Property$ 2 is valid, which shows that the data can be transmitted to the legal user who subscribes to the service.
- $Property$ 3 is invalid. It illustrates that the architecture can cause action keys leakage.
- $Property$ 4 is invalid. It means that the intruder can pretend to be a legal device to publish fake data.
- $Property$ 5 is valid, which represents that the intruder cannot disguise as a legal user successfully.

C. Attack Analysis

In this section, we discuss the reasons for the above insecure results. When the aggregator assigns ak_d to a device, it uses prk_d to encrypt the message. Once getting puk_d, the intruder can use it to decrypt the message to acquire ak_d. Moreover, the intruder can tamper with the collected data and fake as the device to publish modified data. An example that leads to $Device\ Faking$ and the leakage of ak_d is given as follows:

$A1.\ D \longrightarrow I : D.A.E(prk_d, req).cer_k$
$A2.\ I \longrightarrow A : D.A.E(prk_d, req).cer_k$
$A3.\ A \longrightarrow I : A.D.E(prk_d, ak_d)$
$A4.\ I \longrightarrow D : A.D.E(prk_d, ak_d)$
$A5.\ D \longrightarrow I : D.A.reqName$
$A6.\ I \longrightarrow A : D.A.reqName$
$A7.\ A \longrightarrow I : A.D.E(ak_d, cer_n)$
$A8.\ I \longrightarrow D : A.D.E(ak_d, cer_n)$
$A9.\ D \longrightarrow I : D.A.E(prk_d, cer_n).E(ak_d, d)$
$A10.\ I \longrightarrow A : D.A.E(prk_d, cer_n).E(ak_d, fakeD)$

- $A1$: The device sends a request to the aggregator.
- $A2$: The intruder eavesdrops on the request and gets cer_k. Since the certificate is issued by CA and every entity has its public key, the intruder can decrypt cer_k to get puk_d using the public key of CA.
- $A3$: The aggregator uses prk_d to encrypt ak_d and sends the message to the device.
- $A4$: The intruder intercepts the message and decrypts it to get ak_d using the acquired puk_d. At this point, the leakage of ak_d occurs.
- $A5 - A8$: The device requests for a name and receives cer_n, during which the intruder intercepts the activities.
- $A9$: The device publishes collected data to the aggregator.
- $A10$: The intruder intercepts the message and gets the data using ak_d. Then it disguises as the legal device to send modified data $fakeD$ along with the original cer_n to the aggregator without being recognized.

Similarly, the intruder can get ak_u using puk_u. Hence, the model cannot ensure the security of action keys and data.

D. Improved Model and Verification

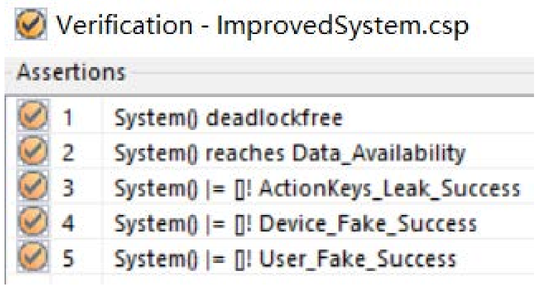
In order to address the above issues, we improve the model by using a method similar to the digital signature. When distributing the action key, we use the receiver's public key to encrypt the message. Furthermore, we introduce a method similar to digital signature for the aggregator to authenticate the device. That is to say, the device needs to sign with its private key when publishing the collected data. Therefore, the intruder can neither get the action keys nor fake as a legal device since it does not know prk_d or prk_u. We modify the message definitions of the model. MSG_{key} and MSG_{data} are replaced by the following MSG_{key1} and MSG_{data1}.

$$\begin{aligned} MSG_{key1} = \{&msg_{key1}.a.b.E(k_1, k_2) \mid a, b \in Entity, \\ &k_1 \in Puk, k_2 \in Smk\} \\ MSG_{data1} = \{&msg_{data1}.a.b.E(k, E(k1, d).c), \\ &msg_{data1}.a.b.E(k, d) \mid a, b \in Entity, k \in Smk, \\ &k1 \in Prk, d \in Data, c \in Cer\} \end{aligned}$$

Then we formalize the improved processes of $Device_1$, Agg_1, LSG_1, $Server_1$ and $User_1$ using the new message definitions. The improved model is given as follows:

$$\begin{aligned} &System_1 =_{df} Device_1 \| Agg_1 \| LSG_1 \| Server_1 \| User_1 \\ &System =_{df} System_1 [|INTRUDER\_PATH|] Intruder \end{aligned}$$

The verification results are shown in Fig. 6. $Property$ 3 and $Property$ 4 are valid. It means that $Action\ Keys\ Leakage$ and $Device\ Faking$ problems are solved now.


Verification - ImprovedSystem.csp

Assertions

✓	1	System() deadlockfree
✓	2	System() reaches Data_Availability
✓	3	System() |= []! ActionKeys_Leak_Success
✓	4	System() |= []! Device_Fake_Success
✓	5	System() |= []! User_Fake_Success

Fig. 6: Verification results of the improved model

V. CONCLUSION AND FUTURE WORK

ICN-IoT middleware architecture is constructed by applying ICN into IoT. In this paper, we formalized the architecture using CSP. Feeding the model into PAT, we verified the functional and security properties of the model including deadlock freedom, data availability, action keys leakage, device faking and user faking. The verification results show that action keys leakage and device faking may occur once intruders appear. Thus, we improved the model by encrypting messages with the receiver's public key. Moreover, we introduced a method similar to digital signature to the model. The new verification results indicate that the improved model can prevent intruders from invading the architecture. In the future, we will focus on more security issues of IoT systems. Formal methods will be used to verify other security properties of IoT systems.

Acknowledgements. This work was partly supported by National Key Research and Development Program of China (Grant No. 2018YFB2101300), National Natural Science Foundation of China (Grant No. 61872145, 62032024), Shanghai Collaborative Innovation Center of Trustworthy Software for Internet of Things (Grant No. ZF1213).

REFERENCES

[1] Tewari A, Gupta B B. Security, privacy and trust of different layers in Internet-of-Things (IoTs) framework. Future generation computer systems, 2020, 108: 909-920.
[2] Ayoade G, El-Ghamry A, Karande V, et al. Secure data processing for IoT middleware systems. The Journal of Supercomputing, 2019, 75(8): 4684-4709.
[3] Park S, Park S. A Cloud-based Middleware for Self-Adaptive IoT-Collaboration Services. Sensors, 2019, 19(20): 4559.
[4] Sicari S, Rizzardi A, Miorandi D, et al. A secure and quality-aware prototypical architecture for the Internet of Things. Information Systems, 2016, 58: 43-55.
[5] Shi Y, Zhang Y, Jacobsen H A, et al. Using machine learning to provide reliable differentiated services for IoT in SDN-like Publish/Subscribe middleware. Sensors, 2019, 19(6): 1449.
[6] Sicari S, Rizzardi A, Grieco L A, et al. A secure ICN-IoT architecture. 2017 IEEE international conference on communications workshops (ICC workshops). IEEE, 2017: 259-264.
[7] Mars D, Gammar S M, Lahmadi A, et al. Using information centric networking in internet of things: a survey. Wireless Personal Communications, 2019, 105(1): 87-103.
[8] Arshad S, Azam M A, Rehmani M H, et al. Recent advances in information-centric networking-based Internet of Things (ICN-IoT). IEEE Internet of Things Journal, 2018, 6(2): 2128-2158.
[9] Hoare C A R. Communicating sequential processes. Communications of the ACM, 1978, 21(8): 666-677.
[10] PAT, Pat: Process Analysis Toolkit(2019), http://pat.comp.nus.edu.sg.