Formal Modelling and Verification of MCAC Router Architecture in ICN

Junya Xu[1], Huibiao Zhu*[1], Lili Xiao[1], Jiaqi Yin[1], Yuan Fei*[2], Gang Lu[1]
[1]Shanghai Key Laboratory of Trustworthy Computing,
East China Normal University, Shanghai, China
[2] School of Information, Mechanical and Electrical Engineering,
Shanghai Normal University, Shanghai, China

***Abstract*—As Information Center Network (ICN) becomes a candidate for the future Internet architecture, its security and privacy issues have aroused extensive attention. Mandatory Content Access Control (MCAC) router architecture, which extends the existing mainstream ones with hardware-rooted trust, is proposed to implement MCAC protocol to protect privacy. Therefore, it is necessary to study the security of this architecture from the perspective of formal methods.**

In this paper, we use the process algebra Communicating Sequential Processes (CSP) to model and analyze MCAC router architecture in ICN. By adopting model checking tool Process Analysis Toolkit (PAT), we verify five important properties, namely *Deadlock Freedom, Key Faking, Level Mechanism, Data Availability* and *Data Leaking*. The results of verification show the correctness and security of MCAC router architecture, from which it can be concluded that this architecture is reliable.

***Index Terms*—MCAC Router Architecture, ICN, CSP, Modeling, Verification**

I. Introduction

Information Centric Network (ICN) has been proposed as a candidate for future Internet architecture [1]. It breaks the host-centric pattern, replacing the traditional approach with end-to-end connectivity and unique named data based on content distribution architecture. Due to the fact that named data is isolated from physical locations in the network, caching and replication of data in ICN can more easily support network storage and forwarding [2], [6]. Since ICN is attracting more and more attention, security and privacy have also become the important concern [7]. Some security and privacy issues have been identified with current ICN architectures. For instance, as one of the ICN architectures, Content-Centric Network (CCN) [2] has been found that it still leads to data leaks, since it allows that routers in the forwarding path can cache contents.

The traditional Mandatory Access Control (MAC) [1] provides confidentiality for the network by security level labels mechanism. MAC is usually valid in a system, but not common in large-scale distributed systems. Thus, it is difficult to be applied in ICN. In order to address the issue, Li et al. [4] have proposed Mandatory Content Access Control (MCAC) to provide security protection for each component in ICN. Similar to MAC, MCAC also provides security level mechanism for different components to realize the security and privacy for data in ICN. Subjects including processes in routers and objects such as contents and contest requests in MCAC, are respectively labeled into four levels $\{h,\ n,\ d,\ p\}$, where the relationship of label's level is defined: $h > n > d > p$. In other words, subjects in MACA can only read the objects with the labels equal or lower than themselves.

Since MCAC policies are implemented by content routers in ICN, Li et al. [4] proposed a design of MCAC router architecture. This router architecture is based on the existing router architecture and hardware-rooted trust, with the addition of an authentication protocol. In previous work, A. Datta et al. [11] have only verified the authentication protocol by using cord calculus, but this architecture has not yet been modeled and verified by using formal methods. In this paper, we propose a formal verification of the security-related properties of this architecture. First, we use CSP [8], [10] to model MCAC router architecture in ICN. Then, we choose PAT [5] to verify some properties of our system. The results of verification show that the security of MCAC router architecture is still guaranteed despite the presence of the intruder.

This paper is organized as follows. Section II gives a brief introduction to MCAC router architecture and CSP. In section III, we model the modules of one router in MCAC router architecture in CSP. In section IV, we use PAT to implement the model and verify five properties. Finally, we conclude this paper and make a discussion on the future work in section V.

II. Background

In this section, we give an overview of MCAC router architecture and a brief introduction to CSP.

A. MCAC Router Architecture

To implement MCAC policies, routers need to make forwarding decisions based on each content request and have the function of storage to implement content caching. It mainly consists of the following modules.

- *Trusted Storage Module* (TSM) is responsible for negotiating secret keys with TSM on neighbor routers and writing the secret keys into module TEM. In addition, TSM also handles caching contents.
- *Trusted Labeling Module* (TLM) checks whether each read operation and cache operation are legal by reading the labels in content packets.

*Corresponding Authors. E-mail address: hbzhu@sei.ecnu.edu.cn (H. Zhu), yuanfei@shnu.edu.cn (Y. Fei).

DOI reference number: 10.18293/SEKE2020-048

- *Trusted Enforcement Module* (TEM) provides private protection for some contents by encrypting the contents. In addition, it also reclassifies the content labels according to the reclassification rules.

The procedures of router communication based on MCAC router architecture can be mainly divided into two stages: key negotiation (Fig. 1) and transmission of content packets (Fig. 2).

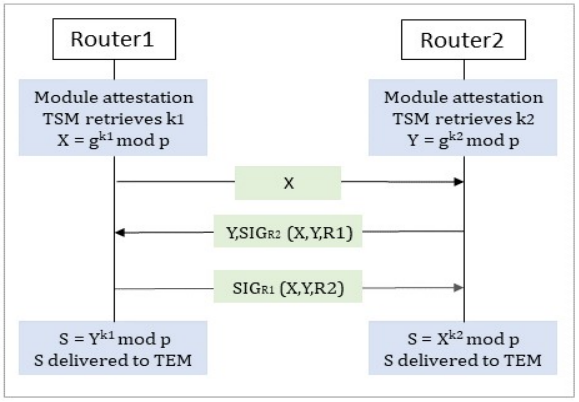

Fig. 1. Process of Key Negotiation

As shown in Fig. 1, identity authentication is added to MCAC router architecture during the process of key negotiation. In the beginning, all the modules should be verified for integrity. Then TSM retrieves the private key k_i. R1 as an initiator sends the Diffie-Hellman exponent composed of private key to R2. After receiving the message, the responder R2 sends its own Diffie-Hellman exponent and signature including mutual exponent and R1's ID to R1. Then R1 replies a message with signature to complete the authentication. R1 and R2 can get the same key S by calculating the Diffie-Hellman exponents. After generating their session keys, routers deliver these keys to the corresponding TEM modules.

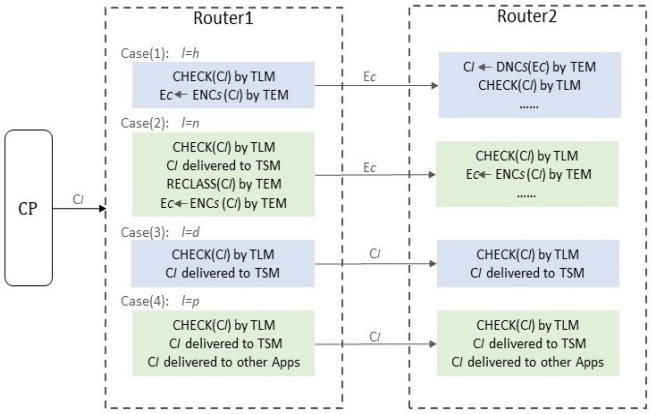

Fig. 2. Transmission of Content Packets

ContentProvider (CP) categorizes contents into four tag levels before sending them. Thus each router in MCAC router architecture needs to handle these contents in four situations as shown in Fig. 2.

- In case (1), CP generates the content with label h and sends the content to R1. After receiving the content, TLM in R1 needs to check the label embedded in the content at first. Because the label is h, the content is not sent to TSM but directly forwarded to R2 after TEM encrypts it using the key S. When R2 receives the content, it firstly decrypts the content and repeats the process like R1.
- In case (2), the content labeled with n is provided by CP for R1. TLM checks the label and sends the content to TSM since the label is lower than h. Then, TEM reclassifies the content by changing the content label from n to h and encrypts the content before delivering to R2. That is to say, R2 receives the content with label h and performs actions like R2 in case (1).
- In case (3), CP generates the content with label d and sends it to R1. After checked the label by TLM, the content with label d means that the content is sent to TSM and directly forwarded to R2 without encryption. R2 performs the same actions in the process of the communication.
- Case (4) is similar to case (3), the only difference is that the content with label p can also be delivered to all the applications.

B. CSP

CSP is a process algebra proposed by C. A. R. Hoare. The language is mainly designed to describe and analyze the behavior of concurrent systems and processes, which has been successfully applied in modeling and verifying various concurrent systems and protocols [3], [9].

We give the syntax of the CSP language used to describe the process in this paper, where P and Q are processes, a denotes the event and c represents the name of channel.

$$P, Q = Skip \mid Stop \mid a \rightarrow P \mid c?x \rightarrow P \mid c!x \rightarrow P \mid P \,\square\, Q \mid P \parallel Q \mid P \lhd b \rhd Q \mid P; Q \mid P[|X|]Q$$

- $Skip$ represents the process which does nothing but terminates successfully.
- $Stop$ denotes that the process does nothing and it is in the state of deadlock.
- $a \rightarrow P$ describes an object which first performs the event a and then behaves like P.
- $c?x \rightarrow P$ receives a message through channel c and stores the value in variable x and then the behavior is like process P.
- $c!x \rightarrow P$ sends message x through channel c and then behaves like process P.
- $P \,\square\, Q$ stands for the choice between process P and process Q. The election is decided by the environment.
- $P \parallel Q$ denotes that processes P and Q execute concurrently and are synchronized with the same communication events.
- $P \lhd b \rhd Q$ indicates if condition b is true, the process behaves like P, otherwise like Q.
- $P; Q$ describes that processes P and Q execute in sequence.
- $P[|X|]Q$ denotes that the parallel composition of P and Q performs the concurrent events on set X of channels.

III. Modeling

In this section, we formalize the model of MCAC router architecture in Information Centric Networks.

A. Sets, Messages and Channels

In order to model the communication of routers and the behaviors of the modules in a router, we give the definitions of sets, messages and channels we use in this paper.

First, we introduce some sets we use in the model. **Modules** set is composed of modules in the router including TSM, TEM and TLM. **Nonce** set represents Diffie-Hellman exponents. **Key** set consists of keys. **Content** set includes the contents transmitted between the modules or routers. **Label** set defines level of the content. **ID** set is the identity information of routers, and **Ack** set contains acknowledgements.

Then, we define an encryption function E and a decryption function D:

$$E(k, msg);\ D(k, emsg)$$

where the function E makes use of a key to encrypt the message msg while the function D uses a key to decrypt the encrypted message $emsg$. Therefore, we can get the following conclusion:

$$D(k, E(k, msg)) = msg$$

Based on the above sets and functions, we describe the following messages:

$$MSG_{datE} = \{msg_e.g, msg_e.c, msg_e.E(k,c), msg_e.E(k,g,r) \mid g \in Nonce,\ k \in Key, r \in ID, c \in Content\}$$

$$MSG_{datM} = \{msg_m.m.n.k,\ msg_m.m.n.c,\ msg_m.m.n.l.c \mid m, n \in Module, c \in Content, l \in Label\}$$

$$MSG_{ack} = \{msg_{ack}.x \mid x \in Ack\}$$

$$MSG_{pro} = \{msg_{pro}.E(k_1, g, r_1).k.r,\ msg_{pro}.E(k_1, c_1).k \mid g \in Nonce, k \in Key, r \in Name, c \in Content\}$$

$$MSG_{in} = MSG_{ack} \cup MSG_{pro}$$

$$MSG = MSG_{datE} \cup MSG_{datM} \cup MSG_{in}$$

Here, MSG_{datE} represents messages transmitted between the modules in adjacent routers. MSG_{datM} consists of messages sent between the modules in the same router. MSG_{pro} denotes messages delivered to a processing process and MSG_{ack} represents the set of feedback information from the processing process.

Next, we give the definitions of channels in this paper.

- channels of honest routers, using COMR_PATH to represent:
  $$ComTEM,\ ComTSM,$$
- channels of honest modules in the same router, using COMM_PATH to represent:
  $$KeySet,\ ContentProcess,\ ContentCache$$
- channels of intruders who perform intercepting or faking behaviors, defined by INTR_PATH:
  $$FakeTEM,\ FakeTSM$$
- channels of processing messages and feedback messages, represented by PROC_PATH:
  $$CheckKey,\ GetData$$
- In addition, channels of normal communications, represented by COM_PATH:
  $$COM\_PATH = COMR\_PATH \cup COMM\_PATH$$

The declarations of channels are as follows:

Channel $COM\_PATH, INTR\_PATH$: $MSG_{datE} \cup MSG_{datM}$

Channel $PROC\_PATH$: MSG_{in}

B. Overall Modeling

In this paper, we focus on the messages transmission of the modules inside a router and the external information transmission with the modules on neighbor routers. Meanwhile, we also consider the presence of intruders in normal communications. Note that in this paper, we allow intruders to eavesdrop on or intercept communication messages between routers, but these intruders are unable to obtain communication message between modules within the same router. Fig. 3 shows the communication between routers without intruders and Fig. 4 shows the communication with an intruder.

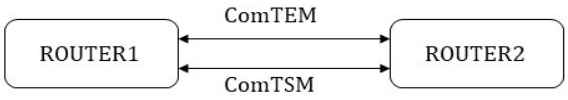

Fig. 3. Communication between Routers without Intruders

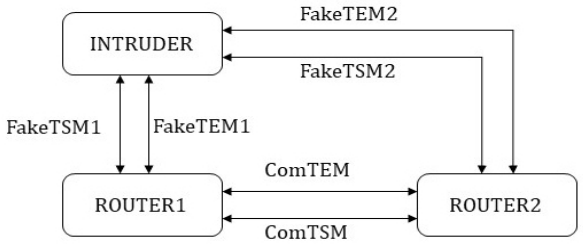

Fig. 4. Communication between Routers with an Intruder

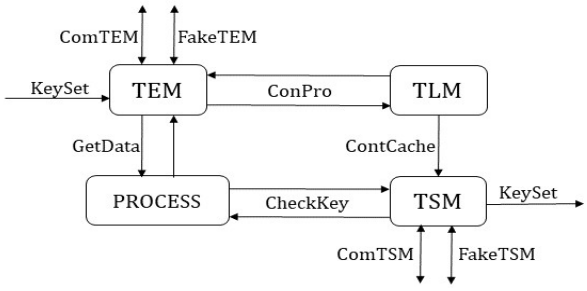

Fig. 5. Communication between Modules in a Router

Since the structure and communication function of each router are the same in MCAC router architecture, now we present the model of one router. Therefore as described in Fig. 5, any router can be abstracted as a system consisting of three modules including TSM, TEM, TLM and a process for internal information processing. That is to say, we do not consider other internal components of the router in this router architecture in this paper.

In order to take full account of the system security issues, we also define the process $INTRUDER$ to simulate intruders who eavesdrop and tamper messages.

Based on the above descriptions, we formalize the whole models $System$ and $System_I$ as follows.

$$System() =_{df} PROCESS() \ [|PROC\_PATH|] \ TEM_0() \ [|COM\_PATH|] \ TSM_0() \ [|COM\_PATH|] \ TLM_0()$$

$$System_I() =_{df} System() \ [|INTR\_PATH|] \ INTRUDER$$

Process $System$ is composed of processes TSM_0, TLM_0, TEM_0 and $PROCESS$, which perform the concurrent events on the sets $PROC\_PATH$ and COM_PATH of channels. Process $System_I$ consists of processes $System$ and $INTRUDER$, which perform the concurrent events on the set $INTR\_PATH$ of channels.

C. TSM Modeling

First, we formalize process TSM_0 to describe the behaviors of Trusted Storage Module in a router.

$$\begin{aligned}
&TSM_0() =_{df} Init\{k = false, m = true\} \rightarrow \\
&ComTSM!msg_e.g^{k_1} \rightarrow \\
&ComTSM?msg_e.g^{k_2}.E(k_{RB}^{-1}, (g^{k_1}, g^{k_2}, RA)) \rightarrow \\
&ComTSM!msg_e.E(k_{RA}^{-1}, (g^{k_1}, g^{k_2}, RB))\{k_{SA} := (g^{k_2})^{k_1}\} \\
&\rightarrow KeySet!msg_m.s.e.k_{SA} \rightarrow \\
&CheckKey!msg_{pro}.E(k_{RB}^{-1}, (g^{k_1}, g^{k_2}, RA)).k_{RB} \rightarrow \\
&CheckKey?msg_{ack}.ack \rightarrow \\
&\left(\begin{array}{l} (KeyFakingSuccess\{k = true\} \rightarrow Skip) \\ \lhd ack == YES \rhd \\ (KeyFakingError\{k = false\} \rightarrow Skip) \end{array} \right); \\
&ContCache?msg_m.l.s.c\{l := GetLabel()\} \rightarrow \\
&\left(\begin{array}{l} (LevelMechanismErr\{m = false\} \rightarrow Skip) \\ \lhd l == h \rhd \\ (LevelMechanismCor\{m = true\} \rightarrow Skip) \end{array} \right) \\
&; \ TSM_0()
\end{aligned}$$

The initial state of TSM_0 is defined as $m = true$ and $k = false$, where m indicates whether the level mechanism is safe and k expresses whether key is fake. The following actions correspond to two parts including key negotiation and content caching. By channel CheckKey, we check if the session key is faked and GetLabel() is used to get the level labels of contents.

We need to take the possibility of intruder actions into consideration, therefore, the messages on channel $ComTSM$ can be faked or intercepted. We apply the renaming to process TSM_0. TSM_0 performs an action only on the channel $ComTSM$, but TSM can perform an action either on channel $ComTSM$ or on channel $FakeTSM$.

$$\begin{aligned}
&TSM() =_{df} TSM_0()[[\\
&ComTSM?\{|ComTSM|\} \leftarrow ComTSM?\{|ComTSM|\}, \\
&ComTSM?\{|ComTSM|\} \leftarrow FakeTSM?\{|ComTSM|\}, \\
&ComTSM!\{|ComTSM|\} \leftarrow ComTSM!\{|ComTSM|\}, \\
&ComTSM!\{|ComTSM|\} \leftarrow FakeTSM!\{|ComTSM|\}]]
\end{aligned}$$

D. TLM Modeling

TLM is responsible for identifying the labels in different content packets. We formalize process TLM_0 to describe the behavior of Trusted Labeling Module in a router.

$$\begin{aligned}
&TLM_0() =_{df} ConPro?msg_m.e.l.c \ \{l := GetLabel()\}; \\
&\left(\begin{array}{l} ConPro!msg_m.l.e.h.c \\ \lhd l == h \rhd \\ \left(\begin{array}{l} ContCache!msg_m.l.s.c \rightarrow \\ \left(\begin{array}{l} ConPro!msg_m.l.s.n.c \\ \lhd l == n \rhd \\ ConPro!msg_m.l.s.c \end{array} \right) \end{array} \right) \end{array} \right) \\
&; TLM_0()
\end{aligned}$$

After receiving a content packet, TLM_0 makes a judgement about the label of the content packet and performs the corresponding actions. Here, GetLabel() is used to get the level labels of contents.

E. TEM Modeling

The process TEM_0 describes the behavior of Trusted Enforcement Module in a router which is formalized as follow.

$$\begin{aligned}
&TEM_0() =_{df} Init\{a = false\} \rightarrow KeySet?msg_m.s.e.k_{SA} \rightarrow \\
&\left(\begin{array}{l} (ComTEM?msg_e.E(k_{SB}, c) \\ \{c := D(k_{SA}, E(k_{SB}, c))\} \rightarrow ConPro!msg_m.e.l.c) \\ \Box(ComTEM?msg_e.c \rightarrow ConPro!msg_m.e.l.c) \end{array} \right); \\
&GetData!msg_{pro}.E(k_{SB}, c) \rightarrow GetData?msg_{ack}.ack1 \rightarrow \\
&\left(\begin{array}{l} (DataAcqcquisitionSuccess\{a = true\} \rightarrow Skip) \\ \lhd ackl == YES \rhd \\ (DataAcqcquisitionError\{a = false\} \rightarrow Skip) \end{array} \right); \\
&\left(\begin{array}{l} (ConPro?msg_m.l.e.h.c \rightarrow \\ \ ComTEM!msg_e.E(k_{SA}, c)) \\ \Box(ConPro?msg_m.l.e.n.c \ \{n := h\} \rightarrow \\ \quad ComTEM!msg_e.E(k_{SA}, c)) \\ \Box(ConPro?msg_m.l.e.c \rightarrow ComTEM!\ msg_e.c) \end{array} \right) \\
&; \ TEM_0()
\end{aligned}$$

We define the initial state of TEM_0 as $a = false$, where boolean variable a expresses data availability. At first, TEM_0 receives a message including the negotiated secret key k_S from TSM_0. Then if TEM_0 receives the encrypted message from its neighbor router, it makes use of the negotiated secret key to decrypt the message and sends the content in the message to TLM_0 . If the content is not encrypted, it forwards the content directly to TLM_0. TEM_0 can also check whether the modules can get what it expects by channel $GetData$. In addition, TEM_0 needs to complete message encryption and reclassification functions.

Similarly for TEM, we do this renaming to process TEM_0.

$$\begin{aligned}
&TEM() =_{df} TEM_0()[[\\
&ComTEM?\{|ComTEM|\} \leftarrow ComTEM?\{|ComTEM|\}, \\
&ComTEM?\{|ComTEM|\} \leftarrow FakeTEM?\{|ComTEM|\}, \\
&ComTEM!\{|ComTEM|\} \leftarrow ComTEM!\{|ComTEM|\}, \\
&ComTEM!\{|ComTEM|\} \leftarrow FakeTEM!\{|ComTEM|\}]]
\end{aligned}$$

F. PROCESS Modeling

Ultimately, we give $PROCESS$ to simulate the internal information processing procedure in a module.

$$
\begin{aligned}
&PROCESS() =_{df} \\
&\quad CheckKey?msg_{pro}.E((k_{R1}^{-1}), g^{k_1}, g^{k_2}, R1).k_{R2} \\
&\quad \left(\begin{array}{l} (CheckKey!msg_{ack}.YES \rightarrow PROCESS()) \\ \lhd k_{R1} == k_{R2} \ \&\& \ g^{k_2} == g^{k_2}\_f \rhd \\ (CheckKey!msg_{ack}.NO \rightarrow PROCESS()) \end{array} \right) \\
&\quad \Box GetData?msg_{pro}.E(k_{SB}, c) \\
&\quad \left(\begin{array}{l} (GetData!msg_{ack}.YES \rightarrow PROCESS()) \\ \lhd k_{SB} == k_{SA} \parallel k_{SB} == k_{S\_f} \rhd \\ (GetData!msg_{ack}.NO \rightarrow PROCESS()) \end{array} \right)
\end{aligned}
$$

$PROCESS$ is used to deal with whether the negotiated session key is faked and whether the content can be transmitted to the module requiring for it. Then it sends feedback messages to TSM_0 and TEM_0 separately.

G. INTRUDER Modeling

We also regard $INTRUDER$ as a process and it can intercept messages transmitted on $ComTSM$ and $ComTEM$ or fake messages on $FakeTSM$ and $FakeTEM$ at any time.

First, we define set FACT which contains all the facts that can be learned by the intruder.

$$
\begin{aligned}
FACT =_{df} & TSMs \cup TLMs \cup TEMs \cup MSG_{datE} \\
& \cup \{K, K^{-1}, K_S, K_{S\_f}\} \cup \{N, Name\} \\
& \cup \{E(key, content) \mid key \in \{K^{-1}, K_S, K_{S\_f}\}, \\
& \qquad content \in \{N, Name, Content\}\}
\end{aligned}
$$

Next, we define the rules to express how the intruder can deduce new facts from what it has known, shown as follows:

$$
\begin{aligned}
&\{K, E(K^{-1}, c)\} \mapsto c, \qquad \{K_S, E(K_S, c)\} \mapsto c, \\
&\{K_{S\_f}, E(K_{S\_f}, c)\} \mapsto c, \quad \{K, c\} \mapsto E(K, c), \\
&\{K_S, c\} \mapsto E(K_S, c), \qquad \{K_{S\_f}, c\} \mapsto E(K_{S\_f}, c), \\
&F \mapsto f \wedge F \subseteq F' \Rightarrow F' \mapsto f
\end{aligned}
$$

where, set F denotes the facts the intruder has known, and f is the fact deduced from set F. $F \mapsto f$ represents that fact f can be deduced from the set F.

The first three rules describe that the intruder can use the corresponding key to decrypt the encrypted messages and get some contents. In the same way, the next three rules represent encryption. The ?nal rule is a structural rule, explaining that the intruder can deduce fact f from a lager set F', if f can be deduced from set F.

In addition, we define the $Info$ function to represent the facts which a intruder can learn from the intercepted and eavesdropped messages:

$$
\begin{aligned}
&Info(msg_e.g) =_{df} \{g\} \quad Info(msg_e.c) =_{df} \{c\} \\
&Info(msg_e.E(k, g, r)) =_{df} \{E(k, g, r)\} \\
&Info(msg_e.E(k, c)) =_{df} \{E(k, c)\} \\
&Info(msg_m.m.n.k) =_{df} \{m, n, k\} \\
&Info(msg_m.m.n.c) =_{df} \{m, n, c\} \\
&Info(msg_m.m.n.l.c) =_{df} \{m, n, l, c\}
\end{aligned}
$$

where $g \in Nonce,\ m, n \in Module,\ k \in Key,\ r \in ID,\ l \in Label,\ c \in Content$.

Finally, we declare a channel $Deduce$ used for deducing new facts:

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

We allow that the intruder can overhear all the messages transmitted between routers and learn all the facts from the messages, but it cannot intercept the messages transmitted between the modules in the same router. Meanwhile, the intruder can fake a message if it has learned some facts and deduce a new fact from known ones. Beyond that it can also use a key that it knows in fake sessions.

Based on the above, now we give the formalization of $INTRUDER$ as below:

$$
\begin{aligned}
&INTRUDER(F) =_{df} \\
&\quad \Box_{m \in MSG_E} FakeTSM?m \rightarrow INTRUDER(F \cup Info(m)) \\
&\Box\Box_{m \in MSG_E \cap Info(m) \subset F} FakeTSM!m \rightarrow INTRUDER(F) \\
&\Box\Box_{m \in MSG_E} FakeTEM?m \rightarrow INTRUDER(F \cup Info(m)) \\
&\Box\Box_{m \in MSG_E \cap Info(m) \subset F} FakeTEM!m \rightarrow INTRUDER(F) \\
&\Box\Box_{f \in Fact, f \notin F, F \mapsto f} Init\{e = false\} \rightarrow Deduce.f.F \rightarrow \\
&\quad \left(\begin{array}{l} (DaLeakSuc\{e = true\} \rightarrow INTRUDER(F \cup \{f\})) \\ \lhd (f == c \ \&\& \ f == h) \ || \ (f == c \ \&\& \ f == n) \rhd \\ (DaLeakErr\{e = flase\} \rightarrow INTRUDER(F \cup \{f\})) \end{array} \right)
\end{aligned}
$$

When the intruder intercepts a message in MSG_E, it can deduce some information from this message, and it may also replace some contents and send a fake message to other honest entities.

IV. VERIFICATION

In this section, we use model checker PAT to implement the formal model which has been formalized in section III. At the same time, we carry out some security properties verification of our system.

A. Security Specification

We want to check whether the intruder can intercept or fake messages successfully in the whole system. Thus, we test if our system is against the following specifications:

$$
\begin{aligned}
SPEC_{TSM} &=_{df} CHAOS\ (\Sigma - \{|FakeTSM|\}) \\
SPEC_{TEM} &=_{df} CHAOS\ (\Sigma - \{|FakeTEM|\})
\end{aligned}
$$

CHAOS(A) [3] is the most uncertain and divergent process of alphabet A. It can perform any events from the alphabet A, where Σ is the set of all events. For instance, if the process TSM is allowed to perform any events except those occurring on the channel $FakeTSM$, we can say that it satisfies the specification $SPEC_{TSM} =_{df} CHAOS\ (\Sigma - \{|FakeTSM|\})$. If the system with the intruder refines these specifications, it is indeed secure.

B. Properties Verification

In this subsection, we verify five properties: Deadlock Freedom, Key Faking, Level Mechanism, Data Availability and Data Leaking. As formalised above, $System_I()$ is used to denote the model with an intruder and give the results of verification at the end.

Property 1: Deadlock Freedom

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

The *deadlock-free* property is a primitive in PAT which means a system can avoid the deadlock. This property verifies whether our system can run into the deadlock state.

Property 2: Key Faking

$$\#define\ Key\_Faking\_Success\ k == true;$$
$$\#assert\ System_I()\ reaches\ Key\_Faking\_Success;$$

We define *Key Faking* to denote the situation that the intruder can break the mutual authentication between routers and successfully tamper with the key.

Property 3: Level Mechanism

$$\#define\ Level\_Mechanism\_safe\ m == true;$$
$$\#assert\ System_I()\ reaches\ Level\_Mechanism\_safe;$$

The property *Level Mechanism* means that contents can only be processed and cached by higher-level processes in routers to provide security and privacy.

Property 4: Data Availability

$$\#define\ Data\_Acquisition\_Success\ a == true;$$
$$\#assert\ System_I()\ reaches\ Data\_Acquisition\_Success;$$

The property *Data Availability* represents that the contents can be obtained by the modules which require these contents.

Property 5: Data Leaking

$$\#define\ Data\_Leakaging\_Success\ e == true;$$
$$\#assert\ System_I()\ reaches\ Data\_Leakaging\_Success;$$

We say that a system satisfies this property, if an intruder can intercept and crack the encrypted messages transmitted between the honest entities. This property verifies whether the content is leaked in the process of message transmission.

Fig. 6. Verification Result of the Properties in $System_I$

As shown in Fig. 6, the properties *Deadlock Freedom*, *Level Mechanism* and *Data Availability* are all valid, which means that the system cannot run into the deadlock state and the modules of routers can only process lower-level contents and obtain what they want respectively. Meanwhile, we also find the properties *Key Faking* and *Data Leaking* are invalid. These two properties ensure the correctness of key authentication and the security of data transmission in our model with the intruder respectively. Therefore, we can get a conclusion that data transmission in MCAC router architecture is safe.

V. Conclusion and Future Work

This paper focuses on the security and correctness of MCAC router architecture through formal methods. Firstly, we have formalized three modules comprising TEM, TLM and TSM in MCAC router architecture with CSP. Then we verified five properties related to security through the model checker PAT including *Deadlock Freedom, Key Faking, Level Mechanism, Data Availability* and *Data Leaking*. Consequently, we can conclude that the correctness and security of MCAC router architecture are guaranteed from the results of verification.

In the future, we will follow with interest the other properties of MCAC router architecture. Meanwhile we will also explore the way to model and verify other access control solutions of ICN.

Acknowledgement

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), Shanghai Collaborative Innovation Center of Trustworthy Software for Internet of Things (grant no. ZF1213), the Fundamental Research Funds for the Central Universities of China and the Opening Project of Shanghai Trusted Industrial Control Platform (grant no. TICPSH202003007-ZC).

References

[1] Van Jacobson, Diana K. Smetters, James D. Thornton, Michael F. Plass, Nick Briggs, Rebecca Braynard: Networking named content. Commun. ACM 55(1): 117-124 (2012)

[2] Dinh Nguyen, Kohei Sugiyama, Atsushi Tagami: Cache the Queues: Caching and Forwarding in ICN from a Congestion Control Perspective. ITC 2016: 243-251

[3] Gavin Lowe, A. W. Roscoe: Using CSP to Detect Errors in the TMN Protocol. IEEE Trans. Software Eng. 23(10): 659-669 (1997)

[4] Qi Li, Ravi Sandhu, Xinwen Zhang, Mingwei Xu: Mandatory Content Access Control for Privacy Protection in Information Centric Networks. IEEE Trans. Dependable Sec. Comput. 14(5): 494-506 (2017)

[5] PAT: Process Analysis Toolkit. http://pat.comp.nus.edu.sg/

[6] Anand Seetharam: On Caching and Routing in Information-Centric Networks. IEEE Communications Magazine 56(3): 204-209 (2018)

[7] Towards seamless mobility in ICN : connectivity, security, and reliability. (Vers une mobilit transparente dans le rseau ICN : connectivit, scurit, et fiabilit). Pierre and Marie Curie University, France, 2018

[8] Stephen D. Brookes, C. A. R. Hoare, A. W. Roscoe: A Theory of Communicating Sequential Processes. J. ACM 31(3): 560-599 (1984)

[9] Yuan Fei, Huibiao Zhu: Modeling and Verifying NDN Access Control Using CSP. ICFEM 2018: 143-159

[10] Hoare, C.A.R.: Communicating Sequential Processes. Prentice-Hall, Upper Saddle River (1985)

[11] Anupam Datta, Ante Derek, John C. Mitchell, Dusko Pavlovic: A derivation system and compositional logic for security protocols. Journal of Computer Security 13(3): 423-482 (2005)