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

SEKE 2022, KSIR Virtual Conference Center, Pittsburgh, USA
9-19 July, 2022

Outline

- Introduction
- Overview of AUPS
- Modeling AUPS
- Verification and Improvement
- Conclusion and Future Work

Outline

- Introduction
- Overview of AUPS
- Modeling AUPS
- Verification and Improvement
- Conclusion and Future Work

Background

- **Authenticated Publish/Subscribe (AUPS)** is an IoT system based on the publish/subscribe scheme.
- It adopted the Attribute-based Access Control (ABAC) and encryption technology to ensure the security and privacy of the system.

Motivation

- AUPS was more efficient than the other existing secure solutions. Due to the superb characteristics of AUPS, it will attract a great amount of attention from industries.
- We formalize the architecture of AUPS Using CSP.
- We use PAT to verify some related properties, including **Deadlock Freedom**, **Data Availability**, **Data Leakage**, **Device Faking**, and **User Privacy Leakage**.

Outline

- Introduction
- Overview of AUPS
- Modeling AUPS
- Verification and Improvement
- Conclusion and Future Work

Overview of AUPS

AUPS involves the following modules:

- **Device** collects data from environment and then publishes them to the NOS.
- **Networked Smart Object (NOS)** deals with the data received from devices.
- **Enforcement Framework (EF)** performs access control and manages the data encryption/decryption keys using a key table.
- **Keys Topics Manager (KTM)** generates the data encryption/decryption keys according to the services.
- **Broker** manages the subscriptions within the system.
- **User** subscribes to IoT services and gets the data from servers.

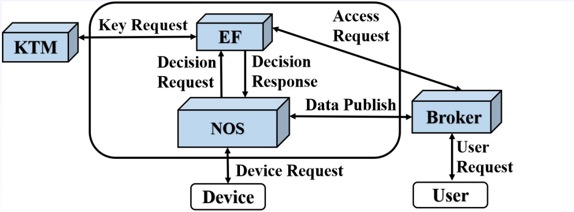


Figure 1: Architecture of AUPS

Overview of AUPS

- Overview of device publishing data:

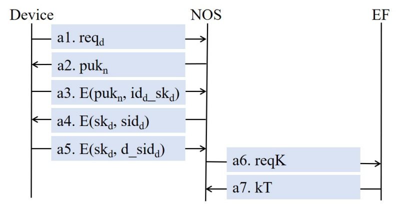
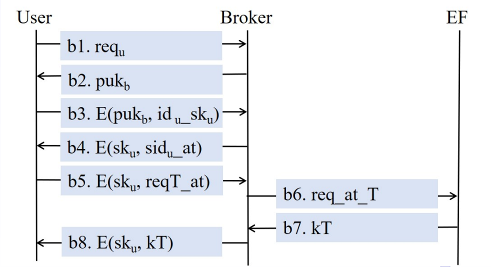


Figure 2: Overview of publishing data

- Overview of user getting data:



Outline

- Introduction
- Overview of AUPS
- Modeling AUPS
- Verification and Improvement
- Conclusion and Future Work

Modeling AUPS

Overview of the Model

$$System_0 =_{df} Broker \| User \| EF \| ProcessE \| KTM \| Device \| NOS \| Clock$$

$$System =_{df} System_0 [| INTRUDER\_PATH |] Intruder$$

- *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.
- *Intruder* process represents the actions of intruders such as intercepting and faking the transmitted messages.

Modeling AUPS

User Modeling:

Process $User_0$ describes the behavior of the user without intruders.

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

Modeling AUPS

User Modeling with Intruders:

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

- 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 same is true for the last two lines.

Modeling AUPS

Other Processes Modeling:

- The process *Broker* deals with the user's subscription request and forwards the data to the user.
- The process *Device* publishes the collected data to the NOS.
- The process *NOS* interacts with the devices to handle the publishment and data delivery.
- The process *EF* interacts with the broker and NOS to perform access control.
- The process *ProcessE* mainly deals with the key requests of entities.
- The process *Clock* serves to record the time and return the current time whenever the entities want it.

Modeling AUPS

Intruder Modeling:

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

- When intercepting a message m, the intruder adds Info(m) to its knowledge.
- Moreover, the intruder can deduce new facts from its knowledge via the channel DEDUCE and add them to its knowledge.
- If the intruder knows *info*(*m*), then it can pretend as a legal entity and fake the message *m*.

Outline

- Introduction
- Overview of AUPS
- Modeling AUPS
- Verification and Improvement
- Conclusion and Future Work

Verification and Improvement

Property 1: Deadlock Freedom

- AUPS 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 a legal user should get the required data. The following assertion is used to check the property.

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

Property 3: Data Leakage

- Data leakage can cause a bad effect to the system. We use a Boolean variable *dl* to verify the property, using the "always" operator [] in LTL. If the intruder obtains the data, we set the value of *dl* to *true*.

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

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*.

$$\#define\ \ Device\_Fake\_Success\ \ df == true;$$

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

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.

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

Verification and Improvement

Verification Results

- *Property* 1 and *Property* 2 are valid. It indicates that the model will never get stuck in a deadlock state and the data can be transmitted to the legal user who subscribes to the service.
- *Property* 3, *Property* 4 and *Property* 5 are invalid. It shows that the model can cause data leakage, device faking and user privacy leakage.

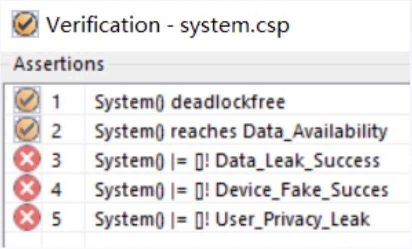


Figure 4: Verification results of the model

Verification and Improvement

Attack Analysis

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

- The intruder can obtain the data using kT.
- Similarly, the intruder can fake as the device to publish data, which leads to device faking.

We modify the message definitions of the model. MSG_{key} is replaced by the following MSG_{key2}.

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

The improved model is given as follows:

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

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

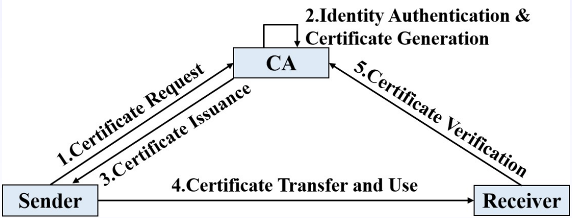


Figure 5: Flows of digital certificate

Verification and Improvement

Verification Result

Property 3, *Property* 4 and *Property* 5 are valid. It means that *Data Leakage*, *Device Faking* and *User Privacy Leakage* problems are solved now.

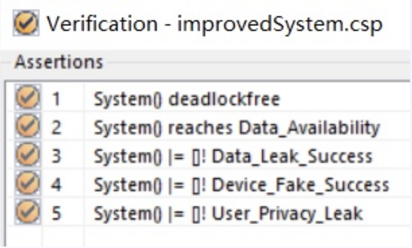


Figure 6: Verification results of the improved model

Outline

- Introduction
- Overview of AUPS
- Modeling AUPS
- Verification and Improvement
- Conclusion and Future Work

Conclusion and Future Work

Conclusion

- We formalized AUPS using CSP.
- We verified five properties of AUPS model, including **deadlock freedom**, **data availability**, **data leakage**, **device faking** and **user privacy leakage**.

Future Work

- Study more security properties of AUPS using formal methods.
- Improve our model to handle more attacks.

Thank you!