AoT: Authentication and Access Control for the Entire IoT Device Life-Cycle

Noura Alomar

November 7th, 2018

1 AoT

The AoT paper is one of the earliest and most cited papers on IoT defense and it considers the whole IoT device life-cycle [1]. The intuition behind proposing AoT is to make sure that smart IoT devices can get commands from authorized users only, and also ensure that only authorized users/devices can read information coming from IoT devices. In AoT, the device-gateway-cloud model is adopted (see Figure 1).

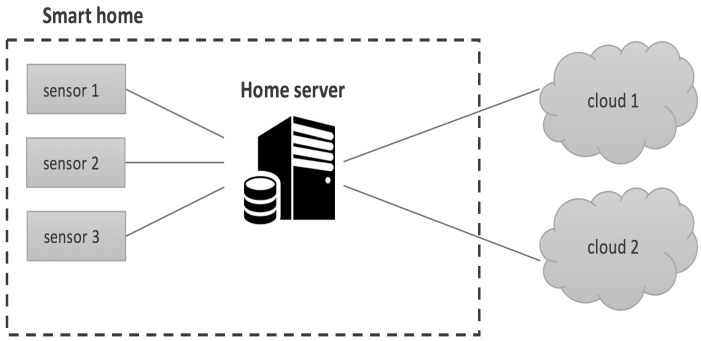

Figure 1: System model

1.1 AoT: System Model

The paper adopts the device-gateway-cloud model as shown in Figure 1. That is, let's say that we have a smart home that has many sensors (e.g., smart thermostats, smart locks and smart fridges), and that the sensors would talk to a home server, and then the home server can talk to the cloud. In this setting, every human user has a device (i.e., a smartphone) and the home server has a public key and a secret key. There is also a root user (home owner) with a root device that has a public key and a secret key as well. We can also assume that when the system is initially installed, the home server and the root device pair each other, so that they know each others' public keys. Some of the points that were discussed in the lecture about this model are:

1. ***Why do we need a home server in this setting?*** Sensors are resource-constrained and therefore cannot talk directly to the cloud, and that's why the home server is needed in this setting. For example, many sensors do not have the ability to maintain connections, do TLS, or even hold connection information and wait for answers. Thus, home servers can help with connecting to the cloud, and managing access control.

2. ***Why would not the smartphone be the home server?*** We need to install a home server because owners' smartphones might not be in the house while there might be other users who would want to use the IoT system.

3. ***In what sense are the smart devices constrained?*** The computational resources of smart devices could be low (e.g., sometimes they cannot do public-key cryptography). Further, they have storage constraints and limited throughput. When it comes to energy and power usage, some IoT devices might have short life times, and spending more energy might further shorten their life times. Furthermore, smart devices might not be always online, and their capabilities might allow them to send or receive data but not both.

4. ***How does the design of AoT make it scalable?*** AoT views each home in isolation; there are not many devices within a home and so each home has its own system and therefore there is no need to worry about scaling one system to the whole world.

1.2 AoT: Threat Model

The threat model of AoT addresses software attackers and not physical attackers. It also assumes that the manufacturing process is trusted, and that the trader is trusted as well.

1.3 AoT: IoT device life-cycle

The paper considers the whole life-cycle of smart devices, and this life-cycle consists of five phases, which are:

1. ***Pre-deployment*** This is when the devices are in the factory.

2. ***Ordering*** This is when the owner places an order to buy an IoT device.

3. ***Deployment*** This is when the owner receives the ordered device and installs it in the home.

4. ***Functioning*** This is when the device functions in the home.

5. ***Retirement*** This is when the IoT device is uninstalled.

1.4 AoT: security at each of the five phases of the life-cycle

Below, we describe the security at each of the five levels:

1. ***Pre-deployment:*** At this phase, the goal is to make sure that a smart device would communicate with the cloud securely after it has been installed in a home. Thus, while the device is in the factory, the public key of the cloud should be hardcoded in the device. This would ensure that the device can establish secure connection with the cloud without Man-In-The-Middle attacks. This helps in establishing secure communication with the cloud. This also helps the cloud send signed updates to the devices, so that the devices can check that these updates actually came from the cloud using the hardcoded public keys. Another alternative strategy is to use a shared symmetric key between the cloud and the devices; however, hardcodng the public key of the cloud is a more common strategy.

   In the pre-deployment phase, we also need to think about the other phases and be prepared for them. In particular, once home owners order IoT devices and want to install them, they need to prove to the devices that they are the owners. Thus, in order to prove to the device that a human user is the owner of that device, a PIN is installed on the device in the Ordering phase, and is also sent to the owner out-of-band (e.g., by email). When the owner receives the device, the owner can enter the PIN during the Deployment phase. This way the device can know that the user is the correct owner of the device

because he/she got the PIN. This is a form of *two-factor authentication*, as the user is getting the PIN and the device through different channels. This strategy prevents against an attacker who could take ownership of the device in transit, know the PIN and then use the device. At the end of this phase, the device is then sent to the trader who sells the device.

2. ***Ordering:*** At this phase, the owner orders an IoT device online and receives it from the trader, or buys it in-person. The trader also informs the cloud about the owner and the cloud then sends the PIN. Thus, if an attacker steals the device in transit, the attacker would not know the PIN. Here, we are trusting the trader, as the trader could: give a different device to the owner or tell the cloud that the trader bought the device and thus get the PIN from the cloud.

3. ***Deployment:*** This is the phase in which the owner is going to install the device. The owner is going to enter the PIN on the device, and then the system would need to do some pairing. That is, the home server and device pair to each other; the device gets the public key of the root user and the public key of the home server (see Figure 2). Then, the owner creates an account in the cloud. To prevent MITM attacks, *authenticated Diffie-Helman key exchange* could be utilized in this context. Namely, each device outputs a string of the key the two parties have agreed on, and then verify each other using QR codes (for example). Once the two parties agree on the symmetric shared key, they would have established a channel to exchange public keys securely. Thus, once we reach this stage, we can be sure that the root device, the home server and the new device know each others' public keys.

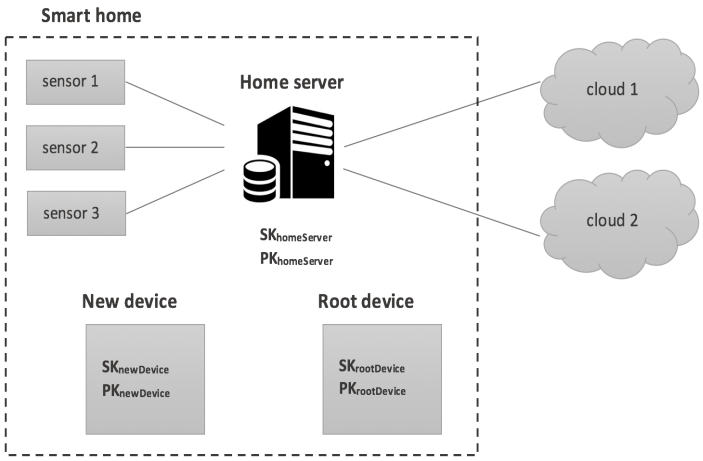

Figure 2: System model

4. ***Functioning:*** This phase involves using the device and giving access to other users. In this phase, the goal is to make sure that only legitimate users can send commands to a device, and can read data from a device. By *legitimate users*, we mean that the root user (or home server) gave them access to the device. For instance, let's say that the root user wants to give access to Alice to smart thermostat, and that the thermostat can take commands in the form of *set temperature* and *read temperature*. That is, let's assume that Alice now have access to *read temperature* and *set temperature*, and does not have access to *shut down thermostat*. This can be enforced cryptographically as follows:

   (a) Alice can prove to the thermostat that she has access to read and set the thermostat, and the thermostat can make sure that Alice cannot shut down the thermostat when she does not have permission to do so.

   $\text{Sign}_{\text{SK}_{rootDevice}}$ (PK_{Alice} has access to set_thermostat and read_thermostat; expiry time)

(b) When Alice decides to send a set_thermostat command to set the thermostat to 80° F, she can:

$\text{Sign}_{\text{SK}_{Alice}}$ (set 80° F; $\text{Sign}_{\text{SK}_{rootDevice}}$ (PK_{Alice} has access to set_thermostat and read_thermostat; expiry time); timestamp)

That is, the root user gives Alice a signature signed by the secret key of the root user on saying Alice's public key has access to set_thermostat and read_thermostat. There is also the expiry time, and Alice can access the commands only up to the specified expiry time. To prevent an attacker from replaying an old message from Alice, Alice appends a timestamp and signs the message with her secret key. This way, the attacker cannot append a secret key of his/her choice. To prevent Alice's attack of attempting to cheat on the timestamp, the device checks both the expiry time and the timestamp.

i. **Identity-Based Encryption (IBE)** The paper uses *identity-based encryption* in the AoT implementation [2]. The idea is that instead of using a public key to encrypt a message, AoT uses a device's name as the public key for the device. For this to work, there is a setup phase where the device needs to get a master secret key and a master public key from an authority that is assumed to be trusted. A master public key is needed for encryption, but only one key is needed for the whole system. In AoT, the home server is assumed to be the IBE authority, and thus it generates the secret key for a device when it is installed. The paper claims that there is no need to use devices' public keys to encrypt messages to them, using devices' names would suffice. One disadvantage of this approach is that key revocation is hard; changing a device's public key would require changing the device's name (i.e., when a device's secret key is compromised, the device would need to change its name). Below, we detail how to encrypt and decrypt using IBE:

   A. To encrypt: Enc(Master Public Key, name, Message) → ciphertext

   B. To decrypt: a user needs the authority to gives him/her a secret key for his/her name, and this secret key is used to decrypt the ciphertexts.

   $\text{Key}(\text{Master Secret Key}) \rightarrow \text{Secret Key}_{\text{name}}$

   $\text{Dec}(\text{Secret Key}_{\text{name}}, \text{ciphertext}) \rightarrow \text{Message}$

ii. **Attribute-Based Signatures (ABS)** ABS is another tool used in AoT [3, 4]. Similar to AoT, there should be a setup to generate a master secret key and a master public key, and this happens at the home server. Then, there is the key generation phase which takes the master secret key and a list of attributes, and then generates a secret key. In the AoT setting, the attributes are the functions that Alice has access to. For instance, Alice can obtain a secret key from the server with her attributes, which could be that Alice can have access to set_thermostat and read_thermostat, and can use the obtained secret key to sign any set_thermostat or read_thermostat commands. Then, Alice can use her secret key to sign a message for a certain predicate, and the Sign function produces a signature if the predicate is satisfied. This is demonstrated below:

   $\text{Key}(\text{Master Secret Key, Attributes}) \rightarrow \text{Secret Key}_{\text{Attributes}}$

   $\text{Sign}(\text{Secret Key}_{\text{Attributes}}, \text{Predicate}, \text{Message}) \rightarrow \text{Signature if Predicate(Attributes)=1}$

Verify(Master Secret Key, Predicate, Message, Signature)

However, it is not clear how this approach is better than just getting a signature from the home server or the root user saying that Alice has access to set_thermostat and read_thermostat. In the lecture, we discussed that this approach might have privacy benefits, as the device does not know that Alice sent the command and does not know the other commands that Alice can send; it only knows that the command was sent by someone who is allowed to do so. Further, this approach is more computationally expensive.

5. ***Retirement:*** This is the last step in the AoT system, and it is mainly concerned with wiping out the cryptographic material before discarding a device.

References

[1] A. L. M. Neto, A. L. Souza, I. Cunha, M. Nogueira, I. O. Nunes, L. Cotta, N. Gentille, A. A. Loureiro, D. F. Aranha, H. K. Patil *et al.*, "Aot: Authentication and access control for the entire iot device life-cycle," in *Proceedings of the 14th ACM Conference on Embedded Network Sensor Systems*. ACM, 2016, pp. 1–15.

[2] A. Shamir, "Identity-based cryptosystems and signature schemes," in *Workshop on the theory and application of cryptographic techniques*. Springer, 1984, pp. 47–53.

[3] V. Goyal, O. Pandey, A. Sahai, and B. Waters, "Attribute-based encryption for fine-grained access control of encrypted data," in *Proceedings of the 13th ACM conference on Computer and Communications Security*. Acm, 2006, pp. 89–98.

[4] J. Bethencourt, A. Sahai, and B. Waters, "Ciphertext-policy attribute-based encryption," in *IEEE Symposium on Security and Privacy*, 2007, pp. 321–334.