Wireless Personal Communications (2020) 114:295–327
https://doi.org/10.1007/s11277-020-07364-5

Design and Implementation of Hardware-Based Remote Attestation for a Secure Internet of Things

Jaehwan Ahn[1,2] · Il-Gu Lee[3] · Myungchul Kim[1]

Published online: 28 April 2020

Abstract
In general, Internet of Things (IoT) devices collect status information or operate according to control commands from other devices. If the safety and reliability of externally accessed devices are compromised, the risk of exposure of internally collected privacy information or abnormal operation of internal devices increases. This paper proposes a method of building a safe smart home environment by pre-blocking devices that may cause a risk by performing mutual safety verification between devices prior to data transmission and reception through the Session Initiation Protocol (SIP) of the home network. Using a Samsung's commercial smartphone, not a development board to implement the device's own verification function, and using an open source application and a SIP server providing free service, we established a test environment that is practically applicable and proved the feasibility of the attestation operation of the device. As a result of an operation test involving the capturing of packet data on a communication channel between two devices, it was confirmed that the transmission of parameter data for the actual attestation in SIP/Session Description Protocol packets succeeded without any problems. It was also confirmed that the final verification result of the target device was correctly derived. With the proposed method, it is possible to establish a safe trust relationship between smart home devices and external smart devices or between various IoT devices while also securing the smart home environment by blocking communications with devices that intentionally seek to do harm.

Keywords Knox · Attestation · SIP · Smart home security · Smartphone

✉ Myungchul Kim
mck@kaist.ac.kr

Jaehwan Ahn
ajaehwan@kaist.ac.kr

Il-Gu Lee
iglee@sungshin.ac.kr

[1] GSIS, School of Computing, KAIST, Daejeon, Korea

[2] The Affiliated Institute of ETRI, Daejeon, Korea

[3] Department of Convergence Security Engineering, Sungshin University, Seoul, Korea

1 Introduction

Based on data from Gartner [3], as shown in the Fig. 1, the number of connected things, referring to IoT devices, is expected to reach 20.4 billion worldwide in 2020, marking steep increases every year. These IoT devices are the main components of smart grids and smart homes. Also, smart homes present readily applicable environments for IoT devices.

Various IoT devices in a smart home, of growing interest to consumers, interoperate while connected to the home's network. There are devices that check the status and collect information about the home environment, such as temperature sensors, air quality sensors, gas sensors, and window open sensors, while other devices perform specific operations through Machine-to-Machine (M2M) communications, such as room temperature controllers, light controllers, gas locks, and door lock controllers. There are also devices that issue commands to perform a particular operation; that is, they perform control functions. These IoT devices are connected to each other via a home network and also to the external Internet through a home gateway.

The emergence of many IoT devices has been of interest to attackers and has resulted in several types of security threats. Examples of these are attacks that take control of smart LED lights, resulting in power outages [19, 40], attacks that collect private information using baby-monitor cameras [25], and attacks that result in severe threats through vulnerabilities in home automation solutions such as WeMo [38] and similar products.

A smart home network is an environment in which internal various IoT devices and communication devices form a single network to undertake information exchanges and where control and media data transmissions occur through PCs and smart devices from outside the home. For distributed device-to-device communication, each IoT device manufacturer can use its own proprietary protocol or can use the universally preferred Session Initiation Protocol (SIP) [27]. Bertran et al. [16] propose a method for communication between heterogeneous devices using SIP as middleware on a home automation platform. It uses messages in SIP, such as "MESSAGE" and "PUBLISH," for command transmission and status notification between devices. Das and Tuna [18] and Hrabovsky et al. [26] also

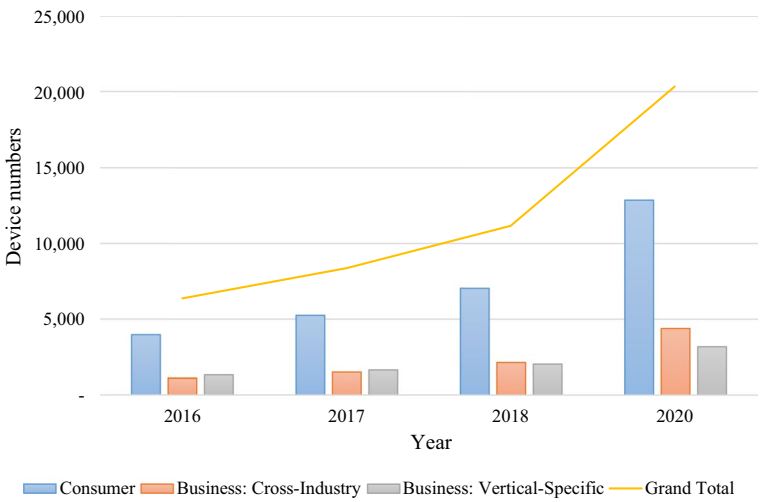

Fig. 1 IoT units installed by category (millions of units) [3]

discuss the usefulness of SIP for device-to-device communication in smart home environments given that SIP has the advantage of being flexible and easy to use compared to other protocols. In this way, SIP makes it possible to perform tasks such as sending commands from an external smart device to a home network device, obtaining measurement information, and sending media between different devices. In this paper, a method for applying the usefulness of SIP to remote attestation is proposed for communication between devices on a home network and external devices.

In this paper, we propose a hardware-based remote attestation scheme and corresponding protocol for a secure Internet of Things. The proposed scheme guarantees a reliable smart home environment using SIP and gives an effective means of mutual safety verification between devices. The implementation and operation experiments conducted here utilized specific scenarios of smart devices outside the home network and internal smart devices as examples to demonstrate the applicability of the proposed method. We propose a practical method using a commercial smartphone by Samsung equipped with the Knox platform rather than the board-type platform. To the best of our knowledge, Samsung's Knox platform is the only commercially available smart device platform capable of device attestation using TrustZone and the platform that provides development interfaces to use attestation capability. Therefore, we used the Knox platform for the attestation test. This paper makes the following contributions.

- We propose a method of device verification on a home network to prevent data exposure and/or abnormal operations within home network.
- By applying the proposed method to commercial smartphones and open-source software, we confirm that it is possible to verify remote devices and demonstrate the applicability of the method.
- The mutual attestation method ensures confirmed safety between devices.

This paper is organized as follows. Section 2 introduces related research areas, such as the work done on remote verification, safety verifications of smart devices, and smart homes via SIP, and Sect. 3 details the components of the method proposed in this paper. Section 4 describes the proposed method, and Sect. 5 describes a system configuration example and corresponding implementation process. Section 6 presents the results of the implemented system with an analysis. Finally, Sect. 7 concludes this paper.

2 Related Work

2.1 Remote Attestation Methods

Device attestation is a method used to verify the safe operation of a target device. Since the 1990's, there have been numerous studies of the device attestation [14]. Attestation results of devices have a variety of uses. Depending on the results, additional operations of the target device may be halted, recovery to the original state may be attempted through the network [13], or other corresponding operations may be performed. There is also local attestation to assess the safety of the device itself when it is booted as well as remote attestation to check the safety of a target device on other devices. In this paper, we focus on remote attestation to assess the level of safety between devices, as remote attestation is essential in a smart home environment.

Attestation can be divided into three categories according to the implementation method. These are software, hardware, and hybrid methods. Software-based attestation involves assessing whether a operation code of a device has been tampered with through a time-based checksum without the help of any hardware [29, 34, 35, 44, 45]. Assumptions related to the software-based attestation are so limited that actual multi-hop remote attestation over the Internet may be in violation of these assumptions, making this approach not suitable for practical use. Hardware attestation undertakes attestation by adding a secure hardware device. Examples of hardware used in the attestation are Trusted Computing Group (TCG)'s Trusted Platform Module (TPM) [10], Intel's Software Guard Extensions (SGX) [4], and ARM's TrustZone [1]. A TPM usage has been proposed in performing attestation in the Android platform [41]. Lastly, hybrid attestation refers to a means of mitigating the higher cost due to the use of additional hardware devices. This method allows attestation through certain modifications of the Micro Controller Unit (MCU) in an embedded device [20, 21, 30].

2.2 Security Enhancement Using TrustZone in the Smartphone

Many Android-based smart devices use ARM processors with TrustZone, and several methods have been proposed to enhance the safety of smart devices using TrustZone. VeriUI [37] proposes a secure password input method using TrustZone for authentication on a smartphone using the OAuth protocol [22, 24], which is a method that undertakes user authentication through the authentication of one representative server instead of password authentication for multiple web servers. AdAttester [32] provides a way to verify with TrustZone whether an actual user, not an automation script code, touches on a mobile advertisement screen and watches a product advertisement. Other studies have introduced technology such as SchrodinText [12] to protect text input by users through a UI and VButton [33] to verify user button input data. Another study has investigated the method of the integration of TrustZone with Android OS [46]

Samsung Electronics, a smartphone manufacturer, designed the Knox platform, which uses TrustZone, on its smartphones to ensure that the smartphones remain safe from a booting state to a running state. It also provides Application Programming Interfaces (APIs) of the Knox to check the current safety status. This feature also includes an attestation function that checks whether the smartphone has been updated using an unauthorized device image, referred to as a custom ROM or has been compromised through real-time checking of the kernel. Researchers have analyzed Knox's internal structure and have noted the possibility of vulnerabilities [15, 28, 39]; however, many improvements have been made since the introduction of Knox 2.0 and 3.0. As the development of TrustZone's OS and Trustlet, an application of TrustZone OS, is security-sensitive issue, the disclosure related with the internal structure and development methodology is thus far extremely limited. The installation of Trustlet on TrustZone is also limited because of the same reason. This explains why it is only handled by smartphone manufacturers, TrustZone OS developers, or carriers. For this reason, there is no other way for general developers to use TrustZone on commercial smart devices when developing security-related applications except using Samsung Knox as an indirect method.

3 System Overview

3.1 Structure of a Smart Home

As an example of a smart home configuration, as shown in Fig. 2, several heterogeneous devices form a network together, with the home gateway connected to the external Internet.

Among the devices on the home network, there are active devices such as a gas lock, door lock, and a light controller that perform specific operations according to commands from other devices. On the other hands, there are passive devices such as sensors that measure values in the environment and transfer these values to other devices. Interphones, IP cameras and Telephony systems are interactive media devices that can continuously transmit voice and video streams. In an environment using the SIP as proposed by Benjamin et al. [16], a SIP server can be included in the home network as a communication connection node between devices, and external smart devices can communicate with the devices on the home network.

The threat situations considered in this paper are as follows. All devices inside or outside the home network are targeted and at least one of them is compromised. There are two possible attack scenarios. The first case is when an external smart device is compromised. In this case, privacy information through IP cameras and various sensors inside the home network may leak in the external compromised smart device. Also the malfunctions of active devices such as gas, light, or door controllers can occur by the control of the compromised device. This leads to a serious situation. The second case is when the internal sensor device on a home network is compromised. In this case, data collected from the sensor may be forged and transmitted to the external device. As an example, the measured data of a smart meter in a smart grid system at home is used for billing. Compromising of a smart meter can be conducted for less billing. In order to cope with such attacks, remote attestation, which is a method of confirming whether a communication device is compromised, is required.

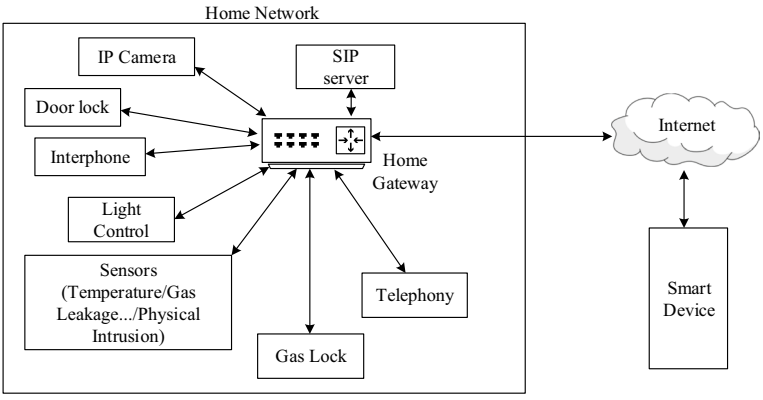

Fig. 2 Smart home configuration

3.2 Remote Attestation Mechanism

Device attestation is a method of verifying the trust of a device. During the attestation operation, there are two entities: a verifier and a prover. The prover is the object to be inspected for the checking of the safety of the device itself. The prover can be the target device of the verifier. The verifier is an object seeking to determine the security status of the prover.

Local attestation is the term used when the verifier and the prover are the same entity. There are several scenarios using verification result after local attestation. If a prover fails to pass the verification procedure, it is no longer operated or the verification result is stored and sent to the requester later, or it is restored to its original state through a recovery process.

Remote attestation is is the term when the verifier and the prover are separate entities remotely located from each other. In general, if it turns out that the prover is not safe after verification, the verifier rejects prover-related service operations.

Root of Trust (RoT) should be implemented and operated on the prover, which is the subject of verification as a basic element of device attestation. RoT is the basis for the trust of the device, meaning that the start of the device is processed from RoT. RoT's operation code is stored in a non-modifiable hardware element such as ROM, and the device starts boot process with this code when powered on. RoT checks the next execution region to verify that the integrity of the code has not been compromised before executing the code. The code executed in this region includes fundamental operation process again to verify the integrity of the subsequent execution region. If there are no problems with the next code region, the current program counter jumps to the next code region. This successive chain of execution steps allows the verification of the kernel area of the Operating System (OS) to determine that the device is running in its intended state. If an integrity mismatch is detected, no further action is taken or the current measurement results are stored in a secure area so that they can be transmitted when requested from an external device. In a PC environment, TPM [10] or SGX [4] is used to build such a trust system. In the mobile environment, the attestation can be done with TrustZone [1].

Checking a prover's safety through the trust chain is done in a verifier by remote device attestation. Figure 3 shows the basic remote attestation method. The verifier creates a

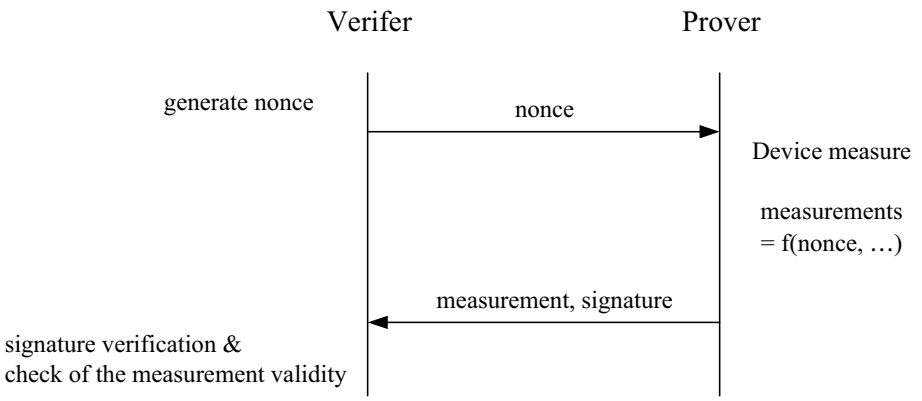

Fig. 3 Basic procedure of remote attestation

nonce with a non-overlapping value, which is used to prevent a replay, and sends it to the prover. The prover performs an attestation measurement taking the nonce as a parameter, signs the measurement result, and sends it back to the verifier so that the verifier can check that the measurement is correct. Through this sequence, the verifier can confirm the safety of the prover, which is the target device, and reject any prover-related request if it is found to be compromised.

In a mobile environment using TrustZone, the following remote attestation method can be considered. Operating environments of ARM-based mobile devices such as Android smartphones equipped with TrustZone are divided into *the secure world* and *the normal world*, referring to the terms used in the TrustZone architecture. In the normal world, the Android OS and general applications are running. For isolation and protection of the secure world from the normal world, TrustZone runs its own OS and trustlets, applications running in the TrustZone's OS, run in the secure world. As soon as the device boots up, TrustZone's OS takes control of the device, and after taking measurements of the normal world, it transfers control to the normal world. Measurements for the normal world are stored into a secure area within TrustZone. If there is an attestation request outside of the mobile device, TrustZone responds with a stored measurement or a re-measurement for that device. The public key method is used to verify the integrity of the measurement itself. TrustZone of the mobile device signs the measurement with the device's private key and exports it, and the external device verifies the signature of the measurement with the public key of the mobile device.

In summary, RoT should exist on the smart device for accurate operation of the remote attestation system so that the execution of start-up code can be performed from it, and there should be a Trusted Execution Environment (TEE) such as TrustZone that performs safe operations without external influences. Measurements and signatures must be available, and procedures for remotely verifying signatures and measurements should be available.

3.3 Knox Attestation

The Knox platform, mounted on Samsung's Android smartphones, provides APIs to perform attestation of the device to verify its safety. A detailed description of the downloading of the SDK, necessary to use the attestation API, and the procedure of the attestation operation are provided in the literature [8]. A summary of the attestation sequence is shown in Fig. 4.

1. A verifier that wants to check the safety of a smart device sends a request of a nonce for attestation to an attestation server. The nonce is used to prevent replay attacks, and the nonce is managed to prevent duplication on the attestation server.
2. The attestation server creates a nonce and sends it to the verifier.
3. The verifier sends a nonce to the prover and requests a binary large object (blob). A blob is a binary string that contains various measurements, identifications, a nonce, and signatures of measurements generated from the Knox platform of the device.
4. The API of the Knox platform is used to perform measurements on the device. The Knox platform, based on TrustZone, operates safely regardless of external factors of the device. The Knox platform takes measurements and outputs blob, which contains signature values that cannot be tampered with even if the device is compromised. The reliability of the device is measured using the TrustZone-based Integrity Management Architecture (TIMA), implemented on the Knox platform, which not only checks the

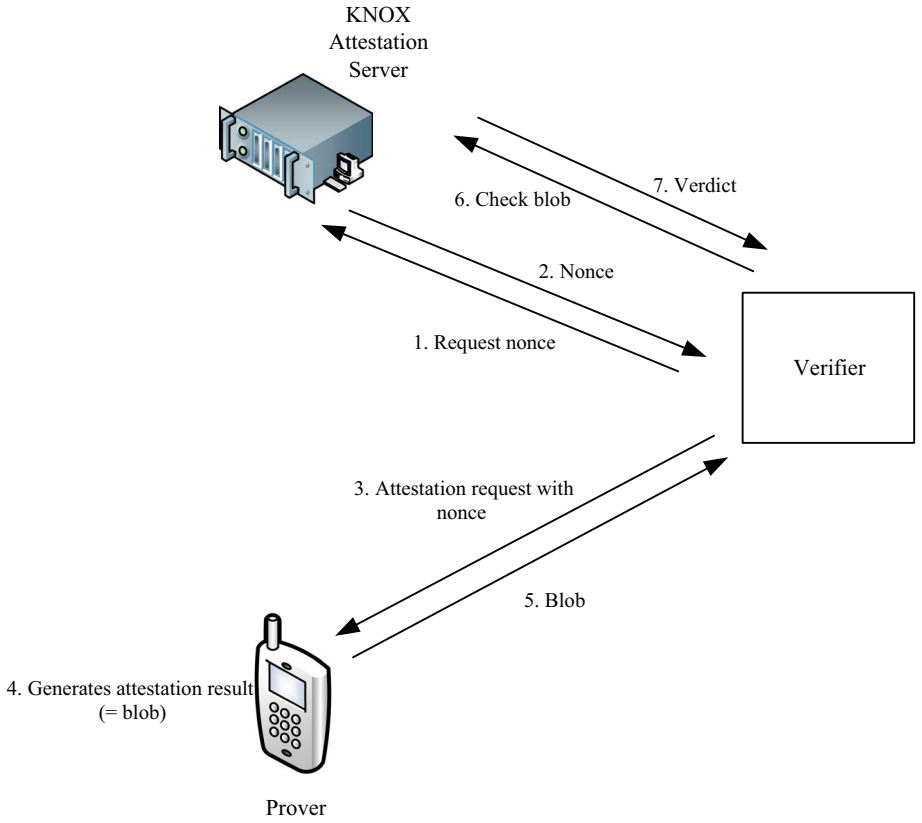

Fig. 4 Attestation process of the Knox platform

integrity of the static code, but also performs real-time and periodic monitoring of the operating kernel by means of Real-time Kernel Protection (RKP) and Periodic Kernel Monitoring (PKM).

5. The blob is passed to the verifier, which sends it to the Knox attestation server to request validation of the blob.
6. The Knox attestation server checks the signature and verifies that the blob has been sent from the target device and that the blob has not been tampered with, and then deduces the attestation result from the blob. The blob contains the nonce used during the attestation process. The attestation server checks the expiration time of the nonce as recorded in the server database upon the generation of the nonce. If the nonce is expired, the attestation server returns an error code.
7. Knox attestation server terminates the process by sending the final result to the verifier in the form of a verdict.

The data in the transmission channel between the prover, the verifier, and the Knox attestation server must be protected by a channel encryption method such as Secure Socket Layer (SSL) so that data integrity is not compromised to ensure the reliability of the final result.

3.4 Session Initiation Protocol (SIP)

SIP is a protocol for managing sessions between devices. Figure 5 shows the SIP procedures performed between device A, device B, and SIP server. One session can be comprised of session connection, Real-time Transport Protocol (RTP) media delivery, and session termination.

In order to establish a connection between two devices with the SIP, the three SIP messages of INVITE, 200 OK, and ACK are generally used in a three-way-handshake method. Device A wants to connect to device B using an INVITE message, and when device B acknowledges (ACK), device A and device B can perform mutual media transmissions. The extensibility of the SIP allows each message (e.g., INVITE, ACK, 200 OK, Re-INVITE) of the SIP to include an additional data parameter called a Session Description Protocol (SDP) [23] parameter. The SDP is described as *"parameter name = parameter data"* in a SIP message, and it can be considered as a method for transmitting attestation data.

4 Proposed Scheme

4.1 Attack Scenarios

We can consider the following security threats in a smart home. Smart devices such as smartphones and tablets can perform various status checks of home and control the

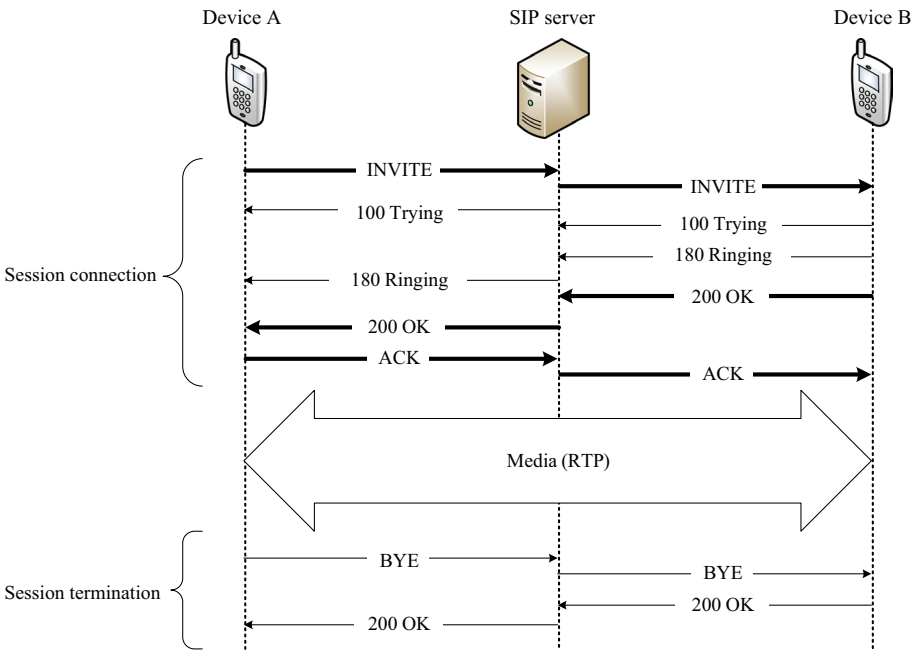

Fig. 5 SIP procedures for session management

home devices via the Internet from outside of the home network. Home network attacks can be categorized as passive attacks and active attacks. There may be a passive attack such as exposing the homeowner's life patterns or privacy information to the outside through the collection of internal state information, or an active attack such as controlling specific IoT devices to perform a specific operation. In order to respond safely to these operations from an external device, an access control function, that is, a device authentication or user authentication function, must initially be performed. Many IoT devices can be targeted due to a lack of authentication [42].

There are two types of threats to consider when performing access control. The first is when an attacker attempts direct access through his device, and the second is when an attacker attempts indirect access through a normal user's device by invading and taking control of it. The normal user does not know whether the device is compromised. In the first case, to prevent unauthorized access to the home network from the outside, a variety of authentication methods can be used. Such methods range from simple methods such as password authentication to more in-depth methods such as Fast IDentity Online (FIDO) [36, 47]. FIDO is a method which integrates device authentication and online authentication [17]. However, in the second threat case, in addition to authentication for devices and users for home network access control, it is necessary to check the safety of the smart device itself. If access to the home network is granted only through authentication, compromised smart devices can expose all information or take over control of all communications with the home network to attacker. There must be confirmation that the smart device is not compromised and will operate safely.

External smart devices may be exposed to various security threats, and the users of smart devices controlled by an attacker may not be aware of the situation. In such a state, an attacker may cause a harmful situation by exposing sensitive data between the smart device and the home network to the outside or by controlling an internal IoT device after the normal user's home network authentication process. With regard to information transmitted through data-collection devices such as IP cameras and voice devices on the home network, if taken over by an attacker, there is a risk of privacy exposure. In the smart grid environment, with data measured by smart meters attackers can identify the signature of separate devices and analyze the behavior patterns of the owners, possibly learning when the house is empty, increasing the potential for burglary [31]. For a temperature control device, a fire can be caused by an overload through an attack; for a door lock, the door may be opened by an externally compromised smart device, which may also be used for burglary.

In a smart home, there are some attack scenarios utilizing the aforementioned security threats. It is assumed that the external smartphone used as a controlling or monitoring device has been compromised by one of a variety of potential attacks. When the smartphone is compromised, the smartphone is expressed as *rooted* in other term. Rooting is accomplished by burning a custom ROM image of the smartphone or by exploiting a vulnerability of the OS when the smartphone is running. There are many rooted custom ROM images available on the web sites such as firmware.mobi [2] and TWRP [9]. Moreover, numerous OS vulnerabilities have been released and patched thus far. An authentication mechanism between a smart home and a smartphone is useless on a rooted smartphone because the user of a smartphone who does not recognize that the smartphone is compromised can pass the authentication procedure of the smart home. After the authentication, the attack code in the rooted smartphone can access an application connected to the smart home and leak data to the external attacker, or can control

the application to execute malicious command on a smart home device. One case of malicious ordering is the opening of a door connected to the smart home.

Another specific attack scenario is for VoIP communication between a smart home call device and an external smartphone. Users of devices want voice data to be securely transmitted and not to be leaked to a third party. It is assumed that the network traffic encryption methods are used for the protection of data transmitted between two users. While the data are protected in the communication channel, the data in the smartphone are not. A compromised external smartphone can leak the voice stream of a VoIP application without being known to the application or the user. The attack procedure is as follows. On Android, sound processing is handled by the Advanced Linux Sound Architecture (ALSA) driver library which is called tinyAlsa. Code modification of the tinyAlsa library can leak all sound stream data processed on an Android smartphone. Specifically, when an Android smartphone is rooted and the tinyAlsa library is replaced with a malicious library, all voice data can be exposed to the attacker.

On an Android smartphone, without modification of the tinyAlsa driver, the acquisition of voice data of other application is very hard because the sound library cannot be accessed from multiple applications simultaneously. If two applications access the sound library at the same time, a conflict occurs. Android framework does not allow simultaneous access to the sound library. Because only one of the two applications can preempt the library and use a sound-related function, the other application cannot access the sound library. This means that a malicious application cannot reveal other application's voice data in a normal device which is not rooted. However, in a compromised smartphone, even when the users of voice calls are authenticated by each other and voice data are transmitted securely, the voice data are not protected from leakages. Moreover, these leakages are not noticeable to the users.

4.2 Security Remedies

The system model for a secure smart home can be explained as Fig. 6 and Fig. 7. Smart home device in the internal network is device (A) and external smart device is device (B). Devices (A) and (B) communicate using the protocol of SIP through SIP server. The security level check of

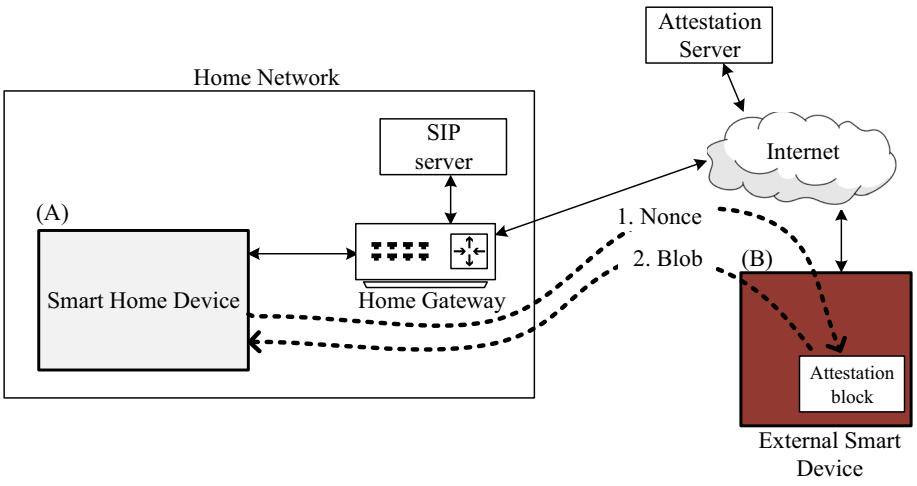

Fig. 6 System model when the security check of an external smart device should be performed

the other device in each device should be performed to protect various attacks. There are basic two system models to solve this problem. The first is when a smart home device should check an external smart device. One of examples of these cases is when an external smart device tries to access the smart home device and control it. Before communicating between device (A) and (B), the security level check of device (B) should be performed in device (A). This means that device (B) should have attestation capability. Device (A) sends a nonce and gets a blob calculated based on the nonce in the attestation block of device (B). During the blob calculation process, the security check is performed in device (B). This blob is analyzed in an attestation server. Device (A) can get a result whether a blob is correct or not. The nonce and blob are transferred through SDP in SIP. These are not secure data and are used only for the verification of device (B).

The second is when an external smart device should check a smart home device. One of examples of these cases is when an external power measuring system of power company wants to get smart meter data in home. The data of home should be trusted by the power company. Before communicating between device (A) and (B), the security level check of device (A) should be performed in device (B). The scenario to solve the problem is same as above model. If the two devices try to check the security of the other device at the same time, above two models can be applied in duplicate.

In two system models, if the attestation process succeeds, then device (A) or (B) can trust the security of the counterpart device and accept the access of the other device. If the result of the attestation shows failure, device (A) or (B) cannot trust the counterpart device. The target device is considered to be compromised. Therefore, further communication between the devices are not allowed and sensitive or secure information don't leak out of the device.

The basic suggestions for building a secure smart home are as follows. First of all, the target device should have attestation capability. In the device, the attestation function, the booting process starting with the RoT, and the measurement function should exist to verify the safety of the device itself and to respond to external attestation requests. For one-way attestation between a device in a smart home and an external smart device, the external smart device should have an attestation function. Regarding mutual attestation, both the smart home device and the external smart device should have an attestation function. If a device has a attestation capability, the other device trying to contact to the device can check the security level of the device.

In addition, devices participating in attestation should perform functions such as media data transfers, command transmissions, and should perform specific operations for command through the SIP server of the smart home.

Finally, the design and implementation of the attestation operation in the smart home configuration are necessary to confirm the safety between devices.

4.3 Proposed Scheme

The blob, i.e., the attestation output of the device created by the TrustZone-based TEE, should contain information which can be used to check the safety of the target device on the attestation server, and also the integrity of the blob must be guaranteed. A public-key-based signature is used to ensure the integrity when the blob is composed.

The blob configuration is as follows. The notations are summarized in Table 1. The TEE can take a nonce as a variable factor to calculate the measurement or can use a fixed measurement. In the first case, the measurement value is calculated according to an external measurement request. In the second case, the measurement value recorded when the device boots is used. The result of each measurement on request is a function of the nonce.

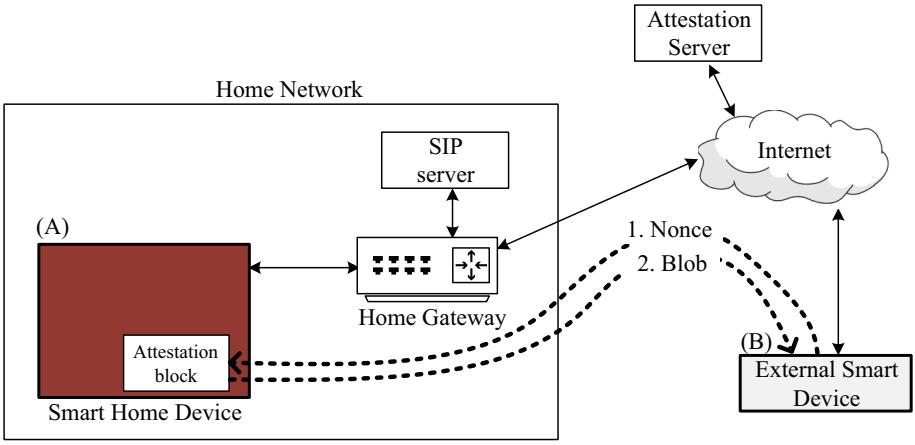

Fig. 7 System model when the security check of an internal smart device should be performed

Table 1 Notations used in the attestation procedures

Notation	Definition
M	Measurements of the device
AS	Attestation server
$h(data)$	Hash operation over data
$E_{Pub_A}[data]$	Encryption of data using A's public key
$D_{Pri_A}[data]$	Signing or decryption of data using A's private key
Pri_A	Private key of A
Pub_A	Public key of A
$Sign_A[data]$	Digital signature with Private Key of A over data. Same with $D_{Pri_A}[data]$
$Cert_A$	Certificate of A
$\|\|$	Message concatenation
N_A	Nonce obtained from an attestation server in device A
B_A	Blob calculated by TEE in device A
V_A	Verdict obtained from an attestation server in device A

$$M = f(nonce, ...) \tag{1}$$

This measurements and various eigenvalues make up the data for the blob.

$$\begin{aligned} data = {} & measurements||deviceIMEI||nonce \\ & ||packageName||creationTime \\ & ||deviceImageVersion \end{aligned} \tag{2}$$

The blob consists of the data, the corresponding signature value, and a certificate. Signing is accomplished using a hash function and public key encryption.

$$blob = data||Sign_{device}[h(data)]||Cert_{device} \tag{3}$$

In the above formula, $Cert_{device}$ includes the public key of the device and the signature of the public key made by the attestation server.

$$Cert_{device} \ni Pub_{device}, Sign_{AS}[h(Pub_{device})] \tag{4}$$

In addition, $Sign_{AS}[h(Pub_{device})]$ included in the blob is the value signed with the private key of the device paired with the public key included in the certificate. Signing using the public key cryptography guarantees the integrity of the transferred data. This blob is sent to the attestation server through the verifier. The attestation server checks that the $Cert_{device}$ in blob is valid using attestation server's public key, and then checks whether $Sign_{device}[h(data)]$ is correct using the device's public key incorporated in the certificate. The signature value is generated using the private key of a device as in Eq. 5 and verified using the public key of a device as in Eq. 6.

$$Signing : D_{Pri_{device}}[h(data)] \tag{5}$$

$$Verifying : E_{Pub_{device}}[E_{Pri_{device}}[h(data)]] = h(data) \tag{6}$$

After it is confirmed that all of the above are correct, the attestation server checks whether the *nonce* included in the data is the nonce issued during attestation within a valid time after creation. Subsequently, the measurement value is checked using the version information and International Mobile Equipment Identity (IMEI) which the unique value of the device, and the device is finally concluded to be safe.

The basic algorithm for the remote attestation is explained in Algorithm 1. There are three entities of a verifier, a prover, and an attestation server. A verifier decides whether a prover is trusted or not. A prover makes a blob to prove that it can be trusted. A attestation server manages a lot of nonce and checks that a blob is correct or wrong based on a given nonce. A communication between a verifier and a prover is accomplished via SIP.

Algorithm 1 Basic algorithm for the remote attestation

Require: Verifier and Prover are registered in SIP server
Ensure: verdict==Yes
1: Verifier: Nonce request → AS
2: AS: Nonce (N_v) generation
3: AS: N_v → Verifier
4: Verifier: Embedding N_v in SDP in $INVITE$ SIP message
5: Verifier: SIP message ($INVITE$) → Prover
6: Prover: Fetching N_v in SIP message
7: Prover: Calculating blob (B_p) based on N_v in device TEE
8: Prover: Embedding B_P in SDP in *200 OK* SIP message
9: Prover: SIP message (*200 OK*) → Verifier
10: Verifier: Fetching B_p in SIP message
11: Verifier: B_p & N_v → AS
12: AS: Checking B_p & N_v and outputs verification result (V_v)
13: AS: V_v → Verifier
14: Verifier: Parsing V_v and check *verdict* field
15: **if** verdict == Yes **then**
16: continue the session
17: **else**
18: refuse the session and stop the communication with Prover
19: **end if**

The detailed explanation of the attestation protocol based on the Algorithm 1 is as following. The attestation protocol between the two devices can be divided into one-way single attestation and two-way mutual attestation. One-way attestation serves to check the safety of external smart devices only, and it is not necessary to check the safety of smart home devices. The detailed communication protocol is shown in Fig. 8. In this case, the smart home device becomes a verifier and the external smart device becomes a prover.

The SIP session procedure for one-way device attestation is as follows.

1. The external smart device A attempts to send a SIP INVITE message to connect to device B inside the smart home.
2. Device B that receives the SIP INVITE message submits a request to the attestation server to obtain a nonce.
3. Device B receives a nonce from the attestation server.
4. This nonce is transmitted to device A as the data of the N_B parameter, which is the Session Description Protocol (SDP) parameter of the SIP 200 OK message.
5. Device A reads the N_B value of the 200 OK message and calls the Knox API to get B_A, a blob data, which is the output of device attestation using the Knox platform.
6. Device A sends the B_A to device B in a SIP message via Re-INVITE. The added parameter at this time is defined as the B_A. In the three-way-handshake method of INVITE, 200 OK and ACK for session setup, if the SDP parameter is included in INVITE, only the message header is transmitted without data in the ACK. That is, because the B_A cannot be transmitted in ACK, it is transmitted using SIP Re-INVITE which is a message after ACK.
7. Device B that received the B_A sends it to the attestation server to receive a verdict, V_B, which is the safety result of the external smart device, to check whether the external smart device can be trusted. If the device cannot be trusted, the connection between the two devices is refused by device B.

In the case of mutual attestation, attestation is performed by exchanging additional data for the other way one-way attestation. When using this method, the external smart device checks the safety of the smart home internal device, and the smart home internal device checks the safety of the external smart device. The detailed communication protocol is similar to that shown in Fig. 8. Figure 9 shows the additional flow of data. During the step to verify device A, device A becomes the prover and device B becomes the verifier. During the step of verifying device B, device B becomes the prover and device A becomes the verifier. Mutual attestation adds extra fields to the SDP in the SIP to perform both processes in one SIP session.

The procedures for mutual device attestation are as follows.

1. The external smart device A attempts to send an INVITE message to device B for access to smart home device B. Before this operation, a nonce should be received from the attestation server.
2. This nonce is transmitted as the data of the N_A parameter, which is an additional parameter defined in the SIP INVITE message.
3. Receiving the INVITE message, device B obtains the value of the N_A parameter from the message and gives this value to the KNOX platform's attestation API as input to obtain the B_B. At the same time, it asks the attestation server for another nonce, N_B.

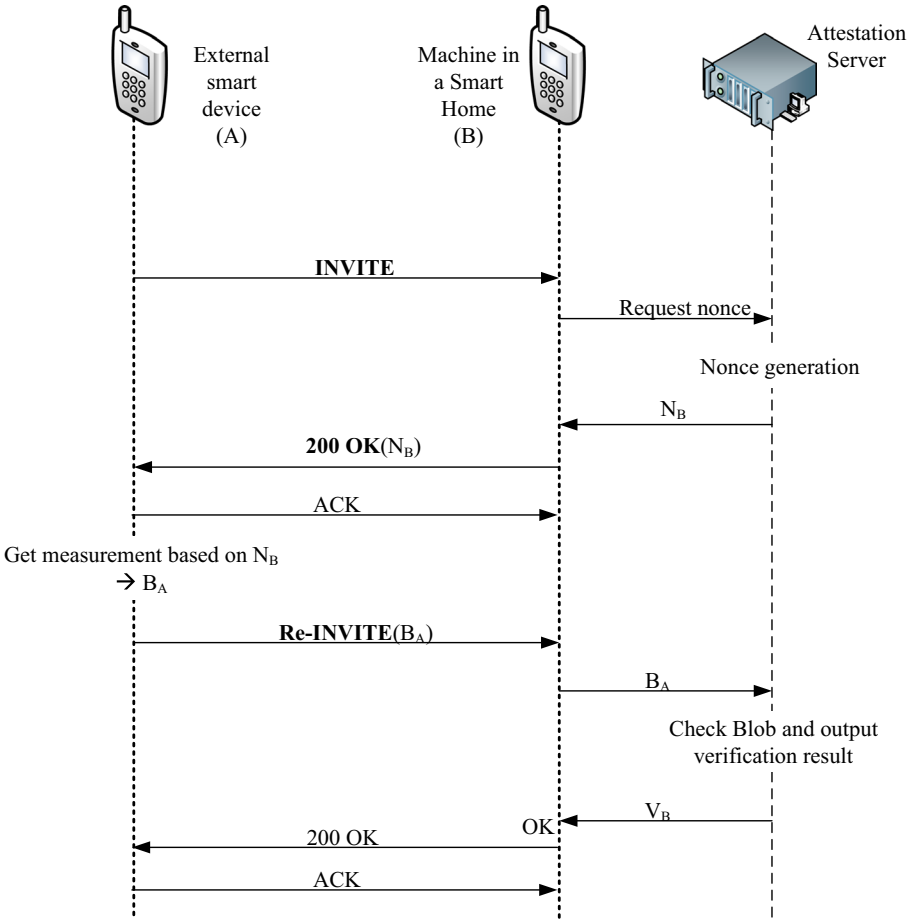

Fig. 8 SIP session setup for the one-way device attestation

4. The B_B and N_B of device B are sent to device A in the SDP parameter of the SIP 200 OK message.
5. Device A sends the B_B to the attestation server to receive a verdict, V_A, a value regarding device B's safety. At the same time, it reads the N_B value of the 200 OK message and gets a blob, B_B, the output of attestation function from the Knox platform.
6. Device A sends the attestation output B_A to device B in the form of a SIP Re-INVITE message. The parameter at this time is the B_A.
7. Receiving B_A, device B sends it to the attestation server to verify that device A can be trusted by receiving a verdict, V_B, the safety result for device A. If all of these procedures are successfully performed, the external device can check the safety of the smart home device, and the smart home device can check the safety of the external smart device.

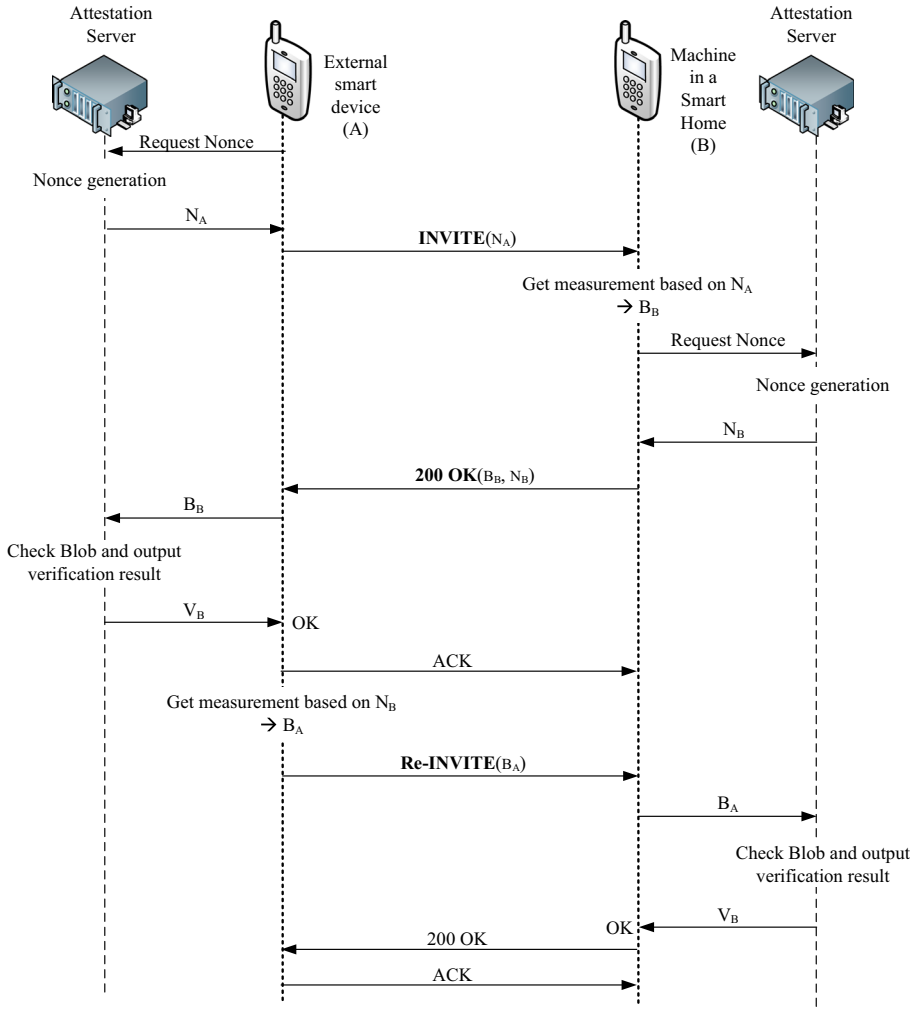

Fig. 9 SIP session setup for mutual device attestation

4.4 Analysis of the Proposed Scheme

Using the proposed method, two devices in the communication channel can trust each other. Here, trust means that a peer device performs only normal and intended operations. If a peer device is not trusted, it is assumed that the device is compromised and will undertake unexpected behavior irrespective of the intentions of the device owner. In an untrusted situation, privacy is compromised, secure information between two devices will be leaked and unintended operations will occur through control of the compromised device. In the IoT environment, it is important to know precisely whether the communication party is secure or not. We can say that the proposed method provides a reliable environment in the smart home.

The method described above has the advantage of checking the safety of the device using the smart device TrustZone which is implemented in most ARM CPUs recently, but it also has the following limitations. First, an attestation server, a separate server that performs nonce management and verification of the measurement data generated by TrustZone, is needed. Because an attestation server built by a smart device manufacturer has a database of information about commercial devices, and it knows the device image and kernel image information operating on such devices, the measurement values can be verified without any problems. If an individual or a company wants his own attestation server, he has to build and manage an attestation server independently. This requires the cooperation of the manufacturer. The difficulty in managing an independent attestation server is the limitation. Next, an individual device requires a separate procedure to create a private key with which to sign the measurement and a public key for signature verification, while also embedding these keys in the device's TEE. Because these keys are security-sensitive data, there should be no key injection process after the market release of the commercial product. These keys should be injected during the factory process. Of course, when using an attestation server built by the manufacturer, there are no special considerations because the above conditions are provided.

In the next section, we design a testbed for mutual attestation that can be experimented on to verify the proposed method. We also explain how to implement it.

5 Testbed Design and Implementation

5.1 Testbed Structure

The following experimental testbed was constructed to check the attestation operation of the device using the SIP, an application protocol used between smart devices and a home network.

A hypothetical scenario is the process of a call setup between a telephone device inside a home network and an external smart device. Each device performs attestation on the other device. In case of verification failure, it refuses to provide service. Specifically, it disconnects the call setup. This protects sensitive data from being transmitted to the other side. Although channel data between A, B, and the SIP server should be protected by SSL in real environment, encryption is omitted to monitor the raw data in the original form during this test.

The smart devices used in the experiments were Samsung smartphones. We used a commercial smartphone by Samsung Electronics that can utilize the Knox platform [15, 28] to check the attestation operation. To the best of our knowledge, Samsung's Knox platform is the only commercially available smart device platform that can perform device attestation using TrustZone. The attestation API of the Knox platform was called to check the safety of the device, and the modified source code of Linphone [6], an open-source-based Android smartphone app, was used to test voice data transmission through a SIP. The Linphone server and an IPTEL [5] server, providing free services as

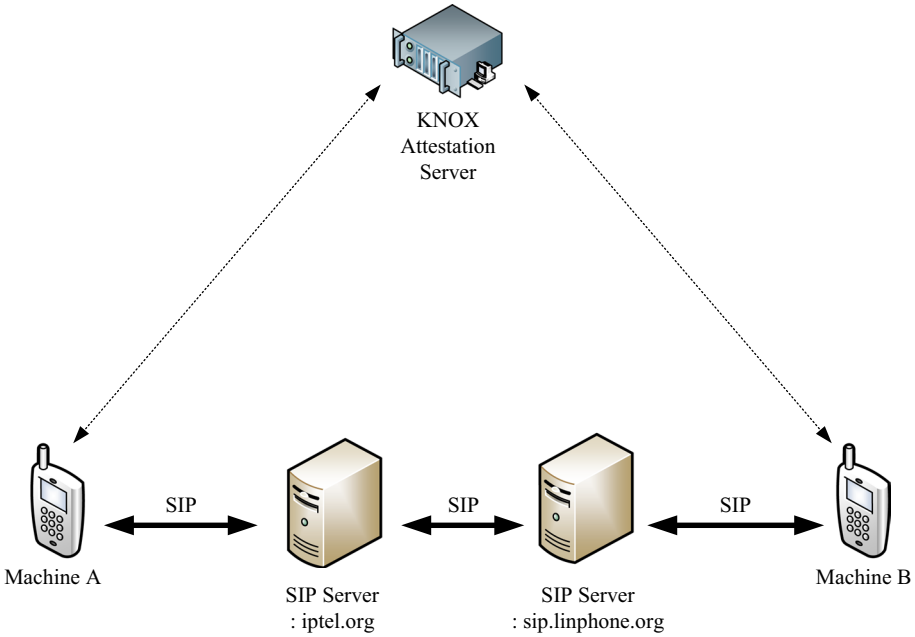

Fig. 10 Experimental setup for the remote attestation test

a SIP server for relaying SIP signals were used. Although only one SIP server may be configured, two servers were used to confirm that there were no compatibility problems.

For the experiment, Machine A was registered on the SIP server of IPTEL and Machine B was registered on Linphone's SIP server, as shown in Fig. 10. A SIP session example in this scenario is a session for voice data transfer between Machine A and Machine B. The session setup process for voice transmission between Machine A and Machine B is performed through the SIP, and when the setup is successfully completed, Machine A and Machine B undertake voice communication with each other through the RTP [43] which is a media data packet delivery protocol.

5.2 Remote Attestation Protocol Design and Implementation

In accordance with the proposed attestation method, the source code of the Linphone application was modified to add parameters for attestation to the SDP so that the corresponding operation could be performed at every step of the SIP. The attestation algorithms using the SIP for sender's part and receiver's part are shown in Algorithm 3 and Algorithm 4, respectively. Algorithm 2 explains the common functions used in the sender and the receiver's algorithms. In these algorithms, the sender is the session initiation part and the receiver is the peer part. The abbreviated notations in these algorithms are defined in Table 2. The overall operation sequence is shown in Fig. 11.

Algorithm 2 Common functions

function GetNonce()
    Pass In: nothing
    Requesting *nonce* to the attestation server
    Pass Out: *nonce*
end function

function GetBlobInMessage(SIP message)
    Pass In: SIP message
    Parsing SIP message
    Fetching *blob* binary
    Pass Out: *blob*
end function

function GetNonceInMessage(SIP message)
    Pass In: SIP message
    Parsing SIP message
    Fetching *nonce* binary
    Pass Out: *nonce*
end function

function BlobCalculationKnox(nonce)
    Pass In: *nonce*
    Calculating *blob* in the Knox Platform
    Pass Out: *blob*
end function

function GetResultBasedOnBlob(blob)
    Pass In: *blob*
    blob → the attestation server
    Receiving result for the *blob*'s correctness
    Pass Out: result
end function

Algorithm 3 Procedure on the Sender's Application

Require: Sender is registered in SIP server
Ensure: verdict==Yes
N_S = GetNonce()
N_S → SDP in *INVITE* SIP message
INVITE SIP message → SIP server
wait Receiver's *200 OK* SIP message
B_R = GetBlobInMessage(200 OK message)
N_R = GetNonceInMessage(200 OK message)
B_S = BlobCalculationKnox(N_R)
Result = GetResultBasedOnBlob(B_R)
result parsing
verdict field fetching in result
if verdict == Yes **then**
    B_S → SDP in *Re-INVITE* SIP message
    Re-INVITE SIP message → SIP server
    continue the session
else
    refuse the session and stop the operation of the application
end if

Algorithm 4 Procedure on the Receiver's Application

Require: Receiver is registered in SIP server
Ensure: verdict==Yes
1: **wait** Sender's *INVITE* SIP message
2: N_S = GetNonceInMessage(INVITE)
3: B_R = BlobCalculationKnox(N_S)
4: N_R = GetNonce()
5: $N_R, B_R \rightarrow$ SDP in *200 OK* SIP message
6: *200 OK* SIP message → SIP server
7: **wait** Sender's *Re-INVITE* SIP message
8: B_S = GetBlobInMessage(Re-INVITE)
9: result = GetResultBasedOnBlob(SB)
10: result parsing
11: verdict field fetching in result
12: **if** verdict == Yes **then**
13: continue the session
14: **else**
15: refuse the session and stop app's operation
16: **end if**

Because we used a third-party commercial SIP server operated irrespective of the proposed method, there were no modifications of the SIP server part. In the final step of attestation, the verdict received from the attestation server was checked to determine whether the session connection would continue.

The attestation server is managed by Samsung on the Internet and provides https service as a protocol for attestation operation with the devices. For the http protocol implementation in the Linphone app, the open source OkHttp package [7] was imported. This http package is used during the transmission of the nonce, blob, and verdict information related to the attestation process.

There are two main roles of an attestation server. The first is the management of the nonce. When there is a request for a nonce, the attestation server must create a non-duplicable value and retain this value in memory. When a nonce-blob pair is received, the server must check the elapsed time after the creation of the nonce relative to the received time. There should be a limitation on the elapsed time to reduce the security surface. This is implemented to prevent a replay attack. The second role is to check that the value is correct through a verification process using a public-key-based signing and a blob calculation process, and to notify the verifier.

6 Experiment and Result Analysis

6.1 Experimental Setup

The basic experimental environment was configured as shown in Fig. 10, and the Linphone application was modified to include the attestation protocol on smart devices A and B. The IP addresses and SIP addresses of A and B are configured as shown in Table 3. Machine A was registered with the account name of *kajae1* at iptel.org, and Machine B was registered with the account name of *kajae2* at linphone.org.

Table 2 Notations in the attestation application using Knox

Notation	Definition
N_S	Nonce of the sender which is obtained from an attestation server
N_R	Nonce of the receiver which is obtained from an attestation server
B_S	Blob calculated in the sender
B_R	Blob calculated in the receiver
V_S	Verdict of the sender which is obtained from an attestation server
V_R	Verdict of the receiver which is obtained from an attestation server

Twelve different smartphones were used as smart devices to test the attestation function in this experiment and the experimental results in those devices were analyzed. The basic information about smartphones and a tablet used in the experiment are shown in Table 4.

6.2 The Evaluation Results

To check the SDP parameters related to attestation in the SIP packet between Machine A and B, the capturing of network packets was performed using Wireshark [11] PC program and WiFi access point. The capture point is between Machine A and the SIP server. It was shown that all SDP parameters in each SIP message during each step were included in the SIP packets.

The verifier sends the blobs received from the prover to the attestation server to check the status of the prover. The verifier then receives the verdict contained in the response from the attestation server. The procedures between a device and an attestation server are explained in Figs. 12 and 13. Figure 12 shows the case when two devices are safe and not compromised. In the figures, ① and ①′ represent the verdict request to the attestation server. On the other hand, ② and ②′ represent the verification result from the attestation server for the request. The procedure of ① corresponds to the procedure [9. B_R] in Fig. 11. and the procedure of ② corresponds to the procedure [10. V_S] in Fig. 11. The procedure of ①′ corresponds to the procedure [13. B_S] in Fig. 11. and the procedure of ②′ corresponds to the procedure [14. V_R] in Fig. 11.

There are several fields that indicate unique information of the device and several fields that show security marks in the verification result, the verdict data. When there is no security problem, the *verdict* field of the verdict data has a value of "*Yes*." If the result is "*verdict*"="*No*," no further sessions should proceed. The verdict contents shown in Fig. 12 indicate that neither device is compromised because the verdict field values of both devices are all "*Yes*." Among the fields, *tamperBit* is related to whether the smartphone binary image has changed. If the smartphone is overwritten with a custom image binary for rooting instead of the manufacturer's pure binary, *tamperBit* changes to 1 and the verdict result is always "*No*." This is a value that is fused into the hardware, so it is cannot be restored to 0. Whenever attestation is performed, this value is checked to determine whether it is the manufacturer's smartphone binary image. The *tamperBit* field of 1 represents that the smartphone manufacturer's binary was replaced with a custom image, and the device is no longer reliable.

For comparison, another smartphone rooted using a custom recovery image is used in the experiment. The verdict content in this case is shown in Fig. 13. The data of the verdict

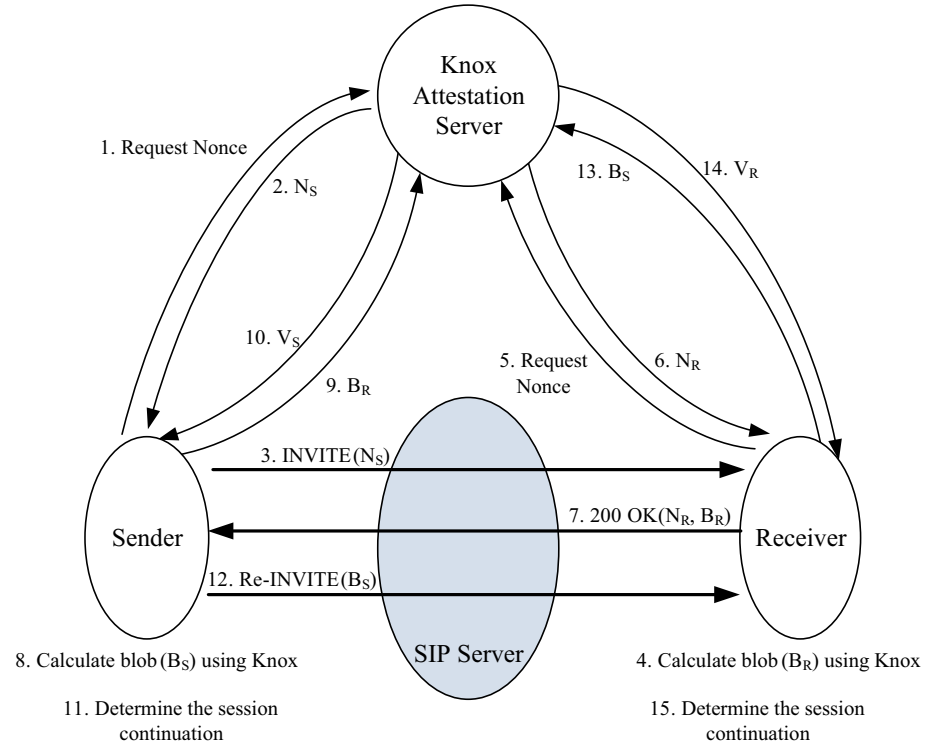

Fig. 11 Flow sequences of remote attestation of this experiment

Table 3 Connection configuration of machines A and B

Device	IP Address	SIP Address
A	192.168.10.173	kajae1@iptel.org
B	192.168.10.221	kajae2@linphone.org

field in the verdict data on the left is "*Yes*," indicating that device B is is not compromised. The data of the verdict field on the right verdict data is "*No*." This indicates that device A is compromised. In this case, the device B is assured that its peer device is compromised, and further operation should be stopped.

Smartphones A and B, which have Knox platforms, could connect successfully to each other after verifying the respective attestation results through each Linphone app, and after the connection was made it was verified that the transmission and reception of RTP media data were accomplished without any problems in the absence of a compromised device. When there was a compromised device, the verdict was checked and no further operations were performed. These verdict check tests were performed 100 times, sufficient to assess the operation reliability for each configuration, with the verdict fields always having correct values. This is also a natural result of the circuit design of the device.

An additional experiment was performed to assess whether voice data were leaked on the compromised smartphone. In this experiment, another smartphone was rooted using the custom image, and the "tinyalsa.so" library file was replaced with one capable of leaking

Table 4 Android versions, Knox versions and CPU specification of smartphones used in the experiment

Name	Android version	Knox version	CPU
Galaxy A8	8.0.0	3.1	2× 2.2 GHz ARM Cortex-A73
			6× 1.6 GHz ARM Cortex-A53
Galaxy A70	9	3.3	2× 2.0 GHz Kryo 460
			6× 1.7 GHz Kryo 460
Galaxy A90	9	3.4	1× 2.84 GHz Kryo 485
			3× 2.42 GHz Kryo 485
			4× 1.8 GHz Kryo 485
Galaxy Fold	9	3.3	1 2.84 GHz Kryo 485
			3× 2.42 GHz Kryo 485
			4× 1.8 GHz Kryo 485
Galaxy Note 10	9	3.4	2× 2.73 GHz Exynos M4
			2× 2.4 GHz ARM Cortex-A75
			4× 1.95 GHz ARM Cortex-A55
Galaxy Note 10 Plus	9	3.2.1	1× 2.84 GHz Kryo 485
			3× 2.42 GHz Kryo 485
			4× 1.8 GHz Kryo 485
Galaxy S8	9	3.2.1	4× 2.3 GHz Exynos M2 Mongoose
			4× 1.7 GHz ARM Cortex-A53
Galaxy S8 Plus	9	3.2.1	4× 2.3 GHz Exynos M2 Mongoose
			4× 1.7 GHz ARM Cortex-A53
Galaxy S9	8.0.0	3.1	4× 2.7 GHz Exynos M3 Mongoose
			4× 1.79 GHz ARM Cortex-A55
Galaxy S9 Plus	8.0.0	3.1	4× 2.7 GHz Exynos M3 Mongoose
			4× 1.79 GHz ARM Cortex-A55
Galaxy S10 Plus	9	3.3	2× 2.73 GHz Exynos M4
			2× 2.31 GHz ARM Cortex-A75
			4× 1.95 GHz ARM Cortex-A55
Galaxy Tab S6	9	3.4	1× 2.84 Hz Kryo 485
			3× 2.42 GHz Kryo 485
			4× 1.8 GHZ Kryo 485

sound stream data. Several steps were taken to disable the Android dm-verity function which guarantees system directory integrity. Finally, all voice data on this smartphone could be transferred to a third party.

In the experiment in which the compromised smartphone, rooted and with the tinyAlsa library patched, is used for communication instead of a normal smartphone, the verdict content is similar to that shown in the left part of Fig. 13. The result indicates that the verdict is "*No*," meaning that the target smartphone is compromised and should no longer be used for communication. If this attestation method is not used, the voice data of VoIP communication would be leaked without any restriction in the rooted smartphone with the tinyAlsa library patched as in the experiment. For the prevention of further voice data leakage in the rooted smartphone, this attestation method should be used. Also when the verdict field from the attestation server shows "*No*," communication with the target device should be restrained.

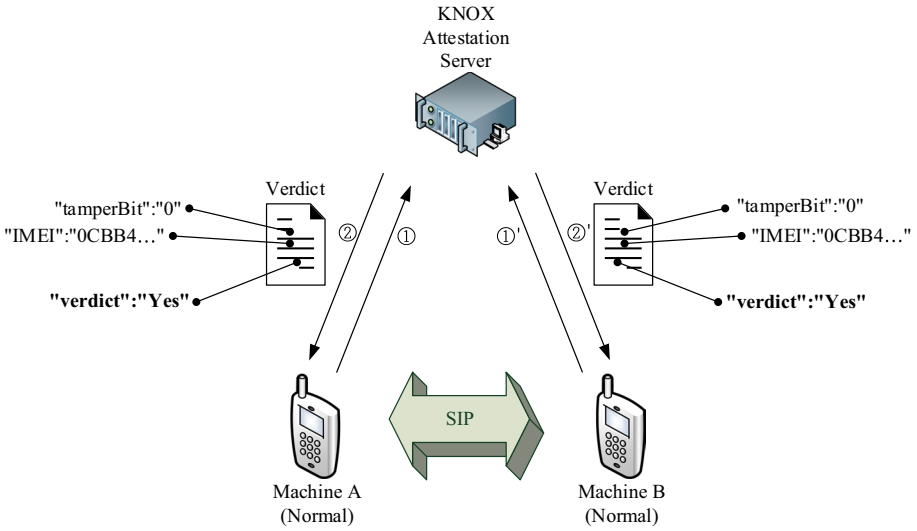

Fig. 12 Verdict content in the absence of a compromised device

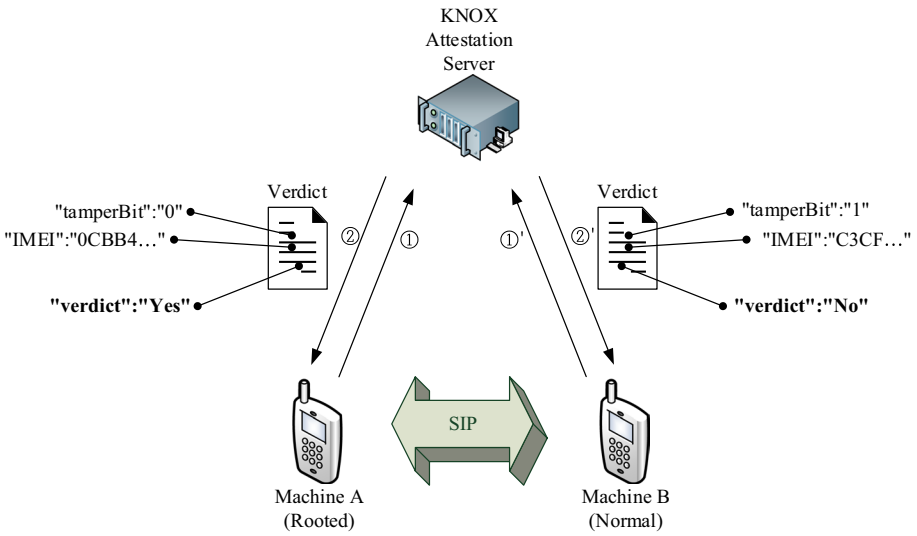

Fig. 13 Verdict content in the presence of a compromised device

The latency times of each of the operations, referring to the time required to obtain a nonce from the attestation server, the time required to calculate the blob on the Knox platform, and the time required to obtain a verdict from the attestation server, are shown in Figs. 14, 15 and 16 for the test smartphones. All the graphs were shown in the form of box plots, which represent minimal, first quartile, median, mean, third quartile, and maximum value.

The time for obtaining nonce can be composed of network latency and nonce generation time as follows. Network latency is the sum of t_{req} and t_{res}.

$$t_{nonce} = t_{req} + t_{res} + t_{ng} \tag{7}$$

where:

t_{nonce} = nonce obtaining time in device
t_{req} = the latency of request packet from a device to an attestation server
t_{res} = the latency of response packet from an attestation server to a device
t_{ng} = nonce generation and management time in an attestation server

Similarly, the time for obtaining verdict can be composed of as follows.

$$t_{verdict} = t_{req} + t_{res} + t_{veri} \tag{8}$$

where:

$t_{verdict}$ = verdict obtaining time in device
t_{req} = the latency of request packet from a device to an attestation server
t_{res} = the latency of response packet from an attestation server to a device
t_{veri} = device verification time based on request data in an attestation server

Network latency, $t_{req} + t_{res}$, is a variable value depending on the network environment, not on the test device. As the server processing time, t_{ng} or t_{veri}, is assumed to be similar for all test devices, the times of t_{nonce} or $t_{verdict}$ are also similar. Figures 14 and 15 shows that the nonce obtaining times or the verdict obtaining times are similar for all 12 devices.

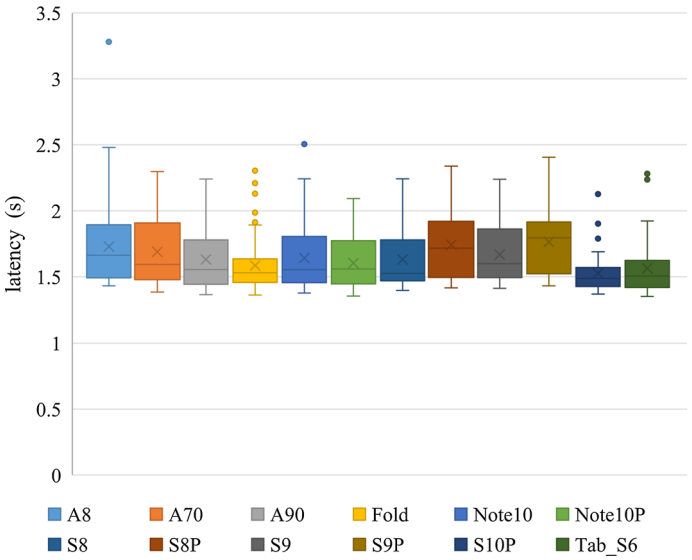

Fig. 14 Latencies (t_{nonce}) when obtaining the *nonce* from an Attestation Server for 12 devices

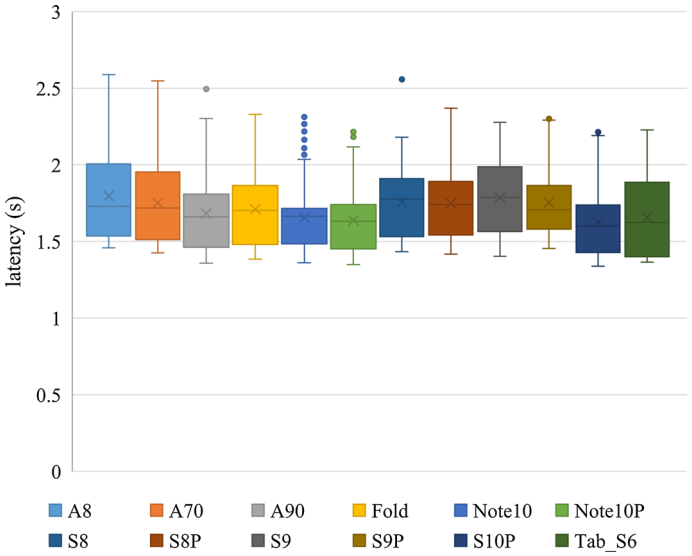

Fig. 15 Latencies ($t_{verdict}$) when obtaining the *verdict* using Knox platform for 12 devices

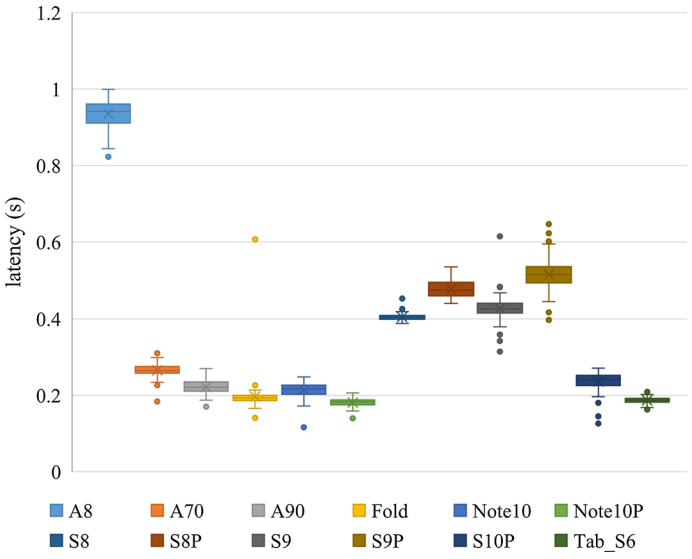

Fig. 16 Latencies when calculating the *blob* from an attestation server for 12 devices

The times required to calculate the blob shown in Fig. 16 are different depending on the device type. This is related to the device performance. As the figure shows, the device of Galaxy A8 is worst and the device of Galaxy Note 10 Plus is best.

The latency values are summarized in Table 5 and shown as a graph in Fig. 17. This table shows that the latency time when obtaining data such as a nonce and a verdict from

Table 5 Average values of nonce obtaining time from the attestation server, the blob calculation time on the Knox platform, and the verdict checking time from the attestation server

Device	Obtaining nonce	Calculating blob	Obtaining verdict	Sum
Galaxy A8	1.729	0.935	1.798	4.463
Galaxy A70	1.690	0.265	1.753	3.708
Galaxy A90	1.631	0.221	1.683	3.536
Galaxy Fold	1.585	0.196	1.710	3.492
Galaxy Note 10	1.643	0.214	1.657	3.514
Galaxy Note 10 Plus	1.603	0.181	1.637	3.421
Galaxy S8	1.632	0.404	1.756	3.792
Galaxy S8 Plus	1.742	0.477	1.749	3.968
Galaxy S9	1.669	0.427	1.787	3.882
Galaxy S9 Plus	1.764	0.515	1.753	4.032
Galaxy S10 Plus	1.527	0.236	1.625	3.388
Galaxy Tab S6	1.564	0.187	1.656	3.406

unit: s

the attestation server is not large considering practical usage. Because calculating the blob is an inner process on the Knox platform, the time is very small compared to other network-related operation times. As Fig. 17 shows, the blob calculation time is very small compared to other two values of nonce obtaining time and verdict obtaining time. This value is related to only the device itself. This is the reason why the latencies vary between devices. It is considered that in general the effect of blob calculation time is small as the

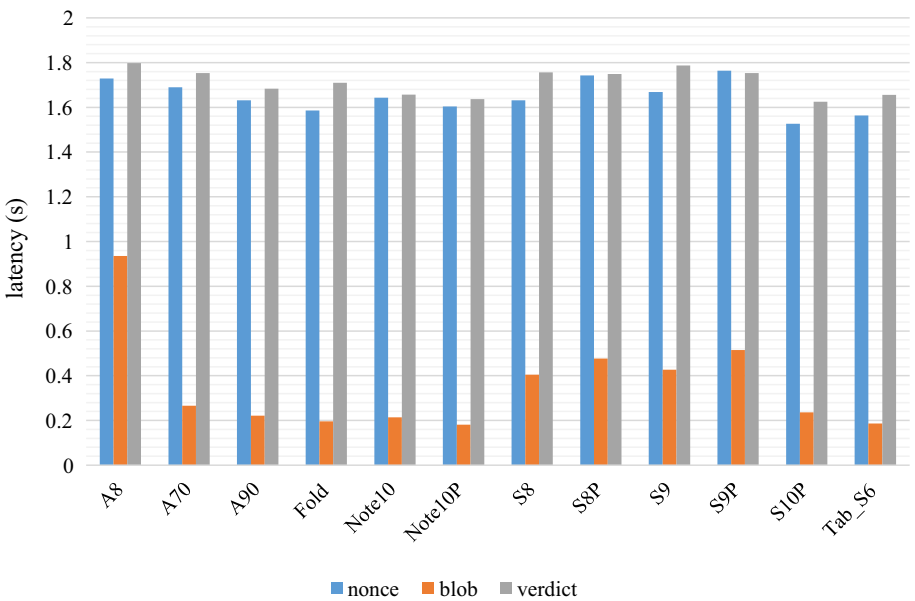

Fig. 17 Average latencies of obtaining nonce, calculating blob, checking verdict in each device

CPU performance of smart devices is getting better. For one SIP session setup, there is additional time of 3.3 $s \sim 4.5$ s on average for entire attestation. It is assumed that this additional time should be accepted to ensure the proper checking of the security of the devices.

These values were influenced by the location of the server, the connection speed of the network, the time required to generate and manage the nonce, and the time to check the validity of the blob at the attestation server, But in general, the entire attestation operation can be completed within a few seconds, making it practical to apply.

When a smartphone is rooted and the relevant library is patched for malicious activity, all data on the smartphone is assumed to be exposed to the unknown external attacker. This is a critical weakness. Hence, a checking procedure for the devices of other parties is essential for secure communication.

Because new methods of attack are appearing daily, we cannot be sure that such a method can prevent all attacks, but at least it is assumed that known kernel attacks including rooting can be detected and prohibited from executing malicious operations.

A limitation of these experiments is that smart devices on which it will work are limited to devices with the Samsung Knox platform installed. However, thus far only Samsung devices can provide developers with an attestation environment using TrustZone. Accordingly, experimenting with the proposed method universally was limited.

As a further work, experiments in various environment such as poor quality network situation will be accomplished. Based on the experiment, the attestation feasibility for this environment will be analyzed. Then, the mathematical complexity analysis for the attestation dependent on device performance and algorithms will be evaluated. A way to overcome the limitation of the proposed method will be considered.

7 Conclusions

Remote attestation can be considered as a method by which the level of the safety is checked between devices in a smart home environment. In this paper, we designed an attestation protocol between devices using the scalable SIP of the type used in a smart home and implemented the protocol on a smartphone with a modified open source application. In the attestation experiment, the testbed was composed of a SIP server, smart devices, and an attestation server. The experiment using twelve different smart devices showed that the nonce obtaining and the verdict obtaining from the attestation server took a few seconds which was larger than the blob calculation time in the device, because the blob calculation time was dependent on the device itself and in general it is small than the network related time. Depending on the performance of the device, the blob calculation time varied somewhat from device to device. The additional time of 3.3 $s \sim 4.5$ s which is necessary for the entire attestation is assumed to be endurable for the smart home network security.

This method can be applied directly in the field because it utilizes the functions of actual commercial smartphones instead of the experiment board-level type, and the experimental results confirm that there were no problems related to attestation operations. The attestation experiment is limited to a specific manufacturer's smartphone device because the attestation function is implemented only on the Knox platform of Samsung. However, if the attestation protocol using the SIP is adopted as presented in this paper and attestation functions using TrustZone are implemented on the smart devices of other manufacturers', attestation between heterogeneous devices will be possible and will be a universal method of smart home protection from attackers.

With the proposed method, it is possible to assess the safety of devices connected to a smart home, thereby preventing an attack with the exposure of internal privacy information through a compromised external smart device or the unintentional malfunctions of internal devices caused by a compromised device.

References

1. ARM TrustZone. Retrieved August 20, 2019 from https://developer.arm.com/ip-products/security-ip/trustzone.
2. firmware.mobi. Retrieved August 29, 2019 from https://desktop.firmware.mobi.
3. Gartner statistics for connected things. Retrieved September 4, 2019 from https://www.gartner.com/en/newsroom/press-releases/2017-02-07-gartner-says-8-billion-connected-things-will-be-in-use-in-2017-up-31-percent-from-2016.
4. Intel Software Guard Extensions (SGX). Retrieved August 20, 2019 from http://software.intel.com/en-us/sgx.
5. IPTEL. Retrieved August 20, 2019 from https://www.iptel.org.
6. Linphone. Belledonne Communications SARL. Retrieved August 20, 2019 from https://www.linphone.org.
7. OkHttp. Retrieved August 20, 2019 from https://square.github.io/okhttp.
8. Samsung Enterprise Alliance Program. Retrieved August 20, 2019 from https://seap.samsung.com/html-docs/android/Content/knox-attestation.htm.
9. Team Win Recovery Project. Retrieved August 29, 2019 from https://twrp.me/Devices.
10. Trusted Platform Module. Retrieved August 20, 2019 from http://trustedcomputinggroup.org/work-groups/trusted-platform-module.
11. Wireshark. Network protocol analyzer. Retrieved August 20, 2019 from https://www.wireshark.org.
12. Amiri Sani, A. (2017). SchrodinText: Strong protection of sensitive textual content of mobile applications. In *Proceedings of the 15th annual international conference on mobile systems, applications, and services* (pp. 197–210). https://doi.org/10.1145/3081333.3081346.
13. Arbaugh, W., Farber, D. J., & Smith, J. M. (1997). A secure and reliable bootstrap architecture. In *Proceedings of 1997 IEEE symposium on security and privacy* (pp. 65–71). https://doi.org/10.1109/SECPRI.1997.601317.
14. Arias, O., Rahman, F., Tehranipoor, M., & Jin, Y. (2018). Device attestation: Past, present, and future. In *Proceedings of 2018 design, automation & test in europe conference & exhibition (DATE)* (pp. 473–478). https://doi.org/10.23919/DATE.2018.8342055.
15. Atamli-Reineh, A., Borgaonkar, R., Balisane, R. A., Petracca, G., & Martin, A. (2016). Analysis of trusted execution environment usage in Samsung KNOX. In *SysTEX '16: Proceedings of the 1st workshop on system software for trusted execution* (pp. 1–6). https://doi.org/10.1145/3007788.3007795.
16. Bertran, B., Consel, C., Kadionik, P., & Lamer, B. (2009). A SIP-based home automation platform: An experimental study. In *Proceedings of 2009 13th international conference on intelligence in next generation networks* (pp. 1–6). https://doi.org/10.1109/ICIN.2009.5357075.
17. Chifor, B.-C., Bica, I., Patriciu, V.-V., & Pop, F. (2018). A security authorization scheme for smart home Internet of Things devices. *Future Generation Computer Systems*, *86*, 740–749. https://doi.org/10.1016/j.future.2017.05.048.
18. Daş, R., & Tuna, G. (2015). Machine-to-machine communications for smart homes. *International Journal of Computer Networks and Applications*, *2*(4), 196–202.
19. Dhanjani, N. (2013). Hacking lightbulbs: Security evaluation of the Philips hue personal wireless lighting system. Retrieved August 29, 2019 from https://www.dhanjani.com/blog/2013/08/hacking-lightbulbs.html.
20. Eldefrawy, K., Rattanavipanon, N., & Tsudik, G. (2017). HYDRA: HYbrid Design for Remote Attestation using a formally verified microkernel. In *Proceedings of the 10th ACM conference on security and privacy in wireless and mobile networks* (pp. 99–110). https://doi.org/10.1145/3098243.3098261.
21. Eldefrawy, K., Tsudik, G., Francillon, A., & Perito, D. (2012). SMART: Secure and Minimal Architecture for (establishing dynamic) Root of Trust. In *Proceedings of NDSS* (pp. 1–15).
22. Hammer-Lahav, E. (2010). RFC 5849: The OAuth 1.0 protocol. Internet Engineering Task Force. *(IETF)*, *4*. https://doi.org/10.17487/RFC5849.

23. Handley, M., Jacobson, V., & Perkins, C. (2006). RFC 4566: SDP: Session description protocol. *Internet Engineering Task Force (IETF)*, *7*. https://doi.org/10.17487/RFC4566.
24. Hardt, D. (2012). RFC 6749: The OAuth 2.0 authorization framework. *Internet Engineering Task Force (IETF), 10*. https://doi.org/10.17487/RFC6749.
25. Hill, K. (2014). Baby monitor hacker still terrorizing babies and their parents. Forbes online article 2014. Retrieved August 20, 2019 from https://www.forbes.com/sites/kashmirhill/2014/04/29/baby-monitor-hacker-still-terrorizing-babies-and-their-parents.
26. Hrabovsky, J., Segec, P., Paluch, P., Moravcik, M., & Papan, J. (2016). Usability of the SIP protocol within smart home solutions. *Communications-Scientific letters of the University of Zilina*, *18*(1A), 4–12.
27. Rosenberg, J., Schulzrinne, H., Camarillo, G., Johnston, A., Peterson, J., Sparks, R., Handly, M., & Schooler, E. (2002). RFC 3261: SIP: Session initiation protocol. *Internet Engineering Task Force (IETF), 6*. https://doi.org/10.17487/RFC3261.
28. Kanonov, U., & Woo, A. (2016). Secure containers in Android: the Samsung KNOX case study. In *Proceedings of the 6th workshop on security and privacy in smartphones and mobile devices* (pp. 3–12). https://doi.org/10.1145/2994459.2994470.
29. Kil, C., Sezer, E. C., Azab, A. M., Ning, P., & Zhang, X. (2009). Remote attestation to dynamic system properties: Towards providing complete system integrity evidence. In *Proceedings of 2009 IEEE/IFIP international conference on dependable systems & networks* (pp. 115–124). https://doi.org/10.1109/DSN.2009.5270348.
30. Koeberl, P., Schulz, S., Sadeghi, A.-R., & Varadharajan, V. (2014). TrustLite: A security architecture for tiny embedded devices. In *Proceedings of the ninth European conference on computer systems* (pp. 1–14). https://doi.org/10.1145/2592798.2592824.
31. Komninos, N., Philippou, E., & Pitsillides, A. (2014). Survey in smart grid and smart home security: Issues, challenges and countermeasures. *IEEE Communications Surveys & Tutorials*, *16*(4), 1933–1954. https://doi.org/10.1109/COMST.2014.2320093.
32. Li, W., Li, H., Chen, H., & Xia, Y. (2015). AdAttester: Secure online mobile advertisement attestation using TrustZone. In *Proceedings of the 13th annual international conference on mobile systems, applications, and services* (pp. 75–88). https://doi.org/10.1145/2742647.2742676.
33. Li, W., Luo, S., Sun, Z., Xia, Y., Lu, L., Chen, H., Zang, B., & Guan, Ha. (2018). VButton: Practical attestation of user-driven operations in mobile apps. In *Proceedings of the 16th annual international conference on mobile systems, applications, and services* (pp. 28–40). https://doi.org/10.1145/3210240.3210330.
34. Li, Y., McCune, J. M., & Perrig, A. (2010). SBAP: Software-Based Attestation for Peripherals. In *Proceedings of international conference on trust and trustworthy computing* (pp. 16–29). https://doi.org/10.1007/978-3-642-13869-0_2.
35. Li, Y., McCune, J. M., & Perrig, A. (2011). VIPER: Verifying the Integrity of PERipherals' firmware. In *Proceedings of the 18th ACM conference on computer and communications security* (pp. 3–16). https://doi.org/10.1145/2046707.2046711.
36. Lindemann, R., Baghdasaryan, D., & Tiffany, E. (2014). FIDO UAF protocol specification v1.0. *FIDO Alliance, 12*. https://fidoalliance.org/specs/fido-uaf-v1.0-ps-20141208/fido-uaf-protocol-v1.0-ps-20141208.html.
37. Liu, D. & Cox, L. P. (2014). VeriUI: Attested login for mobile devices. In *Proceedings of the 15th workshop on mobile computing systems and applications* (pp. 1–6). https://doi.org/10.1145/2565585.2565591.
38. Liu, H., Spink, T., & Patras, P. (2019). Uncovering security vulnerabilities in the Belkin WeMo home automation ecosystem. In *Proceedings of 2019 IEEE international conference on pervasive computing and communications workshops (PerCom workshops)* (pp. 894–899). https://doi.org/10.1109/PERCOMW.2019.8730685.
39. Mendoza, S. (2016). Samsung Pay: Tokenized numbers, flaws and issues. In *Proceedings of Black Hat USA* (pp. 1–11).
40. Morgner, P., Mattejat, S., & Benenson, Z. (2016). All your bulbs are belong to us: Investigating the current state of security in connected lighting systems. arXiv preprint arXiv:1608.03732.
41. Nauman, M., Khan, S., Zhang, X., & Seifert, J.-P. (2010). Beyond Kernel-level integrity measurement: Enabling remote attestation for the android platform. In *Proceedings of international conference on trust and trustworthy computing* (pp. 1–15). https://doi.org/10.1007/978-3-642-13869-0_1.
42. Notra, S., Siddiqi, M., Gharakheili, H. H., Sivaraman, V., & Boreli, R. (2014). An experimental study of security and privacy risks with emerging household appliances. In *Proceedings of 2014 IEEE conference on communications and network security* (pp. 79–84). https://doi.org/10.1109/CNS.2014.6997469.

43. Schulzrinne, H., Casner, S., Frederick, R., & Jacobson, V. (2003). RFC 3550: RTP: A transport protocol for real-time applications. *Internet Engineering Task Force (IETF), 7*. https://doi.org/10.17487/RFC3550.
44. Seshadri, A., Luk, M., & Perrig, A. (2008). SAKE: Software Attestation for Key Establishment in sensor networks. In *Proceedings of international conference on distributed computing in sensor systems* (pp. 372–385). https://doi.org/10.1007/978-3-540-69170-9_25.
45. Seshadri, A., Perrig, A., Van Doorn, L., & Khosla, P. (2004). SWATT: SoftWare-based ATTestation for embedded devices. In *Proceedings of IEEE symposium on security and privacy* (pp. 272–282). https://doi.org/10.1109/SECPRI.2004.1301329.
46. Ying, K., Ahlawat, A., Alsharifi, B., Jiang, Y., Thavai, P., & Du, W. (2018). TruZ-Droid: Integrating TrustZone with mobile operating system. In *Proceedings of the 16th annual international conference on mobile systems, applications, and services* (pp. 14–27). https://doi.org/10.1145/3210240.3210338.
47. Zhang, P.-j., Ji, Y.-F., Liu, Y., & Song, X. (2009). Design and implementation of the middleware for smart home gateway based on SIP. In *Proceedings of 2018 33rd youth academic annual conference of chinese association of automation (YAC)* (pp. 489–492). https://doi.org/10.1109/YAC.2018.8406424.

Jaehwan Ahn is a Ph.D. candidate at the Graduate School of Information Security, School of Computing, KAIST, Daejeon, Korea. Also he is a senior researcher at The Affiliated Institute of ETRI. He received his B.S. degree in Electronics Engineering from Sungkyunkwan University, Suwon, Korea at 1999 and his M.S. degree in the Department of Information and Communications Engineering from Gwangju Institute of Science and Technology (GIST), Gwangju, Korea at 2002. His current research interests include security token, mobile security, and IoT security.

Il-Gu Lee is a professor at the Department of Convergence Security Engineering, Sungshin University (SU), Seoul, Korea. Before joining SU in March 2017, he was with the Electronics and Telecommunications Research Institute (ETRI) as a senior researcher from 2005 to 2017, and served as a principal architect and a project leader for Newratek (KR) and Newracom (US) from 2014 to 2017. He received his B.S. degree in electrical engineering from Sogang University, Seoul, Korea, at 2003, and his M.S. degree in the Department of Information and Communications Engineering from Korea Advanced Institute of Science and Technology (KAIST), Daejeon, Korea, at 2005. He also received his M.A. degree in Intellectual Property from KAIST at 2012. He received his Ph.D. degree in the Graduate School of Information Security in Computer Science & Engineering Department from KAIST at 2016. He has participated in the IEEE 802.11 standardization since 2007. He is currently a guest editor of special issue in Electronics. His current research interests are in the area of wireless/mobile networks with an emphasis on information security, networks, wireless circuit and systems. He has authored/coauthored more than 40 technical papers in the areas of information security, wireless networks and communications, and holds about 140 patents.

Myungchul Kim received his B.A. in Electronics Engineering from Ajou University in 1982, M.S. in Computer Science from the Korea Advanced Institute of Science and Technology (KAIST) in 1984, and Ph.D. in Computer Science from the University of British Columbia, Vancouver, Canada, in 1993. Currently, he is with the faculty of the KAIST as Professor in the School of Computing. Before joining the university, he was a managing director in Korea Telecom Research and Development Group during 1984–1997 where he was in charge of research and development of protocol and QoS testing on ATM/B-ISDN, IN, PCS and Internet. He has also served as a member of Program Committees for numerous numbers of conferences and served as chair of the IWTCS'97 and the FORTE'01. He has published over 150 conference proceedings, book chapters, and journal articles in the areas of computer networks, wireless mobile networks, protocol engineering and network security. His research interests include Internet, protocol engineering, mobile computing, and information security.