SCAtt-man: Side-Channel-Based Remote Attestation for Embedded Devices that Users Understand

Sebastian Surminski
University of Duisburg-Essen
Essen, Germany
sebastian.surminski@uni-due.de

Christian Niesler
University of Duisburg-Essen
Essen, Germany
christian.niesler@uni-due.de

Sebastian Linsner
Technical University of Darmstadt
Darmstadt, Germany
linsner@peasec.tu-darmstadt.de

Lucas Davi
University of Duisburg-Essen
Essen, Germany
lucas.davi@uni-due.de

Christian Reuter
Technical University of Darmstadt
Darmstadt, Germany
reuter@peasec.tu-darmstadt.de

ABSTRACT

From the perspective of end-users, IoT devices behave like a black box: As long as they work as intended, users will not detect any compromise. Users have minimal control over the software. Hence, it is very likely that the user misses that illegal recordings and transmissions occur if a security camera or a smart speaker is hacked. In this paper, we present SCAtt-man, the first remote attestation scheme that is specifically designed with the user in mind. SCAtt-man deploys software-based attestation to check the integrity of remote devices, allowing users to verify the integrity of IoT devices with their smartphones. The key novelty of SCAtt-man resides in the utilization of user-observable side-channels such as light or sound in the attestation protocol. Our proof-of-concept implementation targets a smart speaker and an attestation protocol that is based on a data-over-sound protocol. Our evaluation demonstrates the effectiveness of SCAtt-man against a variety of attacks and its usability based on a user study with 20 participants.

CCS CONCEPTS

• **Security and privacy** → **Embedded systems security**; • **Computer systems organization** → *Embedded systems.*

KEYWORDS

attestation; firmware security; IoT; smart speaker

ACM Reference Format:
Sebastian Surminski, Christian Niesler, Sebastian Linsner, Lucas Davi, and Christian Reuter. 2023. SCAtt-man: Side-Channel-Based Remote Attestation for Embedded Devices that Users Understand. In *Proceedings of the Thirteenth ACM Conference on Data and Application Security and Privacy (CODASPY '23), April 24–26, 2023, Charlotte, NC, USA.* ACM, New York, NY, USA, 12 pages. *https://doi.org/10.1145/3577923.3583652*

CODASPY '23, April 24–26, 2023, Charlotte, NC, USA

ACM ISBN 979-8-4007-0067-5/23/04…$15.00
https://doi.org/10.1145/3577923.3583652

1 INTRODUCTION

The Internet of Things (IoT) enhances previously unconnected devices with Internet access. Popular examples are smart lamps, household appliances, security cameras, smart TVs, and smart speakers. Nowadays, IoT devices are ubiquitous: in 2019 the number of connected IoT devices already reached 35.7 billion [57]. It is expected that the global IoT would triple in size over the next 10 years, from an estimated 182 billion dollars in 2020 to more than 621 billion dollars in 2030 [69]. IoT devices often perform important security-critical tasks, for example in a smart door lock, or operate in privacy-sensitive areas, such as a security camera or as a smart speaker [62]. Moreover, IoT devices, like any other computing system, suffer from vulnerabilities and offer a large attack surface [4]. Compromised IoT devices can have severe consequences, which are not limited to the individual user, as the Mirai botnet has shown. This botnet, consisting of consumer devices like routers and IP cameras, performed one of the largest denial-of-service attacks peaking at 1.1 Tbit/s, targeting the OVH hoster and many popular websites [39].

IoT devices are worthwhile targets as typical IoT devices are black-box systems to the user with a limited understanding of threats to security and privacy [72]. Typically, a user does not have any direct control over the software running on the IoT device, but solely configures and maintains IoT devices using a web interface or a companion app [4]. Malware can be deployed before the device was purchased by the user or afterward through backdoors and vulnerabilities. A compromise with malware will not be detected if the functionality does not fail [5]. For instance, 48% of companies reported they are unable to detect whether an IoT device on their network suffers from a breach or is part of a botnet [30].

Remote attestation is a popular method to verify the integrity of a remote and untrusted device. Many attestation methods have been proposed for embedded and IoT devices as well as sensor networks [63]. The main challenge of remote attestation is to obtain trustworthy measurements from an untrusted device. In general, three different attestation approaches exist to address this problem: hardware-based, software-based, and a combination of both. Hardware-based approaches rely on a trusted computing subsystem inside the untrusted device [6, 15, 35, 52, 68]. Hybrid attestation leverages a hardware/software co-design to take trustworthy self-measurements [17, 47]. In contrast, software-based attestation

approaches [7, 61, 64] do not require any special hardware features but rely on precise timing measurements of the attestation function. That is, they rely on the assumption that an attacker cannot accelerate the measurement function [61]. Furthermore, the exploitation of implementation flaws undermines the security of software-based attestation [12]. Hence, there is a common agreement that hardware-based and hybrid attestation schemes are preferable. However, many IoT devices neither feature trusted computing components nor the necessary hardware extensions for hybrid attestation. Replacing these legacy IoT devices to enable hardware-based or hybrid attestation protocols is often not an option in practice; especially considering that the computing hardware is deeply integrated into the respective IoT device. Consequently, software-based attestation is often the only viable approach.

For hardware-based and hybrid attestation schemes, the attested devices feature secret keys to authenticate themselves. However, as there is no secure key storage in software-based attestation, there is no root of trust that allows authentication of the attested device. In fact, any information on the attested device is accessible to the attacker. Hence, an attacker can relay attestation requests to a different or even a simulated device without being noticed by a remote verifier. Such so-called offloading attacks, i.e., forged attestation reports from other devices, are an inherent problem in software-based attestation due to the lack of a secure root of trust.

Contributions. We propose SCAtt-man, the first remote attestation scheme specifically designed to allow user-observable attestation and thereby solve the problem of missing device authentication in software-based attestation. For the first time, we exploit side-channel information (e.g., light or sound) observable by the user to evaluate the attestation result allowing the user to identify the device that is being attested and detect offloading attacks.

Although it has been popular to use side-channels such as sound [76], ambient sound [31, 41, 60], or acceleration [33, 71] for context-based authentication, that is, key exchange or device pairing, we are not aware of any remote attestation scheme that leverages such side-channels. Using such communication channels for attestation is not straightforward, as implementing a secure software-based attestation scheme involves tackling the aforementioned challenges [61]. The usage of side-channels allows secure deployment of software-based attestation on legacy IoT systems without unrealistic hardware requirements and changes. In fact, we demonstrate that users can use their smartphones together with built-in sensors and actuators to attest the IoT device.

Attestation via side channels has the following advantages: (1) Communication is user-observable. (2) Communication is limited to short distances, limiting a remote attacker. (3) The Internet connectivity of the attested device can be interrupted to prevent offloading attacks. (4) The transmission time can be predicted precisely, which is a crucial requirement for software-based attestation. The missing device authentication in software-based attestation is being replaced by the user, who can manually identify the device that is currently being attested. This approach offers users an intuitive way to identify IoT devices and the devices' integrity. This makes SCAtt-man the first framework for user-friendly software-based attestation. SCAtt-man does not require complex profiling or measurements during installation. If the users' smartphone already knows the correct configuration of the attested device, e.g. because it was used to initially configure the device, the smartphone can confirm the correctness of the attested device's configuration, the integrity of the device's software, and detect unwanted modifications or malware. In practice, this attestation functionality can also be integrated into the vendor's companion app that is often used to configure or use the IoT device [4].

In summary, we provide the following contributions:

- We propose SCAtt-man, a new attestation scheme that leverages side-channels for software-based attestation that the user can observe and thereby efficiently solves the root-of-trust problem in software-based attestation.
- SCAtt-man works on legacy hardware avoiding the need for additional hardware modules or actuators. It uses built-in hardware features, such as the built-in microphone and speaker of a smart speaker and a standard smartphone as a verifier.
- We implemented SCAtt-man in a smart speaker based on the popular ESP32 microcontroller and implemented a verifier as an Android app.
- In the evaluation, we show, based on a full end-to-end example, how SCAtt-man detects a real-world attack on a smart speaker via an insecure configuration interface (Section 7.3). In extensive experiments, we verified that SCAtt-man ran without any false positives and negatives, allowing a reliable attestation.
- We performed a user study to evaluate the usability and user experience of SCAtt-man (Section 7.5). Our user study not only showed that SCAtt-man provides good usability, but participants also stated that they actually believe that attestation can detect a device's compromise and that they would use such functionality if their own devices featured such an attestation functionality.

2 BACKGROUND

In this section, we explain the foundations and concepts required to understand SCAtt-man.

2.1 Software-based Remote Attestation

Remote attestation is a security service that allows a device to check the integrity of an untrusted remote device [47]. To do so, the attested system inhabits a proving mechanism that allows taking self-measurements; therefore the attested device is called prover. These attestation reports are sent to an external party, called verifier, for verification. The main challenge of any remote attestation scheme is how to obtain a secure self-measurement of an untrusted device [14]. There exist different approaches to tackle this problem. In hardware-based attestation schemes, the attestation is performed by an isolated trusted computing module [6, 15, 35, 52, 68]. In hybrid attestation schemes, measurements are taken based on a hardware/software co-design [17, 47]. However, both hybrid and hardware-based attestation require special hardware. In contrast, software-based attestation schemes do not require special hardware features and can hence be used on commodity and legacy hardware including IoT devices. In the following, we focus on software-based attestation schemes. We elaborate on hardware-based and hybrid

attestation schemes in Section 8. The security of software-based attestation relies on the execution time of the software on the prover. The verifier initiates the attestation by sending a request to the prover. This request typically includes a nonce to prevent replay attacks. The prover takes this nonce to perform a self-measurement, typically using a hashing function. Thereafter, the prover sends the result back to the verifier. The verifier measures the response time of the prover. If the response time is within a predefined margin, the measurement is assumed to be benign. A delayed response indicates that the prover has been compromised. Obviously, this requires precise measurement of the actual execution time and exact prediction of the expected execution [61]. Hence, this poses strict requirements for the implementation: the attestation process may not be able to be accelerated under any circumstance. Otherwise, the attacker can alter the result of the attestation process without violating timing constraints [12].

2.2 Smart Speakers

Smart speakers are IoT devices that take natural spoken language as input and react appropriately by responding using synthesized voices or performing tasks. These tasks range from the supply of information such as the current time or weather, telling jokes, playing music, over the setting of timers and creation of lists, up to sending messages and controlling other connected IoT devices such as lights or smart locks [34]. Smart speakers are very popular; In 2020, more than 150 million units have been sold worldwide [10]. Studies have shown that users are highly concerned about the security of smart speakers. Although users often do not see threats in devices that do not record audio or video, such as smart plugs or light bulbs, they are aware of the privacy risks of smart speakers and IP cameras [72, 77]. Smart speakers feature microphones, audio-processing hardware, and speakers and constantly listen for a so-called 'wake word' [70]. This wake word starts the interaction with the smart speaker: the smart speaker records the user's voice and sends it to a cloud service for further processing [19]. Smart speakers suffer from a large attack surface as they incorporate a complex architecture [19]. That is, they combine an IoT device with a combination of local and cloud features, including natural language processing techniques. The functionality of smart speakers can often be enhanced with third-party extensions developed by external vendors. For example, there are more than 18,000 English extensions available for Google Assistant [37]. For Amazon's Alexa speech assistant, more than 80,000 so-called skills are available in the US [36]. Malicious extensions can illegally access sensitive user data [13, 32, 65, 75], eavesdrop on private conversations [32], or even take over the smart speaker and connected smart home devices, including a door lock and the home security system [43].

3 PROBLEM STATEMENT & CHALLENGES

Implementing reliable self-measurements for remote attestation on untrusted and potentially compromised devices involves tackling several challenges, especially on IoT devices that lack hardware security modules like trusted computing components. IoT devices are often integrated into complex ecosystems, where multiple devices are working together, communicating over the Internet, and relying on the vendor's cloud services. This poses challenges towards timing measurements and reliably identifying devices in software-based attestation, i.e., precise timing measurements require uninterrupted and direct communication. An unexpected distortion in communication, e.g., an unexpectedly long delay will make the attestation fail [61]. Often, software-based remote attestation even requires one-hop communication [64]. The lack of reliable device authentication allows attackers to shift attestation tasks to other devices or emulate devices without being detected. To solve this problem, research proposed to restrict communication with other devices, for example by using a dedicated IoT gateway to prevent the attested device to communicate with other, unattested devices [67], thereby detecting all offloading attacks. While this is a solution for managed environments such as companies, this is not a practical solution for home deployments where there are many devices that are not centrally managed, including IoT devices which are often vulnerable [4], and more powerful devices like PCs and smartphones. Furthermore, designing systems to allow end-users to verify a device's integrity, and hence put trust into this particular device, is a difficult task combining technical soundness and usability. In summary, implementing a secure attestation scheme for legacy IoT devices poses the following challenges:

Challenge 1: Secure self-measurement. Secure software-based attestation requires careful implementation. The security of software-based attestation relies on its deterministic minimal runtime. As discussed in Section 2.1, an attacker may not be able to speed up the execution. If an attacker can find any faster implementation of the attestation function, the saved time can be used to compromise the attested device or alter attestation reports.

Challenge 2: Offloading attacks. In software-based attestation, attested devices do not feature a hardware root of trust, but the security solely relies on timing properties. Hence, the verifier cannot securely authenticate the attested device and detect if an attested device is replaced by a different device or a simulation, or if an attestation request is relayed to another instance.

Challenge 3: Precise response times. IoT devices often use indirect communication through cloud services. They do not operate on their own but are integrated into ecosystems consisting of multiple different devices and are closely operating with their vendor's cloud services. Indirect communication is susceptible to relay attacks and increases round trip times and makes response times more fluctuating, complicating software-based attestation.

Challenge 4: Usability. For the user, IoT systems behave like a black box. Cryptographic functions, Internet communication, and attestation protocols in particular, are abstract and unintuitive concepts. Developing user-understandable attestation protocols that users intuitively understand is crucial to gain the user's trust in IoT devices, especially in critical security and privacy domains.

Challenge 5: Legacy devices. There are billions of legacy IoT devices that do not feature trusted computing components. Security solutions for these systems must not require additional hardware extensions, extra sensors, or new communication technologies.

SCAtt-man addresses these challenges by developing a user-observable communication channel to perform software-based attestation. As we will show, this effectively solves the root-of-trust problem in software-based remote attestation and results in a user-comprehensible attestation process.

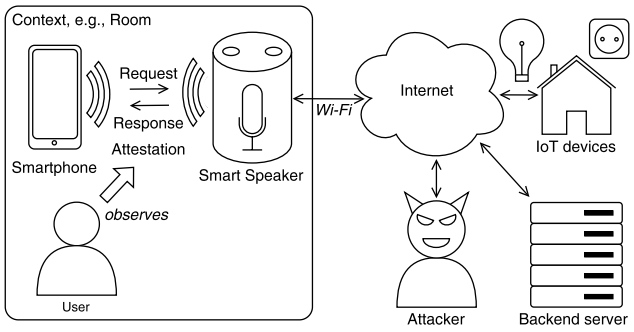

Figure 1: Concept of SCAtt-man attestation: The user can observe the communication between the prover and the verifier.

4 ASSUMPTIONS AND THREAT MODEL

Assumptions. We consider an untrusted IoT device that features a suitable sensor and actuator to be used as a side-channel for communication. In our implementation, we leverage a microphone and a speaker, as they are always available in smart speakers. Note that other combinations are possible as well, e.g., a light sensor or camera and a lamp or display [58]. Furthermore, we assume that the user is close to the IoT device and within reach of this side-channel. The user has a trusted device for verification with suitable sensors and actuators. In our implementation, we use a standard a smartphone with built-in speaker and microphone. We furthermore assume that there are no ambient disturbances. In particular, we assume that the attacker is not in range of the sensors, i.e., we consider a remote attacker. We assume that the Internet connectivity of the attested device can be interrupted, e.g., using a switch. Alternatively, we also propose a solution to limit the Internet connectivity of the attested device using the user's smartphone. To do so, both the smartphone and the IoT device feature a Wi-Fi interface.

Threat model. We assume that a remote attacker can remotely compromise the smart speaker, e.g., alter the configuration or install malware. This can be achieved by means of typical software vulnerabilities, such as memory errors [4], or insecure or insufficiently protected interfaces, a common problem in IoT devices [51]. Similar to all existing software-based attestation proposals, we do not consider physical attacks [61, 63, 67]. This means the attacker cannot alter the hardware of the attested device, e.g., replace or modify it. Furthermore, the attacker is not in range of the sensors used for the communication side-channel, i.e., in the hearing range. We assume the user's smartphone to be trusted.

5 CONCEPT OF SCATT-MAN

The goal of SCAtt-man is to make communication observable by the user, allowing the user to oversee the attestation process and identify the device being attested, thereby solving the inherent problem of missing authentication in software-based remote attestation. We achieve this goal by using a side-channel for the communication between the verifier and the prover. While in traditional attestation schemes communication is assumed to take place via wires or wireless, such as Ethernet or Wi-Fi, we explicitly opt for alternative means of communication. The limited reach of such transmissions reduces possible attacks. Furthermore, in contrast to radio communication, attacks will be noticed by the user. A suitable communication channel fulfills two properties: (1) It is user-observable. (2) It can be sent and received with built-in sensors in both the attested device and the device used for verification, i.e., a standard smartphone. For example, well-suited communication channels are sound and light. A standard smartphone features a microphone and speaker, as well as a camera and a display or an LED used as a flash. Although using side channels for attestation seems like a straightforward concept, the development of a secure software-based attestation scheme using side-channels needs tackling specific challenges such as developing a suitable communication protocol and coping with the manifold attacks on software-based attestation. In the following, we explain how we developed a reliable audio transmission protocol, a secure attestation function, restrict Internet access for the attested device to prevent offloading attacks, and integrated a proof-of-concept of SCAtt-man into a smart speaker. Figure 1 illustrates the concept of a smart speaker enhanced with SCAtt-man attestation. As explained in Section 2.2, smart speakers are a good example to show the applicability of SCAtt-man attestation. First, a smart speaker is a typical and popular type of IoT device. Second, users are particularly concerned about the security of those devices [72, 77]. This is an important aspect, as the user needs to initiate the SCAtt-man attestation manually and observe the attestation process.

5.1 Audio Protocol

The communication between the prover, i.e., the attested smart speaker, and the verifier, in this case the user's smartphone, is performed via audible sound. This makes the attestation process user-observable. The communication consists of digital data encoded into sounds. It is not necessary that the user is able to decode the data, it is sufficient that the user can identify the devices that are communicating. This way, offloading attacks, i.e., when another device responds to the attestation request, are effectively prevented. Furthermore, a remote attacker is not able to influence this local communication. Due to this communication channel, the attestation cannot be run in the background but requires interaction with the user. Users manually run the attestation process. To do so, we provide a smartphone app that guides users through the attestation process. The app explains the necessary steps, runs the attestation, and shows the result. A strict requirement for software-based attestation are predictable execution and transmission times. Hence, the protocol should feature a fixed message length and do not have error-correction codes such as parity bits. Otherwise, an attacker could exploit this circumstance to gain a timing advantage by starting the attestation process before the transmission has been completed by using error correction to complete incomplete transmissions. At the same time, the protocol must have a reliable transmission: the user cannot distinguish between a transmission error and attestation failure on a compromised device. Data-over-sound protocols have many parameters, e.g., frequencies, encoding of bits, and duration of the transmission. We perform a detailed explanation of the implementation of the data-over-sound protocol in Section 6.3. In Section 7 we show how we fine-tuned these transmission parameters to obtain a reliable transmission.

5.2 Attestation Function

The main component of any attestation scheme is the attestation function. The attestation function takes the self-measurement of the attested device which is then transmitted to the verifier. SCAtt-man attests the integrity of software and configuration of the attested device. We restrain SCAtt-man to static attestation due to the infrequent attestation runs: Every attestation run has to be manually initiated by the user. Furthermore, the attestation cannot be performed in the background as it plays audio and fully utilizes computing resources of the attested device, forcing one to pause the speech assistant while the attestation is executed. More sophisticated attestation schemes like runtime attestation [1] and data-flow attestation [2, 17] need to be run in background during normal operation of the attested device, and rely on frequent communication with the verifier during attestation. Running attestation without sending the results to the verifier does not give any security benefit as a compromise will not be detected. Therefore, attestation schemes that attest the program runtime are by design incompatible with SCAtt-man. There are several aspects to be considered when designing an attestation function:

Optimal implementation. The security of software-based attestation solely relies upon the computational capabilities of the attested device and timing thresholds. This induces that an attacker cannot significantly accelerate the execution speed of the attestation function. We solve this problem by using built-in hardware modules to run the attestation function. The hardware-accelerated execution of the attestation function is faster than any software-based implementation on the same device. In case no hardware-based acceleration is available, we use standard and widely-used hashing functions. To prevent acceleration using parallelization, we use hashing functions that use the Merkle–Damgård scheme, which does not allow parallelization [42].

Replay attacks. In replay attacks, the attacker responds to an attestation request with pre-computed or old attestation reports. SCAtt-man prevents such attacks by including a random nonce into the attestation request. This nonce, chosen by the verifier and hence out of the control of the attacker, ensures freshness of the attestation reports.

Empty memory. An attacker can use any memory not covered by the attestation. Therefore, it must be ensured that the attestation function actually covers all executable memory and that the attacker cannot compress any memory to obtain unattested memory which can be used to store malicious code. SCAtt-man addresses this problem by closely monitoring execution times of the attestation function. Deviations of the runtime of the attestation functions due to the on-the-fly-decompression of data, while the attestation is running in parallel, will be detected.

Runtime. Determining the correct runtime of the attestation function is crucial for the security of software-based attestation. Therefore, we designed the SCAtt-man attestation function such that its runtime can be configured by increasing the number of iterations. This feature can be used to obtain a runtime that can be clearly distinguished from compromised ones within the attestation process. In Section 7.1, we investigate in detail the runtime of the attestation function to determine strict thresholds and distinguish between correct and compromised runs of the attestation function.

5.3 Limiting Internet Access

Preventing offloading to an external party is crucial for secure software-based attestation. Offloading means relaying the attestation request sent by the verifier to another device. Because in software-based attestation there is no physical security, there is no way to authenticate the device being attested, besides using response times. The runtime of the attestation function on the attested device is longer than transmission to an external party, like a cloud service or another IoT device, takes. Thus, an external transmission must be prevented. There are two possibilities to achieve this. First, use a hardware-based method that disables communication like a hardware kill switch. In case of a wired connection, it would also be sufficient to unplug the network connection. Alternatively, we propose a software-based solution to limit the Internet connection via Wi-Fi of the attested device.

Hardware kill switch. Integrating a hardware kill switch to deactivate network functionality is an elegant solution as it is a simple, user-understandable concept. Furthermore, the usage of such a button is not limited to SCAtt-man attestation. For instance, there are smartphones for privacy-aware users that feature a hardware kill switch to deactivate Wi-Fi and radio communication [16, 54, 56]. Keep in mind that research found that even encrypted traffic of IoT devices can be used to monitor actions [3]. As discussed earlier, users are privacy-sensitive about devices that process speech and pictures, therefore such a button would be beneficial to give users control over the device. Many smart speakers already feature a button to mute the microphone [40].

Software-based locking. We limit the coommunication of the attested device by configuring the attested device as a Wi-Fi access point. The smartphone then connects to this access point and continuously checks the availability of the attested device to ensure that the attested device cannot connect to a third party. This prevents the attested device from connecting to the original Wi-Fi network. However, some Wi-Fi chips can keep connections to multiple simultaneous connections. For example, the ESP32 supports a combined 'station and access point' mode, which allows the ESP32 to open a Wi-Fi access point while being connected to another Wi-Fi network [26, 45]. But this is limited to the same channel as the radio can only listen to a single channel. Consequently, if the verifier maintains a continuous connection to the attested device on a channel different from the regular Wi-Fi connection (i.e., the connection to the Internet) of the attested device, Internet access of this device is effectively prohibited.

Summing up, the nonces that initiate the attestation, as well as the attestation reports, are transmitted via audio between the attested device and the verifier. The attested device is disconnected from the Internet during attestation to prevent communication with the attacker. The attestation function is designed such that the execution cannot be accelerated without altering the hardware. In Section 6 we show how we combined these techniques into a secure attestation scheme and implemented them into a smart speaker.

5.4 Attestation Without Human Interaction

SCAtt-man was primarily designed to address attestation in IoT home installations allowing end-users to verify the integrity and trustworthiness of their devices. By design, SCAtt-man requires the

user to start and observe the attestation process. But SCAtt-man attestation can also be adapted to work without human interaction. To do so, (1) the manual steps need to be automatized, and (2) the verifier app that currently runs on the user's smartphone needs to be replaced. For example, the button to trigger the attestation could be replaced by a timer automatically starting the attestation. The attestation could be performed either by a dedicated trusted attestation device or another nearby smart speaker. These devices could implement a mutual attestation protocol using our proposed sound side-channel.

6 IMPLEMENTATION

In this section, we describe the implementation of the key components of SCAtt-man. We developed a smart speaker that supports SCAtt-man attestation to verify both the program code as well as configuration data, and an Android application that implements the verifier and guides the user through the attestation process. We explain in detail the attestation functionality and the data-over-sound protocol. Furthermore, we show how the user experiences the attestation process. In Section 7, we verify the attestation functionality and show that SCAtt-man can detect compromises. Furthermore, we evaluate the reliability of the data-over-sound transmissions.

6.1 Smart Speaker

To implement the smart speaker, we used an M5Stack ATOM Echo module[1] that combines the popular ESP32 microcontroller with an integrated microphone, a speaker, a button, and a configurable RGB status LED. The ESP32 microcontroller is deployed in various IoT devices [21], it features a dual-core processor, and a Wi-Fi module [22]. On this platform, using the popular FreeRTOS [29], we integrated basic smart speaker functionality, so recording voice commands, sending them to a cloud service, as well as receiving and playing back the response via the integrated speaker. Note that the usage of FreeRTOS or other operating systems is not mandatory to integrate SCAtt-man, as no complex process structures or scheduling are required. When the button is pressed, the smart speaker records the voice command and directly streams the voice command via an HTTP connection to a speech-to-text cloud service[2]. This service determines the spoken text from the recorded sound and sends it back to the smart speaker. The smart speaker then processes the command. Using the text-to-speech functionality of the IBM Watson Rest-API[3], the voice assistant can convert any text to spoken word. The speech assistant sends a string and receives wave audio, which is played back later. A similar operating mode as in standard commercial voice assistants. The current status of the smart speaker is indicated by the color of the status LED, allowing the user to always check the current execution mode of the smart speaker, for example, recording, speaking, or the current step within the attestation process.

6.2 Attestation Functionality

With a long button press, the smart speaker switches to attestation mode, which is confirmed by a red LED light. In attestation mode, the ESP32 microcontroller switches from WiFi-Station mode to Wi-Fi access point mode. The verifier application then connects to the access point of the ESP32. The connection of the smartphone on a different radio channel, other than the home Wi-Fi network, disables the ESP32's capability to maintain an Internet connection during attestation. Once the smartphone connects to the access point, a sound listener is started, which receives the nonce from the smartphone. The received nonce is then passed to the attestation process, which computes the attestation report. The attestation report is then transmitted back to the smartphone. Lastly, the ESP32 switches back to the WiFi-Station mode and connects to the Internet to resume normal operation. For the attestation process, we use a hardware-accelerated SHA-256 for hashing. SHA-256 is a Merkle–Damgård [42] construction [53], that follows a strictly sequential process: Each hash block is used as input for the subsequent block. Therefore, the hashing process cannot be parallelized. Our attestation covers all code and data sections of the ESP32. The hashing is initialized by the received nonce. Since the RAM of the ESP32 cannot fit the entire code and data space, we split the code data and partitions into blocks of 256 B each. Other block sizes are possible but will lead to different attestation runtimes. We use the hash of each block as input for the next one, thus different block sizes will yield different hashes. We loop the hashing of the entire memory of the ESP32 to achieve a suitable attestation runtime. Finally, the resulting hash is transmitted to the smartphone, and we resume all suspended tasks.

6.3 Data-Over-Sound

As explained in Section 5, we use a short-range side-channel for the attestation process. This allows the user in proximity to the IoT device, i.e., the smart speaker, to perform remote attestation without the risk of a remote attacker hijacking the communication channel. In fact, the attestation can be performed on an IoT device with network connections completely turned off. For example, such a short-range side-channel would be sound. In addition, this side-channel is perceptible by users. Only devices in short physical proximity (e.g., the same room) can interfere with the communication. In order to transmit data between the smartphone application and the smart speaker, we implemented a data-over-sound protocol based on SoniTalk [73, 74]. We introduced the following changes to the SoniTalk protocol to adapt it to the requirements for software-based attestation. (1) We introduced a fixed message length, (2) reduced the number of frequencies to increase reliability, and (3) removed the transmission of inverted message blocks. We have chosen the fixed message length of 32 bit, as it corresponds to the chosen length of the transmitted data (i.e., nonces and hashes). A single message is split into eight blocks ($m = 8$) each with a length of 4 bit ($n = 4$) each. Those n bits are encoded by the presence of a corresponding carrier frequency. We reduced the number of carrier frequencies to four since we implemented the entire attestation process on a low-end device (ATOM Echo) with low-quality sensors (speaker and microphone). A reduced number of carrier frequencies makes the transmission process more robust. This is crucial for software-based attestation, as the user cannot distinguish between a failed attestation due to a transmission error and a real compromise. In order to avoid attestation ahead of time, we completely

[1] https://docs.m5stack.com/en/atom/atomecho
[2] https://fanyi-api.baidu.com/
[3] https://www.ibm.com/cloud/watson-speech-to-text

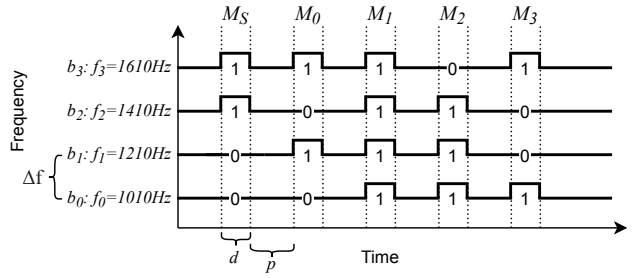

Figure 2: Transmitting 01011111 11101001 via data-over-sound (DOS).

removed the inverted message blocks. Hence, an attacker is required to wait until the data transmission is finished. As our evaluation in Section 7 shows, our chosen protocol parameters allow reliable communication such that the redundant information (i.e., inverted message blocks) is not needed. Each message consists of one start block M_S, followed by m message blocks M. Through an empirical study as shown in Section 7.2, we set the transmission time of a message block to 240 ms. Thus, the transmission of a message takes $9 \cdot 240\,\text{ms} = 2{,}160\,\text{ms}$. The generic data transmission process of our protocol is visualized in Figure 2. We use four carrier frequencies. The presence of a carrier frequency indicates a bit value of 1, and the absence of a frequency indicates a bit value of 0. The message blocks are transmitted in sequence. The message block is always present for the time span of $d = 240\,\text{ms}$. Figure 2 shows also an optional pause p, between message blocks, which has been set to zero in our implementation.

Data-Over-Sound on the ESP32 In contrast to the implementation of a data-over-sound protocol on a smartphone, implementation on a resource-constraint device such as the ESP32 is more challenging. Typical commercial smart-speakers like Alexa[4], Google Nest[5], or Apple Homepod[6] offer more specialized audio hardware and resources. However, by implementing data-over-sound on the ESP32, generic and popular IoT hardware, we show the feasibility of our attestation scheme for a broader range of devices. Sending data via the data-over-sound protocol is straightforward as the ESP32 can encode data on-the-fly. This process can be implemented similarly to the sending process on the smartphone application. However, the receiving component on the ESP32 is challenging due to resource constraints, especially memory constraints on the device. For example, recording and storing large sound blocks is infeasible. Furthermore, the ESP32 provides only a limited number of hardware-accelerated fast Fourier transform (FFT) functions. We solved this challenge by analyzing only the subframes (fraction of 32 ms) of each transmitted tone. We also limited the number of frequencies and separated those frequencies by at least 100 Hz. Therefore, the FTT can detect the active frequencies in each tone more easily. Those optimizations to the SoniTalk protocol enable reliable data transmission over audio on resource-constraint devices such as the ESP32.

[4]https://www.amazon.com/smart-home-devices/b?node=9818047011
[5]https://store.google.com/product/nest_audio
[6]https://www.apple.com/de/homepod-mini/

6.4 Verifier App

We have implemented the verifier as a user-friendly Android application. Since smartphones typically feature a good quality microphone and speaker, they are well suited to communicate with the IoT device over the sound side-channel. Furthermore, it offers a familiar and well-known interface for the user. The verifier application implements a sending and receiving module according to the used data-over-sound protocol. The sending module has two main components: The *Encoder* is responsible for splitting the message (fixed length of 32 bit) into message blocks (each 4 bit). The *ToneGenerator* generates the sounds for each message block. The sound is generated by resampling the active frequencies. Resampling is conducted using a high-resolution sinus lookup table. Afterward, the active frequencies are stacked by adding the sampled values. To avoid clipping in the audio playback, the stacked tone is normalized to a common gain. We based the components for receiving module on the Android Audio Sample Project[7]. The *Recorder* component records and stores sound. The application needs about 60 ms to record, which yields four samples per interval. Each tone is played for 240 ms on the sending side. After recording the audio samples, we perform a fast Fourier transform (FFT) with the three components *AudioCalculator*, *FrequencyCalculator* and *RealDoubleFFT*. The Fast Fourier transform (FFT) yields the frequencies from which the sound is composed. After the frequency decomposition, the *Decoder* can determine the start block and convert each message block to 4 bit integers. In order to properly receive messages over sound, the recording of message blocks must be synchronized with the playback of the message. This is done using the start block M_S.

6.5 Usage Process

To assure that the app is usable for most users, the implementation process followed the guidelines from existing literature [8, 44, 55]. To use the screen space efficiently, the user is guided through the attestation process in several steps. Each step describes a single task in large font size with an additional picture or icon to aid the user's understanding and lower the threshold to use the app. The process is depicted in Figure 3 with screenshots. Furthermore, we provide a video showing the full functionality[8]. To start the attestation, the user opens the attestation app. (1) The app shows a welcome screen explaining the functionality. The user presses 'Start attestation' to start the attestation. (2) The app asks the user to press the button on the smart speaker for 3 s. The status LED on the smart speaker changes to red to indicate the start of the attestation process. Now, the attested device starts an access point. (3) The app asks the user to connect the smartphone with the newly started Wi-Fi access point of the smart speaker. As soon as the smartphone is connected, the status LED of the smart speaker switches to green. In the background, the attestation app now checks the continuous connectivity with the smart speaker. (4) With a click on 'Run attestation' the user starts the attestation process. The smartphone sends the nonce via sound to the attested device. The attested device receives this nonce and then runs the attestation function. The result is sent via sound to the smartphone.

[7]https://github.com/lucns/Android-Audio-Sample
[8]https://youtu.be/HEbm7crMCU8

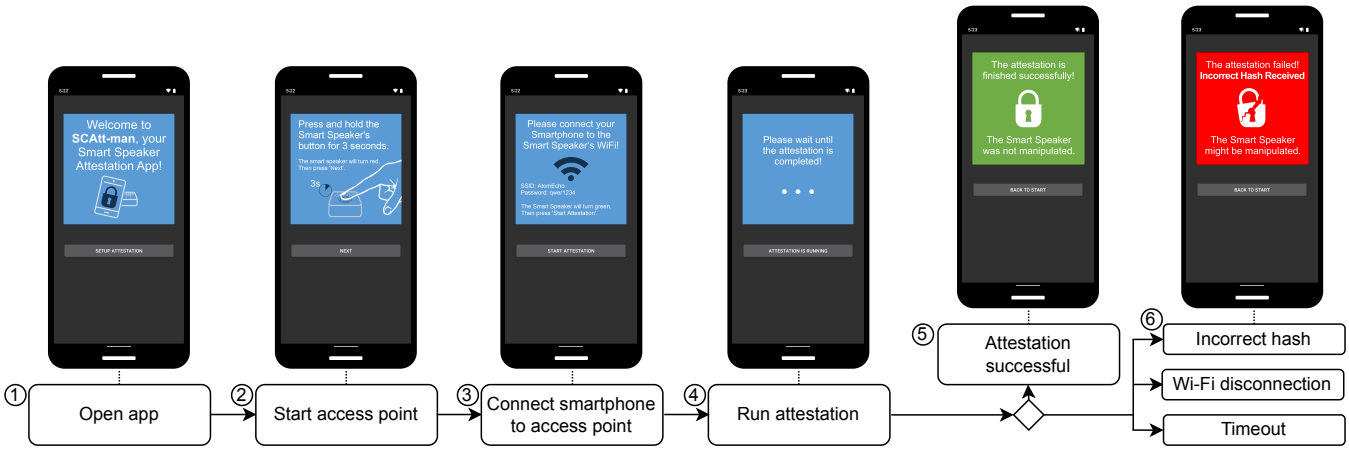

Figure 3: The usage of SCAtt-man. The user is guided through the attestation process. In the end, the smartphone displays whether the attested device was verified successfully or if the attestation failed.

The smartphone app receives this response and then compares this attestation response with the benign state. Furthermore, the app checks the time until the result is received. (5) If the response contains the correct measurements and arrived within the time threshold t_s, then the green screen is shown. (6) If the response took longer than a threshold value t_s, the app shows an error message on a red screen. An error message is also shown if the result of the attestation function does not match the expected value or if the Wi-Fi connection to the smart speaker was interrupted.

If the attestation fails, the user can restart the attestation function. Attestation failure may be caused by transmission error, loss of Wi-Fi-connection, or actual compromise. The error message explains the reason for the failure. Furthermore, since the communication is perceived by the user, the user can detect such distortions and restart the attestation process. In case of transmission errors or connection problems, users just need to repeat the attestation process. However, in our experiments, transmission errors and connection problems rarely occurred as we show in Sections 7.2 and 7.3. Repeated failures indicate a real compromise. The app should then guide the user through a restoration process, for example, by resetting the device to resolve a compromise.

7 EVALUATION

In this section, we show that SCAtt-man is capable of performing a secure and reliable attestation. Furthermore, we explain how we determined the parameters for the data-over-sound protocol. In Section 7.3 we show in a case study how SCAtt-man can detect real-world attacks. We discuss the security of SCAtt-man attestation in detail in Section 7.4.

7.1 Runtime of Attestation Function

The runtime of the attestation function is the main security feature of SCAtt-man attestation, as elaborated in Section 5. We performed a measurement study on our implementation of SCAtt-man on the smart speaker to obtain the runtime of the attestation function. These measurements are crucial in determining the timing thresholds for software-based attestation. The attestation function can be tuned to adapt its runtime to requirements by changing the number of iterations. Each additional iteration increases the runtime of the function. Due to the construction of the attestation function the implementation cannot be parallelized. More details on the implementation of the attestation function can be found in Section 6. We conducted an extensive measurement study on the runtime of the attestation function and report the results in Table 1. We tested one to ten repetitions, yielding an average runtime of 0.81 to 8.1 s; repeating each test 158 times. Our measurement shows that the variance of the runtime of the attestation function is marginal, with a maximum at nine repetitions and a variance of 113 µs. This experiment shows that the attestation function is well-suited for performing software-based attestation.

Table 1: Runtime of attestation function depending on the number of repetitions. All measurements are taken in ms.

Rep.	Min./Max.	Median	Average	Var.	SD
1	812.4/812.9	812.433	812.453	0.006	0.080
2	1624.8/1624.9	1624.927	1624.898	0.003	0.051
3	2437.2/2437.4	2437.372	2437.347	0.005	0.070
4	3249.6/3249.9	3249.769	3249.800	0.016	0.126
5	4062.0/4062.4	4062.152	4062.247	0.023	0.153
6	4874.4/4874.0	4874.559	4874.737	0.058	0.241
7	5686.9/5687.6	5686.980	5687.108	0.045	0.212
8	6499.2/6499.8	6499.398	6499.569	0.066	0.256
9	7311.7/7312.4	7311.824	7312.061	0.113	0.336
10	8124.0/8124.7	8124.249	8124.401	0.074	0.272

7.2 Designing a Reliable Audio Protocol

As alluded to earlier, the communication protocol must allow reliable communication between the prover and the verifier. When the attestation fails, the user cannot determine the cause of this failure: The user cannot distinguish between a failure due to a compromised device and a failure due to a communication error. Thus, it is important to carefully develop the audio protocol such that it allows reliable attestation. The data-over-sound protocol has many parameters that have to be tuned to fit this use case. We conducted an extensive study to determine the optimal values for

these parameters. In particular, these parameters are (1) the block length, i.e., the duration of each tone, (2) the base frequency, i.e., the frequency of each tone, and (3) the frequency separation, i.e., the difference in the frequency between simultaneously transmitted bits. Based on our experiences from the implementation, we limited the state space to a block size of 100–600 ms (steps of 100 ms) and a frequency separation of 100–800 Hz (steps of 100 Hz). We observed that a slight offset of the base frequency by 10 Hz improves the transmission quality mainly because it reduces conflicting overtones. Therefore, we chose a base frequency of 510–2,010 Hz (steps of 250 Hz). In this test, the smart speaker sends random messages to the smartphone, as this is the most tricky part due to the speaker of the ATOM Echo. As receiving device we used a Xiaomi Redmi Note 10. We found that transmissions work best with a block length of 240 ms. We checked the transmission quality between the smartphone and the smart speaker in extensive tests. During our experiments, we found the speaker of the ATOM Echo to be a limiting factor. So we replaced the speaker of the ATOM Echo with a larger model to increase the reliability of the transmission. Note that we did not change any hardware on the microcontroller. The amplifier, microphone, and processor stayed the same. Using the larger speaker, we yielded a success rate of 100 % for the transmission between the smart speaker and the verifier. The following evaluation is conducted using this larger speaker. All of our tests were conducted in a typical office environment. We found that noise disturbances, such as people speaking or traffic noises through the window, did not influence our processes. We attribute this to the use of specific frequencies. We conclude that our attestation method works reliably in typical home and office environments.

7.3 End-to-End Example

We performed a case study to evaluate the full functionality of SCAtt-man. We used a Google Pixel 3 to run the verifier app and the ATOM Echo with the improved speaker. To show how SCAtt-man detects real-world compromises, we integrated a vulnerable web interface into the smart speaker that allows its configuration. This is a typical vulnerability, as the most common weaknesses in IoT devices are weak, guessable, or hard-coded passwords and insecure network interfaces and services [51]. The web interface allows one to change the URL for the speech-to-text service, that is, the service to which the user's voice commands are being sent. This poses a serious risk to the user's security and privacy. An attacker-controlled speech-to-text service allows recording a user's voice commands or arbitrarily altering the commands that the smart speaker executes. This URL is stored in the NVS partition, where all the smart speaker configuration data is kept. As soon as the user starts a speech command, this information is read by the HTTP client that performs the communication with the speech-to-text service. However, as soon as the URL is updated, this changes the content of the NVS partition. Hence, this is detected by the attestation. To check this, after altering the URL via the web interface, we start an attestation run which correctly detects the modifications. To show the reliability of this attestation, we manually repeated the attestation multiple times with the benign and the compromised device. In total, we performed the attestation 50 times on the benign device and ten times on the compromised device. All benign and compromised states were correctly identified. There were no false positives or negatives.

7.4 Security Discussion

A secure software-based attestation scheme requires careful design and implementation. In the following, we discuss typical attacks on software-based attestation and explain how SCAtt-man addresses these.

TOCTOU. Existing attestation schemes are vulnerable to Time-of-Check/Time-of-Use (TOCTOU) attacks [63]. The key idea of this attack is to restore a benign state before the next attestation run, allowing the attacker to stay undetected. However, in SCAtt-man, the attacker cannot predict the attestation. The user randomly triggers attestation runs through physical interaction (for example, a button press). Since the attacker cannot predict the user's attestation request, the attacker cannot restore the benign state in time. Thus, the malicious behavior will be detected over time.

Network delays. For software-based attestation, it is crucial that the transmission time can be predicted precisely [61]. As such, software-based attestation is usually limited to one-hop settings [7]. Short-distance communication over light or sound has a predictable transmission time due to the direct communication from device to device. Users are able to observe the transmission and identify disturbances, e.g., background noise that disturbed the transmission.

Offloading attacks. As communication happens directly over a side channel such as light or sound, there is no need for a remote network connection. In fact, SCAtt-man implements a network disconnection during attestation, effectively preventing any communication to other devices. Consequently, attestation reports cannot be forged.

Compression attacks. A compression attack uses a compression mechanism to free up memory on the attested device, which is not covered by the attestation and may store malicious code [61]. The attacker can perform this attack on-the-fly, compressing and decompressing memory segments on demand to avoid detection. We adopted several techniques from previous work [67]. First, we start the attestation process at a random location, derived from the nonce. Second, we fill the empty memory with random data to make compression itself difficult. Third, writing data on embedded devices has a high latency since write operations need to rewrite entire pages (sections) of memory. Those hardware limitations are described in previous work [67]. Furthermore, the ESP32 and most IoT devices offer hardware acceleration for hashing. Due to the high speed of hashing (using hardware acceleration) and the hardware restrictions on write operations (rewriting flash pages is slow), compression attacks become infeasible for the attacker.

Memory manipulation. In a memory manipulation attack, the attacker uses unmonitored memory such as the RAM to avoid detection. However, due to already existing countermeasures and embedded hardware characteristics, this kind of attack has often become infeasible on embedded devices. For example, RAM is typically marked as non-executable memory [23], so no code can be executed from this memory region. Furthermore, manipulating the memory layout, such as the partition table is highly challenging [25].

Table 2: Responses to the questionnaire on SCAtt-man. Answers range from (1) Completely disagree to (6) Completely agree.

	Question	Avg.	SD
Q1	I used smart speakers before.	2.80	2.11
Q2	I trust IoT devices like smart speakers.	2.85	1.42
Q3	I trust my smartphone.	3.95	1.15
Q4	Additional security measures can increase my trust in IoT devices.	5.40	1.05
Q5	I believe that attestation can detect manipulations in devices.	4.80	1.36
Q6	The audio communication increases my trust in attestation.	4.40	1.10
Q7	When an IoT device has an integrated attestation method I would use this functionality.	5.15	1.50

The ESP32 enforces several preconditions to change the existing memory layout. First, SPI Dangerous Write must be explicitly enabled. Second, such changes are typically implemented by OTA (Over-the-Air Update) [24]. This would require a full reboot, and in addition it would change the stored partition information [25]. Thus, the user-invoked attestation will fail.

Attacks on the attestation mechanism. Since network communication is completely turned off during attestation only direct attacks on the attestation protocol are left. As the attacker can only start attestation once the nonce is received, the critical point is the time span between the reception of the last message containing the nonce and the beginning of the transmission of the response to the verifier. A malicious actor will likely try to speed up the recognition of the last message to gain some time for malicious code execution, as the attestation function by itself cannot be accelerated: The attestation function consists of a non-parallelizable hashing function with an optimized or even hardware-accelerated implementation. In order to avoid speedup of the nonce recognition, we have considered the minimal recording and processing time to receive the complete nonce. Based on those recording and processing times we carefully set the threshold for attestation. As a consequence, there is no usable time gap in the data-over-sound transmission. Thresholds need to be adjusted on a per-device basis, based on the available resources and sound processing capabilities of the device.

7.5 User Study

To ensure that the usability of SCAtt-man is sufficient we conducted a qualitative user study consisting of two parts. First, the users were asked to interact with the SCAtt-man smart speaker and perform the attestation process using a Pixel 3 smartphone. Second, the users completed a set of questionnaires. We recruited 20 participants among company personnel and university students for the study, of which 6 identified as female and 14 as male. The age of the participants ranged from 20 to 65 years with a mean of 37.7. All of them participated voluntarily and no compensation was paid. Each participant was informed about the study objective prior to the study and signed an informed consent that explained which data was collected and how it would be processed. Literature indicates that 20 participants are sufficient to identify at least 95% (mean 98.4%) of all usability problems [27]. Our results did not show a large deviation between users' responses, indicating that saturation was reached. The lack of new input from users signals a sufficient sample size for qualitative studies [11]. Since all participants were recruited among company staff and on the university campus, participants were also tasked with filling out the ATI-Scale [28] questionnaire, in order to make this sample comparable to other studies. The results show that a mean of 4.43 was reached with a standard error of the mean of 1.01 and a Cronbach's alpha of 0.8. To assess the previous knowledge of the users regarding smart home technology, a set of seven questions was included in the questionnaire. Table 2 shows the questions and the answers. Participants were asked to indicate the degree to which they agree/disagree with the statements on a scale ranging from (1) Completely disagree and (6) Completely agree.

The results showed that more than half of the participants had little or no experience with smart speakers (mean 2.8) and that they trusted these devices less than their smartphones (2.85 compared to 3.95). Trust in smart speakers can be enhanced using additional security techniques (5.4). The participants stated that attestation techniques detect manipulations in devices (4.8). In particular, the users think that the observable audio communication further increases the trust in the attestation scheme (4.4). When an IoT device would have an attestation function, the participants would use it (5.15). During the interaction with SCAtt-man the users should speak out their thoughts according to the 'think-aloud' method to identify possible problems in the usage process. Our study demonstrates that the participants considered our app highly usable. However, users tend to always click the button, ignoring the task (i.e., pressing the button on the smart speaker, connecting the Wi-Fi), resulting in a failed attestation. Furthermore, we observed that manually changing the Wi-Fi configuration is tricky due to the many vendor-specific implementations. To further increase the usability of the SCAtt-man app, we plan to integrate checks that prevent continuation before completing the respective task. In order to access the usability of the system, the UEQ-S questionnaire [59] was filled out after the hands-on experiment. Our results show that the usability of the system was rated positively: the overall score was 1.719, the pragmatic quality was rated 1.875, and the hedonic quality with 1.563. Since all three values lie above the threshold of 0.8, the usability of SCAtt-man was evaluated positively. Summing up, we can state that SCAtt-man has both good usability and users trust the attestation. Furthermore, if devices featured an attestation method users would use it.

8 RELATED WORK

In Section 2.1, we discussed approaches for software-based attestation. In this section, we will focus on alternative approaches that use hardware features to obtain a trustworthy self-measurement of an untrusted device. One approach is to use trusted computing components like Intel SGX or ARM TrustZone [6, 15]. These technologies provide a trusted execution environment (TEE), that is separated from the untrusted attested device. Many off-the-shelf processors used in servers, PCs, and smartphones feature such TEEs. C-FLAT uses TrustZone to allow control-flow attestation, thus monitoring the execution of the program flow, detecting deviations like return-oriented programming attacks [1]. OAT introduces the concept of operation execution integrity, covering both control-flow and critical data in the attestation [66]. However, due to cost reasons, such trusted computing components are often not available in low-end and embedded devices. Hybrid attestation schemes schemes rely on a hardware/software co-design that provides hardware extensions

to allow secure remote attestation. These hardware extensions can provide a root of trust, in contrast to pure software-based solutions where such a root of trust is missing. SMART is one of the first hybrid attestation schemes, that uses custom hardware extensions to protect cryptographic keys, while the attestation itself is run in software [20]. A major limitation of attestation schemes such as SMART and SANCUS is that the attestation function cannot be interrupted during execution [20, 46]. TrustLite in contrast enables interrupt handling during the attestation, so that an attestation run is not aborted on interrupt [38]. TyTAN extends this to even allow timing-critical realtime operations [9]. VRASED is a formally verified hybrid attestation scheme [47]. All of these hybrid attestation schemes perform static attestation, i.e., check the memory content of the attested device. Hybrid attestation has also been extended to cover dynamic properties. APEX is able to attest that specific code has been executed [48]. TinyCFA provides control-flow attestation [50]. DIALED extends this work to also monitor data flow, allowing to detect data-only attacks [49]. All these systems are based on the VRASED framework. While these approaches require a few hardware extensions, attestation schemes introducing a more complex hardware design have been also proposed. LO-FAT performs control-flow attestation completely in hardware. This approach does not require instrumentation of the attested software and has less overhead in contrast to other control-flow attestation schemes like C-FLAT [18]. LiteHAX is a hardware-assisted attestation scheme that does not only detect control-flow but also data-only attacks [17].

9 CONCLUSION AND SUMMARY

In this paper we presented SCAtt-man, a solution to perform secure software-based attestation on IoT devices. SCAtt-man solves the inherent problem of missing device authentication in software-based attestation by using user-observable side channels. This approach allows the user to identify the attested device. We implemented SCAtt-man into a smart speaker and developed an app for Android smartphones to perform the attestation. The Android app guides the user through the attestation process. Our evaluation shows that SCAtt-man can reliably perform attestation without failures. In a full end-to-end example, we showed how SCAtt-man can be used to detect a compromise through a typical real-world vulnerability. In a user study, we found not only that SCAtt-man has a good usability, but also that people trust attestation solutions in general and would use them if their devices had such a feature. This makes novel attestation solutions for customer IoT devices a worthwhile research target.

ACKNOWLEDGMENTS

Part of this research was conducted within a student project group at the University of Duisburg-Essen. We thank the participants Roman Heger, Daniel Max, Tai Nguyen Nhan, Niklas Pfützenreuter, Marvin Strauß, and Matteo Viefhaus for their great work.

This work has been partially funded by the Deutsche Forschungsgemeinschaft (DFG, German Research Foundation)—SFB 1119 (CROSSING)—236615297 within project S2 and was supported by the DFG Priority Program SPP 2253 Nano Security (Project RAINCOAT—Number: 440059533).

REFERENCES

[1] Tigist Abera, N. Asokan, Lucas Davi, Jan-Erik Ekberg, Thomas Nyman, Andrew Paverd, Ahmad-Reza Sadeghi, and Gene Tsudik. 2016. C-FLAT: Control-Flow Attestation for Embedded Systems Software. In *ACM Conference on Computer and Communications Security (CCS)*.

[2] Tigist Abera, Raad Bahmani, Ferdinand Brasser, Ahmad Ibrahim, Ahmad-Reza Sadeghi, and Matthias Schunter. 2019. DIAT: Data Integrity Attestation for Resilient Collaboration of Autonomous Systems. In *26th Annual Network and Distributed System Security Symposium (NDSS)*.

[3] Abbas Acar, Hossein Fereidooni, Tigist Abera, Amit Kumar Sikder, Markus Miettinen, Hidayet Aksu, Mauro Conti, Ahmad-Reza Sadeghi, and Selcuk Uluagac. 2020. Peek-a-boo: I see your smart home activities, even encrypted!. In *13th ACM Conference on Security and Privacy in Wireless and Mobile Networks*.

[4] Omar Alrawi, Chaz Lever, Manos Antonakakis, and Fabian Monrose. 2019. SoK: Security Evaluation of Home-Based IoT Deployments. In *IEEE Symposium on Security and Privacy (SP)*.

[5] Omar Alrawi, Charles Lever, Kevin Valakuzhy, Ryan Court, Kevin Z. Snow, Fabian Monrose, and Manos Antonakakis. 2021. The Circle Of Life: A Large-Scale Study of The IoT Malware Lifecycle. In *30th USENIX Security Symposium*.

[6] ARM Limited. 2009. Security Technology Building a Secure System Using Trustzone Technology (White Paper).

[7] Frederik Armknecht, Ahmad-Reza Sadeghi, Steffen Schulz, and Christian Wachsmann. 2013. A security framework for the analysis and design of software attestation. In *ACM Conference on Computer and Communications Security (CCS)*.

[8] Amani Braham, Félix Buendía, Maha Khemaja, and Faiez Gargouri. 2021. User interface design patterns and ontology models for adaptive mobile applications. *Personal and Ubiquitous Computing* (2021).

[9] Ferdinand Brasser, Brahim El Mahjoub, Ahmad-Reza Sadeghi, Christian Wachsmann, and Patrick Koeberl. 2015. TyTAN: Tiny trust anchor for tiny devices. In *52nd Annual Design Automation Conference (DAC)*.

[10] Business Wire. 2021. *Strategy Analytics: Global Smart Speaker Sales Cross 150 Million Units for 2020 Following Robust Q4 Demand*. https://www.businesswire.com/news/home/20210303005852/en/Strategy-Analytics-Global-Smart-Speaker-Sales-Cross-150-Million-Units-for-2020-Following-Robust-Q4-Demand

[11] Kelly Caine. 2016. Local Standards for Sample Size at CHI. In *CHI Conference on Human Factors in Computing Systems*.

[12] Claude Castelluccia, Aurélien Francillon, Daniele Perito, and Claudio Soriente. 2009. On the difficulty of software-based attestation of embedded devices. In *ACM Conference on Computer and Communications Security (CCS)*.

[13] Long Cheng, Christin Wilson, Song Liao, Jeffrey Young, Daniel Dong, and Hongxin Hu. 2020. Dangerous skills got certified: Measuring the trustworthiness of skill certification in voice personal assistant platforms. In *ACM Conference on Computer and Communications Security (CCS)*.

[14] George Coker, Joshua Guttman, Peter Loscocco, Amy Herzog, Jonathan Millen, Brian O'Hanlon, John Ramsdell, Ariel Segall, Justin Sheehy, and Brian Sniffen. 2011. Principles of Remote Attestation. *International Journal of Information Security* 10, 2 (2011).

[15] Victor Costan and Srinivas Devadas. 2016. Intel SGX Explained. *IACR Cryptology ePrint archive* (2016). http://eprint.iacr.org/2016/086

[16] Corbin Davenport. 2020. This smartphone has physical kill switches for its cameras, microphone, data, Bluetooth, and Wi-Fi. Retrieved 2022-03-07 from https://www.androidpolice.com/2020/08/22/this-smartphone-has-physical-kill-switches-for-its-cameras-microphone-data-bluetooth-and-wi-fi/

[17] Ghada Dessouky, Tigist Abera, Ahmad Ibrahim, and Ahmad-Reza Sadeghi. 2018. LiteHAX: Lightweight Hardware-assisted Attestation of Program Execution. In *International Conference on Computer-Aided Design (ICCAD)*.

[18] Ghada Dessouky, Shaza Zeitouni, Thomas Nyman, Andrew Paverd, Lucas Davi, Patrick Koeberl, N Asokan, and Ahmad-Reza Sadeghi. 2017. LO-FAT: Low-Overhead Control Flow ATtestation in Hardware. In *54th Annual Design Automation Conference (DAC)*.

[19] Jide S Edu, Jose M Such, and Guillermo Suarez-Tangil. 2020. Smart Home Personal Assistants: A Security and Privacy Review. *ACM Computing Surveys (CSUR)* 53, 6 (2020).

[20] Karim Eldefrawy, Gene Tsudik, Aurélien Francillon, and Daniele Perito. 2012. SMART: Secure and Minimal Architecture for (Establishing Dynamic) Root of Trust. In *19th Annual Network and Distributed System Security Symposium (NDSS)*.

[21] Espressif Systems. 2018. Espressif Achieves the 100-Million Target for IoT Chip Shipments. Retrieved 2022-03-07 from https://www.espressif.com/en/news/Espressif_Achieves_the_Hundredmillion_Target_for_IoT_Chip_Shipments

[22] Espressif Systems. 2021. ESP32 Series Datasheet. Retrieved 2022-03-07 from https://www.espressif.com/sites/default/files/documentation/esp32_datasheet_en.pdf

[23] Espressif Systems. 2021. Memory Capabilities. Retrieved 2022-03-30 from https://docs.espressif.com/projects/esp-idf/en/latest/esp32/api-reference/system/mem_alloc.html

[24] Espressif Systems. 2021. Over The Air Updates (OTA). Retrieved 2022-03-30

from https://docs.espressif.com/projects/esp-idf/en/latest/esp32/api-reference/system/ota.html
[25] Espressif Systems. 2022. Partition Tables. Retrieved 2022-03-17 from https://docs.espressif.com/projects/esp-idf/en/latest/esp32/api-guides/partition-tables.html
[26] Espressif Systems. 2022. Wi-Fi Driver - ESP32 - ESP-IDF Programming Guide latest documentation. Retrieved 2022-03-16 from https://docs.espressif.com/projects/esp-idf/en/latest/esp32/api-guides/wifi.html
[27] Laura Faulkner. 2003. Beyond the five-user assumption: Benefits of increased sample sizes in usability testing. *Behavior Research Methods, Instruments, & Computers* 35, 3 (2003).
[28] Thomas Franke, Christiane Attig, and Daniel Wessel. 2019. A Personal Resource for Technology Interaction: Development and Validation of the Affinity for Technology Interaction (ATI) Scale. *International Journal of Human–Computer Interaction* 35, 6 (2019).
[29] FreeRTOS. 2022. GitHub - FreeRTOS. Retrieved 2022-03-22 from https://github.com/FreeRTOS/FreeRTOS/tree/master
[30] Gemalto. 2018. The State of IoT Security. Retrieved 2022-03-07 from https://www.infopoint-security.de/media/gemalto-state-of-iot-security-report.pdf
[31] Zhonglei Gu and Yang Liu. 2016. Scalable Group Audio-Based Authentication Scheme for IoT Devices. In *12th International Conference on Computational Intelligence and Security (CIS)*.
[32] Zhixiu Guo, Zijin Lin, Pan Li, and Kai Chen. 2020. SkillExplorer: Understanding the Behavior of Skills in Large Scale. In *29th USENIX Security Symposium*.
[33] Jun Han, Albert Jin Chung, Manal Kumar Sinha, Madhumitha Harishankar, Shijia Pan, Hae Young Noh, Pei Zhang, and Patrick Tague. 2018. Do You Feel What I Hear? Enabling Autonomous IoT Device Pairing Using Different Sensor Types. In *IEEE Symposium on Security and Privacy (SP)*.
[34] Matthew B Hoy. 2018. Alexa, Siri, Cortana, and more: an introduction to voice assistants. *Medical reference services quarterly* 37, 1 (2018).
[35] Chongkyung Kil, Emre C Sezer, Ahmed M Azab, Peng Ning, and Xiaolan Zhang. 2009. Remote attestation to dynamic system properties: Towards providing complete system integrity evidence. In *IEEE/IFIP International Conference on Dependable Systems & Networks (DSN)*.
[36] Bret Kinsella. 2021. Alexa Skill Counts Surpass 80K in US, Spain Adds the Most Skills, New Skill Rate Falls Globally. Retrieved 2022-03-16 from https://voicebot.ai/2021/01/14/alexa-skill-counts-surpass-80k-in-us-spain-adds-the-most-skills-new-skill-introduction-rate-continues-to-fall-across-countries/
[37] Bret Kinsella. 2022. Google Assistant Actions Grew Quickly in Several Languages in 2019, Matched Alexa Growth in English. Retrieved 2022-03-16 from https://voicebot.ai/2020/01/19/google-assistant-actions-grew-quickly-in-several-languages-in-2019-match-alexa-growth-in-english/
[38] Patrick Koeberl, Steffen Schulz, Ahmad-Reza Sadeghi, and Vijay Varadharajan. 2014. TrustLite: A security architecture for tiny embedded devices. In *Ninth European Conference on Computer Systems (EuroSys)*.
[39] Constantinos Kolias, Georgios Kambourakis, Angelos Stavrou, and Jeffrey Voas. 2017. DDoS in the IoT: Mirai and Other Botnets. *Computer* 50, 7 (2017).
[40] Josephine Lau, Benjamin Zimmerman, and Florian Schaub. 2018. Alexa, Are You Listening?: Privacy Perceptions, Concerns and Privacy-seeking Behaviors with Smart Speakers. *ACM on Human-Computer Interaction* 2, CSCW (2018).
[41] Shijia Mei, Zhihong Liu, Yong Zeng, Lin Yang, and Jian Feng Ma. 2019. Listen!: Audio-based Smart IoT Device Pairing Protocol. In *19th International Conference on Communication Technology (ICCT)*.
[42] Ralph Charles Merkle. 1979. *Secrecy, authentication, and public key systems*. Stanford University.
[43] Richard Mitev, Markus Miettinen, and Ahmad-Reza Sadeghi. 2019. Alexa Lied to Me: Skill-based Man-in-the-Middle Attacks on Virtual Assistants. In *ACM Asia Conference on Computer and Communications Security (AsiaCCS)*.
[44] Erik G Nilsson. 2009. Design patterns for user interface for mobile applications. *Advances in engineering software* 40, 12 (2009).
[45] NodeMCU Documentation. 2022. WiFi Module. Retrieved 2022-03-16 from https://nodemcu.readthedocs.io/en/release/modules/wifi/
[46] Job Noorman, Pieter Agten, Wilfried Daniels, Raoul Strackx, Anthony Van Herrewege, Christophe Huygens, Bart Preneel, Ingrid Verbauwhede, and Frank Piessens. 2013. Sancus: Low-cost Trustworthy Extensible Networked Devices with a Zero-software Trusted Computing Base. In *22nd USENIX Security Symposium*.
[47] Ivan De Oliveira Nunes, Karim Eldefrawy, Norrathep Rattanavipanon, Michael Steiner, and Gene Tsudik. 2019. VRASED: A Verified Hardware/Software Co-Design for Remote Attestation. In *28th USENIX Security Symposium*.
[48] Ivan De Oliveira Nunes, Karim Eldefrawy, Norrathep Rattanavipanon, and Gene Tsudik. 2020. APEX: A Verified Architecture for Proofs of Execution on Remote Devices under Full Software Compromise. In *29th USENIX Security Symposium*.
[49] Ivan De Oliveira Nunes, Sashidhar Jakkamsetti, and Gene Tsudik. 2021. DIALED: Data Integrity Attestation for Low-end Embedded Devices. In *58th ACM/IEEE Design Automation Conference (DAC)*.
[50] Ivan De Oliveira Nunes, Sashidhar Jakkamsetti, and Gene Tsudik. 2021. Tiny-CFA: Minimalistic Control-Flow Attestation Using Verified Proofs of Execution. In *Design, Automation & Test in Europe Conference & Exhibition (DATE)*.
[51] OWASP. 2018. Internet of Things (IoT) Top 10 2018. Retrieved 2022-03-07 from https://owasp.org/www-pdf-archive/OWASP-IoT-Top-10-2018-final.pdf
[52] Bryan Parno, Jonathan M McCune, and Adrian Perrig. 2010. Bootstrapping Trust in Commodity Computers. In *IEEE Symposium on Security and Privacy (SP)*.
[53] Wouter Penard and Tim van Werkhoven. 2008. On the secure hash algorithm family. *Cryptography in context* (2008).
[54] Pine Store ltd. 2022. PinePhone. Retrieved 2022-03-07 from https://pine64.com/product-category/pinephone/
[55] Lumpapun Punchoojit and Nuttanont Hongwarittorrn. 2017. Usability studies on mobile user interface design patterns: a systematic literature review. *Advances in Human-Computer Interaction* (2017).
[56] Kyle Rankin. 2019. Lockdown Mode on the Librem 5: Beyond Hardware Kill Switches. Retrieved 2022-03-07 from https://puri.sm/posts/lockdown-mode-on-the-librem-5-beyond-hardware-kill-switches/
[57] Markus Rothmuller and Sam Barker. 2020. IoT the Internet of Transformation 2020. Retrieved 2022-02-23 from https://www.juniperresearch.com/whitepapers/iot-the-internet-of-transformation-2020
[58] Nitesh Saxena, J-E Ekberg, Kari Kostiainen, and N Asokan. 2006. Secure device pairing based on a visual channel. In *IEEE Symposium on Security and Privacy (SP)*.
[59] Martin Schrepp, Andreas Hinderks, and Jörg Thomaschewski. 2017. Design and Evaluation of a Short Version of the User Experience Questionnaire (UEQ-S). *International Journal of Interactive Multimedia and Artificial Intelligence* (2017).
[60] Dominik Schürmann and Stephan Sigg. 2011. Secure Communication Based on Ambient Audio. *IEEE Transactions on Mobile Computing* 12, 2 (2011).
[61] Arvind Seshadri, Adrian Perrig, Leendert Van Doorn, and Pradeep Khosla. 2004. SWATT: Software-based attestation for embedded devices. In *IEEE Symposium on Security and Privacy (SP)*. IEEE. https://doi.org/10.1109/SECPRI.2004.1301329
[62] Smiljanic Stasha. 2021. An In-Depth View into Smart Home Statistics. Retrieved 2022-02-25 from https://policyadvice.net/insurance/insights/smart-home-statistics/
[63] Rodrigo Vieira Steiner and Emil Lupu. 2016. Attestation in Wireless Sensor Networks: A Survey. *ACM Computing Surveys (CSUR)* 49, 3 (2016).
[64] Rodrigo Vieira Steiner and Emil Lupu. 2019. Towards more practical software-based attestation. *Computer Networks* 149 (2019).
[65] Dan Su, Jiqiang Liu, Sencun Zhu, Xiaoyang Wang, and Wei Wang. 2020. "Are you home alone?" "Yes" Disclosing Security and Privacy Vulnerabilities in Alexa Skills. *arXiv preprint arXiv:2010.10788* (2020).
[66] Zhichuang Sun, Bo Feng, Long Lu, and Somesh Jha. 2020. OAT: Attesting Operation Integrity of Embedded Devices. In *IEEE Symposium on Security and Privacy (SP)*.
[67] Sebastian Surminski, Christian Niesler, Ferdinand Brasser, Lucas Davi, and Ahmad-Reza Sadeghi. 2021. RealSWATT: Remote Software-based Attestation for Embedded Devices under Realtime Constraints. In *ACM Conference on Computer and Communications Security (CCS)*.
[68] Trusted Computing Group. 2019. *Trusted Platform Module Library Specification, Family "2.0", Level 00, Revision 01.59 – November 2019.* Retrieved 2022-03-26 from https://trustedcomputinggroup.org/work-groups/trusted-platform-module/
[69] Lionel Sujay Vailshery. 2022. Internet of Things (IoT) total annual revenue worldwide from 2020 to 2030. Retrieved 2023-02-06 from https://www.statista.com/statistics/1194709/iot-revenue-worldwide/
[70] Minhua Wu, Sankaran Panchapagesan, Ming Sun, Jiacheng Gu, Ryan Thomas, Shiv Naga Prasad Vitaladevuni, Bjorn Hoffmeister, and Arindam Mandal. 2018. Monophone-Based Background Modeling for Two-Stage On-Device Wake Word Detection. In *IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*.
[71] Weitao Xu, Chitra Javali, Girish Revadigar, Chengwen Luo, Neil Bergmann, and Wen Hu. 2017. Gait-Key: A Gait-Based Shared Secret Key Generation Protocol for Wearable Devices. *ACM Transactions on Sensor Networks (TOSN)* 13, 1 (2017).
[72] Eric Zeng, Shrirang Mare, and Franziska Roesner. 2017. End User Security and Privacy Concerns with Smart Homes. In *Thirteenth Symposium on Usable Privacy and Security (SOUPS 2017)*.
[73] Matthias Zeppelzauer, Alexis Ringot, and Florian Taurer. 2019. SoniTalk – an open ultrasonic communication protocol. Retrieved 2022-03-07 from https://sonitalk.fhstp.ac.at/
[74] Matthias Zeppelzauer, Alexis Ringot, and Florian Taurer. 2022. SoniTalk. Retrieved 2022-03-07 from https://github.com/fhstp/SoniTalk
[75] Nan Zhang, Xianghang Mi, Xuan Feng, XiaoFeng Wang, Yuan Tian, and Feng Qian. 2019. Dangerous skills: Understanding and mitigating security risks of voice-controlled third-party functions on virtual personal assistant systems. In *IEEE Symposium on Security and Privacy (SP)*.
[76] Shaohu Zhang and Anupam Das. 2021. HandLock: Enabling 2-FA for Smart Home Voice Assistants using Inaudible Acoustic Signal. In *24th International Symposium on Research in Attacks, Intrusions and Defenses (RAID)*.
[77] Serena Zheng, Noah Apthorpe, Marshini Chetty, and Nick Feamster. 2018. User Perceptions of Smart Home IoT Privacy. *ACM on human-computer interaction* 2, CSCW (2018).