DEMO: Secure Bootstrapping of Smart Speakers Using Acoustic Communication

Markus Scheck
Florentin Putz
mscheck@seemoo.tu-darmstadt.de
fputz@seemoo.tu-darmstadt.de
Technical University of Darmstadt
Darmstadt, Germany

Frank Hessel
Hermann Leinweber
fhessel@seemoo.tu-darmstadt.de
hleinweber@seemoo.tu-darmstadt.de
Technical University of Darmstadt
Darmstadt, Germany

Jonatan Crystall
Matthias Hollick
jcrystall@seemoo.tu-darmstadt.de
mhollick@seemoo.tu-darmstadt.de
Technical University of Darmstadt
Darmstadt, Germany

ABSTRACT

Smart speakers are highly privacy-sensitive devices: They are located in our homes and provide an Internet-enabled microphone, making them a prime target for attackers. The pairing between a client device and the speaker must be protected to prohibit adversaries from accessing the device. Most commercial protocols are vulnerable to nearby adversaries as they do not probe for human presence at the speaker or proximity between both devices. In addition to security, the protocol must provide a user-friendly way for initial bootstrapping of the speaker. We design an open pairing protocol for the establishment of a shared secret between both devices using acoustic messaging to guarantee proximity, and release our implementation for the smart speaker as well as Android and Linux clients as open-source software on GitHub.

CCS CONCEPTS

• **Security and privacy** → *Authentication*; • **Computer systems organization** → *Embedded systems*; • **Networks** → **Network protocol design**.

KEYWORDS

Internet of Things, Secure Device Pairing, Device Association, Key Establishment, Key Exchange, Setup, Data over Sound, Authentication

1 INTRODUCTION

Smart speakers are network controlled devices that can play music from any device connected to the user's home network. They also feature voice-controlled virtual assistants backed by an Internet connection and home automation functionality. From a security standpoint, the bootstrapping process of newly bought smart speakers is of particular concern, as this process is responsible for securely setting the speaker up and connecting it to the user's home network. The bootstrapping process establishes the cryptographic keys that protect the secure control channel to the user's client devices (e.g., smartphones or PCs), which makes it a lucrative target for attackers who aim to take control of the smart speaker, monitor the built-in microphones, or gain access to the user's domestic WiFi network.

This demo has been presented at the *16th ACM Conference on Security and Privacy in Wireless and Mobile Networks (WiSec 2023)* in Guildford, Surrey, United Kingdom on 2023-05-29. The WiSec 2023 proceedings contain the list of all presented demos.

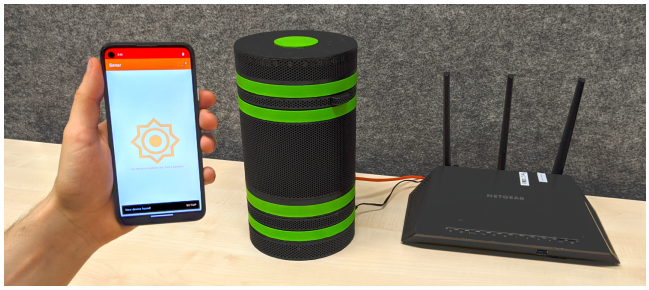

Figure 1: The user can securely setup a smart speaker. The pairing process involves an acoustic out-of-band channel to protect the integrity of the main channel.

As part of the general trend towards the Internet of Things, a wide variety of smart speakers is commercially available from manufacturers such as Google[1] or Amazon[2]. The bootstrapping process of these speakers without pre-shared information, however, is susceptible to remote attackers which try to wirelessly connect to the smart speaker from outside the user's living room. These devices fail to limit new connections to the vicinity of the speaker.

Few vendors, like Sonos[3], employ proprietary acoustic communication to limit the effective pairing range. We expand on this idea and build an open bootstrapping protocol for smart speakers, which improves the security compared to the state of the art by verifying user presence. The user interaction must be carefully designed to avoid rushing behavior where users may skip over security-relevant dialogues [5]. We aim for providing acoustic pairing technology to the community by releasing the source code of our Android and speaker application on GitHub[4]. As smart speakers and smartphones are already equipped with sound transceivers, our implementation is compatible with existing consumer hardware and removes the need for specialized pairing hardware.

2 DESIGN

This section discusses the design of the proposed protocol which establishes a shared secret between client (e.g. a Linux PC or an Android smartphone) and speaker, and provides the speaker with WiFi credentials. After protocol completion, the speaker connects to the user's domestic WiFi network and can communicate with

[1] https://store.google.com/us/product/google_nest_mini
[2] https://press.aboutamazon.com/2015/6/amazon-echo-now-available-to-all-customers
[3] https://www.sonos.com
[4] https://github.com/seemoo-lab/wisec23-speaker-bootstrapping

the client over this network in an encrypted manner. We implement authentication through acoustic communication and by using carefully-designed user involvement that inhibits rushing user behavior, i.e., *select-and-confirm* interaction [8].

2.1 Adversary Model

We assume an adversary who can inject and intercept arbitrary radio and acoustic communication. This includes setting up own APs. We assume that the adversary cannot reside in the same room as the intended user undetected and hence cannot physically access the speaker. We assume the domestic WiFi to be untrusted.

2.2 Protocol Flow

Both devices initially generate a random 128 Bit UUID used for their identification. Figure 2 depicts the three protocol stages.

Secret Establishment. Upon initial powerup, the speaker creates a WPA3-encrypted WiFi AP and repeatedly transmits an acoustic message in the inaudible near-ultrasonic range containing: (1) a randomly generated SSID, (2) a randomly generated PSK, and (3) a hash over the speaker's UUID and public ECDH key. The adversary cannot impersonate the speaker during this transmission as they do not yet know the hash inputs which the client will verify. After the client connects to the AP, both devices exchange their UUIDs and public ECDH information. Both devices derive keys for signing and encrypting further messages from the ECDH master secret.

Mutual Verification. The mutual verification phase is necessary to detect MitM attacks by verifying human presence near the speaker and validating that the exchanged secret matches. A verify key derived from the ECDH master secret is used to pick a word from a word list on both devices. The speaker vocalizes this word among two random options while the client presents it among two different random options (inspired by SafeSlinger [3]). The user is prompted to select the matching word on both the speaker and client.

Setup Phase. The client prompts the user for WiFi credentials and a name for the newly-setup speaker. This information is encrypted and authenticated using the previously derived keys. The speaker connects to the domestic WiFi network and broadcasts its service via *DNS Service Discovery* (DNS-SD) [1].

After initial pairing, subsequent clients can skip the setup phase and discover the speaker via DNS-SD. The recommended request mode requires pairing confirmation from an already paired client to hinder malicious devices in an untrusted domestic network from unnoticed pairing attempts. Users who trust their networks can switch to a more convenient request mode that always allows pairing.

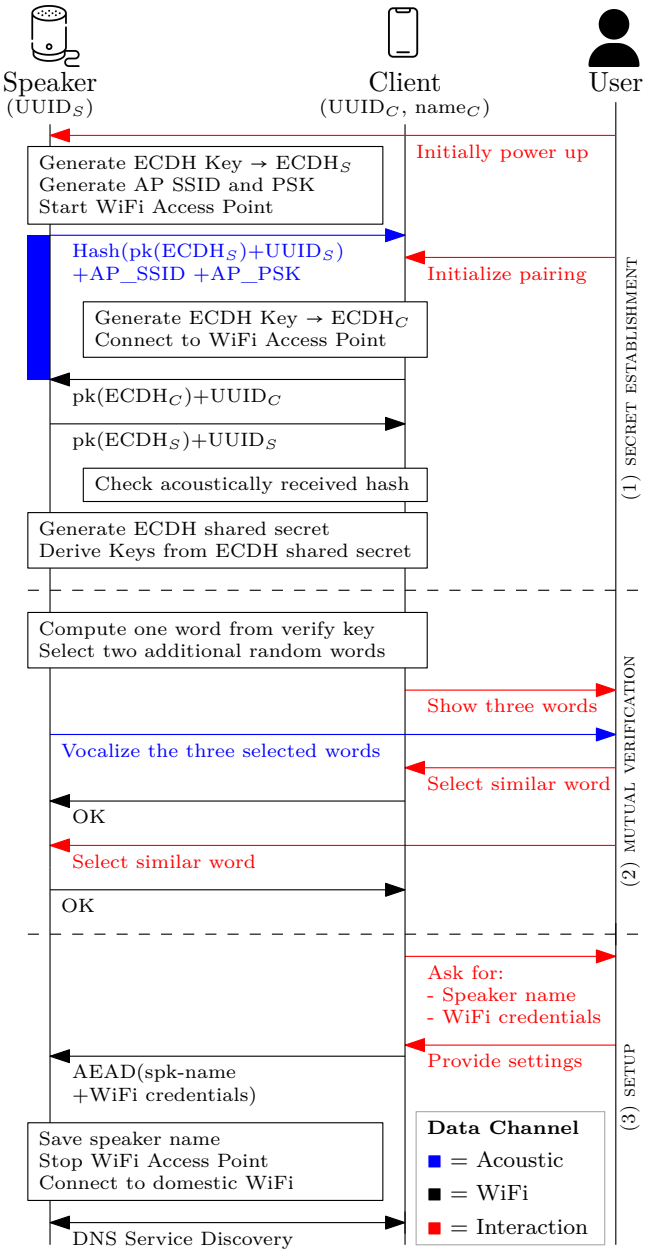

Figure 2: Pairing protocol for initial setup. pk(key) denotes the extraction of the public component from a keypair. MAC(message) denotes a keyed hash. AEAD(message) denotes an authenticated encryption. Hash(message) denotes a hash. All keyed algorithms derive keys from the ECDH shared secret. A bar denotes a recurring transmission until the next message occurs.

3 OPEN SMART SPEAKER PLATFORM

While commercially available smart speakers can provide the necessary hardware for acoustic communication, access to their software is mostly restricted. To facilitate research, our lab currently develops an open, modular smart speaker platform, called *FreeSpeaker*. The platform's core module is a Raspberry Pi Compute Module 4. Specific functionality can be added by stacking "slices". We implemented our pairing protocol on this prototype for evaluation using the five slices: power supply, core module, speaker, microphone, and a lid with a push button. A scenario involving the speaker with clearly visible modules and a client device is shown in Figure 1.

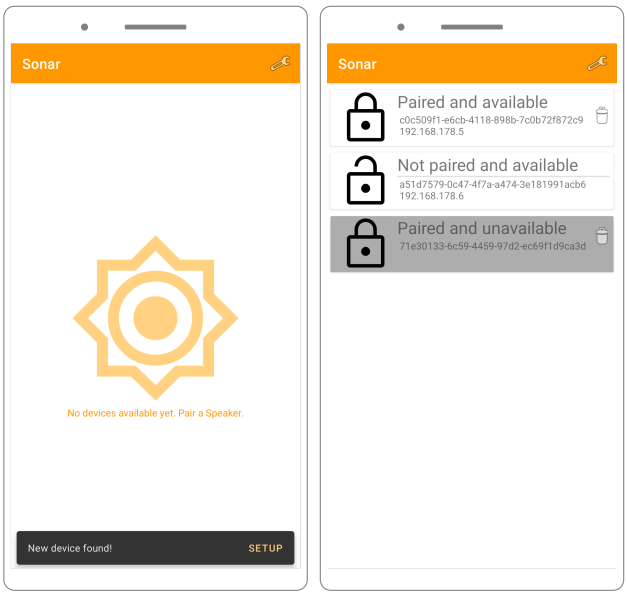

Figure 3: Primary App screen with no paired speakers (left), and with one paired and one unpaired speaker in the current WiFi network and a third unreachable paired speaker (right).

4 BOOTSTRAPPING IMPLEMENTATION

The speaker side was implemented in Python and runs on the *FreeSpeaker*. We implemented clients for Android smartphones (using Kotlin) and PCs (using Python and GTK4 on Linux). Figure 3 shows the Android client, which we will demonstrate at the conference. We use libsodium [2] for cryptography. For vocal text instructions, we pre-generate static parts using the higher quality Mimic-3 [4] and vocalize dynamic text using picoTTS [6].

5 SECURITY EVALUATION

We evaluate the security of our protocol under different threats which we have identified as relevant for smart speakers.

Robustness. We use a second device to transmit white noise. We increase the magnitude of the white noise until the client is unable to receive acoustic messages. Failure occurs at around 11 dB SNR at the client's microphone. Jamming the protocol requires little noise. High directionality of the used speakers poses a usability issue while the effective range of about 1m is a security advantage.

Client impersonation. We assume that the adversary tries to pose as the client, to steal the pairing transaction. They may do this to gain control over the speaker or as part of an MitM attack. The adversary may connect to the setup AP after receiving the acoustic message. The mutual verification will detect the attack: The user must select the correct word on the speaker by matching it with words on their client device. As the user is missing this prompt, the pairing attempt fails.

Speaker impersonation. An adversary may try to impersonate the speaker to retrieve the domestic WiFi PSK and subsequently infiltrate the user's network. The adversary may transmit their own acoustic message to divert the client device to their impersonated speaker. The mutual verification phase detects this attack, as the user must select the matching word at their client. Since the user is missing the speaker's prompt, the pairing will fail. During an MitM attack, it is unlikely that the adversary can get the user's speaker to read out the word matching their pairing attempt as both client and speaker influence the word selection. Consequently, the client will reject the pairing as the user will not select the correct word.

Rushing User. We propose evaluating inputs only after the instructions from the speaker are fully read out. The mutual verification phase inhibits rushing behavior through a *select-and-confirm* design [8] instead of a simple confirmation.

6 CONCLUSION AND FUTURE WORK

Our protocol enhances security compared to current commercial alternatives as it authenticates both the client and the speaker, validates proximity between both devices, and checks for human presence. On the other hand, our protocol adds complexity and time to the pairing process and may be jammed on the acoustic channel. Further security evaluation of the protocol is necessary.

Future work could leverage physical layer security to protect the integrity of messages on the acoustic channel. Combining our protocol with Acoustic Integrity Codes [7] would further strengthen authentication as overshadowing attacks may be detected without user involvement and proximity may be precisely verified.

DEMO

We demonstrate a complete setup (cf. Figure 1) consisting of a WiFi AP, the *FreeSpeaker*, and our software on Android and Linux clients. We present the initial and subsequent pairing processes between those devices and demonstrate that the derived keys can be used to securely control music playback from the client devices on the speaker. Participants are free to try out the pairing themselves.

ACKNOWLEDGMENTS

This work has been funded by the LOEWE initiative (Hesse, Germany) within the emergenCITY center.

REFERENCES

[1] Stuart Cheshire. [n. d.]. *DNS Service Discovery.* http://www.dns-sd.org/
[2] Libsodium contributors. 2023. *Libsodium.* https://github.com/jedisct1/libsodium
[3] Michael Farb, Yue-Hsun Lin, Tiffany Hyun-Jin Kim, Jonathan McCune, and Adrian Perrig. 2013. Safeslinger: easy-to-use and secure public-key exchange. In *Proceedings of the 19th annual international conference on Mobile computing & networking.* 417–428.
[4] Mycroft AI Inc. 2022. *Mimic-3.* https://mycroft-ai.gitbook.io/docs/mycroft-technologies/mimic-tts/mimic-3
[5] Arun Kumar, Nitesh Saxena, Gene Tsudik, and Ersin Uzun. 2009. A comparative study of secure device pairing methods. *Pervasive and Mobile Computing* 5, 6 (2009), 734–749. https://doi.org/10.1016/j.pmcj.2009.07.008 PerCom 2009.
[6] picoTTS contributors. 2021. *picoTTS.* https://github.com/naggety/picotts
[7] Florentin Putz, Flor Álvarez, and Jiska Classen. 2020. Acoustic integrity codes: Secure device pairing using short-range acoustic communication. In *Proceedings of the 13th ACM Conference on Security and Privacy in Wireless and Mobile Networks.* 31–41.
[8] Ersin Uzun, Kristiina Karvonen, and Nadarajah Asokan. 2007. Usability analysis of secure pairing methods. In *International Conference on Financial Cryptography and Data Security.* Springer, 307–324.