Pair-Fi: Integrity Code Protected Secure Device Pairing via SDR-Enabled Wi-Fi Chips on Smartphones

Jakob Link
TU Darmstadt
Darmstadt, Germany
jlink@seemoo.tu-darmstadt.de

Florentin Putz
TU Darmstadt
Darmstadt, Germany
fputz@seemoo.tu-darmstadt.de

Matthias Hollick
TU Darmstadt
Darmstadt, Germany
IMDEA Networks Institute
Madrid, Spain
mhollick@seemoo.tu-darmstadt.de

Abstract

Pairing of wireless devices, such as smartphones, suffers from a plethora of practical challenges if strong security guarantees via integrity codes are desired. Acoustic schemes are easy to deploy, but suffer from limited data rates and high sensitivity to environmental noise. Existing Wi-Fi-based methods, however, lack the necessary timing precision and signal flexibility to implement integrity codes on off-the-shelf devices.

In this paper, we introduce Pair-Fi, a novel method that overcomes these limitations by leveraging integrity codes transmitted using software-defined radio (SDR)-like capabilities enabled through firmware modifications directly on smartphone-integrated Wi-Fi chips. Pair-Fi demonstrates both transmission and reception of raw IQ samples directly on commodity smartphone hardware, allowing for precise on-off keying modulation with timing resolution as low as 4 µs slots. By bypassing the constraints of conventional Wi-Fi frame timings, our approach significantly improves pairing speed, reliability, and resistance to interference. We validate Pair-Fi experimentally on recent smartphone models, such as the Google Pixel 7, showing robust performance in realistic environments. Our results indicate that SDR-enabled integrity code pairing via smartphone Wi-Fi chips provides a practical, secure, and efficient alternative to existing device pairing mechanisms, opening new avenues for secure and seamless device interactions in everyday scenarios.

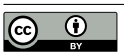
CCS Concepts

• **Security and privacy → Authentication**; **Mobile and wireless security**; **Software reverse engineering**; • **Networks → Network protocol design**; **Wireless local area networks**.

Keywords

Pairing, Smartphone, Integrity Codes, Wi-Fi, COTS, SDR

ACM Reference Format:
Jakob Link, Florentin Putz, and Matthias Hollick. 2026. Pair-Fi: Integrity Code Protected Secure Device Pairing via SDR-Enabled Wi-Fi Chips on Smartphones. In *Proceedings of the 19th ACM Conference on Security and Privacy in Wireless and Mobile Networks (WiSec '26), June 30-July 03, 2026, Saarbrücken, Germany*. ACM, New York, NY, USA, 12 pages. https://doi.org/10.1145/3765613.3811679



WiSec '26, Saarbrücken, Germany

ACM ISBN 979-8-4007-2201-1/2026/06
https://doi.org/10.1145/3765613.3811679

1 Motivation

Recent studies highlight an increased user awareness and concerns about security and privacy in everyday communication scenarios, such as instant messaging and collaboration platforms [34, 35]. Yet, secure channels rely fundamentally on correct device pairing for key exchange, and users cannot benefit from strong encryption if pairing is vulnerable. Secure pairing between previously unknown devices without pre-shared secrets or trusted third parties is crucial to build up underlying secure communication channels. For instance, researchers meeting for the first time at a conference may wish to seamlessly exchange contact information for future collaboration, without fearing that a malicious entity might inject forged data and compromise the resulting end-to-end encrypted communication. Such scenarios emphasize the importance of pairing methods that are both secure and intuitive to use. Research around authentication ceremonies aims to ensure that human participants understand pairing processes and can engage comfortably. However, achieving security and usability simultaneously is challenging. Users commonly experience difficulties with authentication ceremonies due to complexity, misunderstanding, or lack of awareness, causing low adoption and usability barriers [12, 35].

To simplify secure pairing, especially in group settings, cryptographic solutions such as SafeSlinger [11] enable intuitive and scalable public-key exchange on smartphones. While this approach improves usability by abstracting away complex cryptographic operations, it still heavily depends on explicit user actions and attention, potentially limiting its practical usability and security [11, 39].

To further reduce usability overhead and consequently improve security, pairing methods can be extended with out-of-band (OOB) channels, secondary communication channels used to bootstrap or verify cryptographic parameters for an insecure primary channel. Numerous OOB channel technologies have been explored, including visual, acoustic, near-field communication (NFC), and radio-based methods, each offering distinct trade-offs between usability, security, and hardware requirements [27]. Among these, acoustic-based OOB channels have shown particular promise due to their ubiquity on smartphones and ease of interface access. For instance, PairSonic [40] extends SafeSlinger's intuitive group key-exchange paradigm by proposing the use of an acoustic channel, focusing on usability. PairSonic demonstrated high user acceptance in practice but explicitly noted the severe practical limitations of the acoustic channel, notably a high sensitivity to environmental noise, significant propagation delays, and critically low throughput, restricting its practical deployment [39, 40]. PairSonic proposed potential security improvements using advanced physical-layer schemes, such as

acoustic integrity codes [38], which encode authentication information via differential encoding over on-off-keying (OOK) modulation with noise-filled on-slots, enabling robust detection of unintended as well as malicious interference [6–8]. While integrity codes perfectly suit the purpose of authenticating messages at the physical layer for secure pairing, the low throughput and sensitive nature of the acoustic channel still limit its practicality.

In contrast to acoustic methods, radio frequency (RF)-based OOB channels offer potential improvements, including higher data rates, improved reliability, and greater resistance to common environmental disturbances [18, 27]. Prior work shows that integrity code schemes can be realized on stock Wi-Fi hardware, but the use of legacy modulation schemes inherently limits both OOK data rates and signal entropy [18, 28, 38].

Overcoming these limitations requires precise and unpredictable signal modulation beyond the constraints of Wi-Fi standards, which in turn is only possible by achieving fine-grained, software-defined radio (SDR)-like control over the transceiver hardware. While dedicated hardware could be designed for such use-cases, additional hardware contradicts the usability aspect of secure pairing. Alternatively, repurposing hardware built into existing daily-use commercial-off-the-shelf (COTS) devices, like Wi-Fi chips on smartphones, better suits this task. However, such embedded hardware and its accompanying firmware are typically locked down by vendors with little to no public documentation available, making repurposing a significant challenge [41].

Recent research has explored and successfully demonstrated SDR-like arbitrary waveform transmission from raw IQ samples on COTS Wi-Fi chips through firmware modifications on smartphones [41–43]. These demonstrations all employed the open-source Nexmon firmware patching framework [44, 45], a powerful platform specifically designed to modify Broadcom[1] Wi-Fi chip firmware. Nevertheless, all these demonstrations focused exclusively on transmission. To the best of our knowledge, publicly demonstrating the capability to receive waveforms in the form of raw IQ samples directly on Wi-Fi hardware remains an open challenge. Further, the transmission of OOK modulated signals of arbitrary length with short slot-times required for efficient integrity code-based pairing remains a challenging task.

In this paper, we present Pair-Fi, a user-friendly secure group pairing solution leveraging high data-rate integrity codes transmitted and received directly through SDR-enabled Wi-Fi chips embedded in widely-available smartphones, such as the Google Pixel and Samsung Galaxy series.

Summarizing, our main contributions are:

- We design and implement a novel OOB channel, achieving rates of 125 kbps by transmitting OOK waveforms, also known as integrity codes, and receiving raw IQ samples on smartphone Wi-Fi chips. This OOB channel significantly improves throughput over existing methods, enabling fast device pairing.
- We extend the open-source application PairSonic [40] with an interface to our novel OOB secure pairing channel that replaces the current audio channel while adopting most of the app's proven usability. Together with slightly adapted cryptographic processes, this system builds an efficient, scalable, and user-friendly secure device group pairing scheme.
- We present the first public demonstration of SDR-like bidirectional access to commodity Broadcom Wi-Fi chips, enabling both arbitrary waveform transmission and raw IQ sample reception through firmware modifications on modern hardware. This architecture supports the implementation and practical evaluation of novel wireless physical-layer protocols in realistic scenarios, enabling a wide range of advanced applications.
- We experimentally evaluate the security and performance of our OOB channel via simulation and in practice on modern smartphones, considering physical-layer attacks.

To the best of our knowledge, Pair-Fi is the first system to realize smartphone-based RF integrity codes with SDR-like transmission and reception, enabling user-friendly secure device pairing at previously unreached speeds.

By providing our implementation as open-source software to the research community, we ensure replicability and encourage the development of novel applications[2]. With our reverse engineering efforts, we further provide insights on modern IEEE 802.11ax BCM43-based platforms and enable low-level access to proprietary components by contributing to the Nexmon framework [45].

The remainder of this paper is structured as follows. Section 2 introduces the necessary background and related work. Section 3 details the architecture and protocol design of our Pair-Fi system. Section 4 discusses the implementation of Pair-Fi , and Section 5 presents a security analysis. We conclude our work in Section 6.

2 Background and Related Work

This section reviews background and related work in three areas central to our design: secure device pairing (SDP) with a focus on multi-value commitments, group Diffie–Hellman (DH) keys, and integrity codes (Section 2.1), physical-layer signal attacks and defense strategies (Section 2.2), and repurposing commodity Wi-Fi chipsets through low-level access (Section 2.3).

2.1 Secure Device Pairing

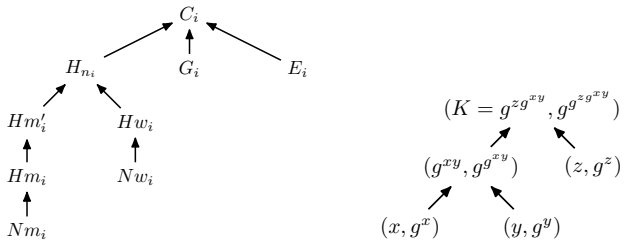
Secure device pairing (SDP) refers to the initial pairing of two or more devices that share no prior security relationship [13]. In wireless settings, this process must account for a strong adversary model in which an attacker can overhear, block, delay, replay, and forge messages [32]. Such capabilities undermine unauthenticated key exchange, making machine-in-the-middle (MitM) attacks trivial [10]. SDP therefore employs OOB channels whose physical or perceptual constraints provide authenticity and integrity, making any tampering detectable [32].

An SDP procedure integrates up to three major components:

(1) an inherently insecure in-band channel used to exchange key-related data,

[1]While the original BCM43 Wi-Fi chipset family design is owned by Broadcom Inc., parts of their technology were sold to Cypress Semiconductor (acquired by Infineon Technologies AG) and Synaptics Inc., resulting in CYW43[48] and SYN43[26] modules that share a common architecture and are equally compatible with Nexmon firmware patches and this project. In this work, we do not further distinguish between different designers, manufacturers, or vendors, and use *Broadcom* or BCM43 as generic terms encompassing all available variants.

[2]The source code of this project is accessible at https://github.com/seemoo-lab/pair-fi.



(a) A commitment derived from hashes of nonces, key material, and user data that can be revealed stepwise for verification and intends.

(b) A group key K is derived from (private key, public key) pairs in a tree layout for three members. Modulo operations omitted.

Figure 1: Exemplary nested multi-value commitment structure and group DH key tree as proposed by [11].

(2) an OOB channel that provides authenticity guarantees, and
(3) user interaction that enables demonstrative identification or supports the correct use of the authenticity mechanism [13].

Together, these components allow devices to establish an authenticated shared key even under strong active attacks [27]. The following subsections discuss each component and give examples of how prior work has realized them in practice.

2.1.1 Insecure In-Band Data Channel. The in-band channel provides efficient data exchange but offers no security. An adversary can perform any action on the channel, making key exchange trivially vulnerable to MitM attacks [10, 32]. But the in-band channel can be used to exchange candidate key material, which is authenticated separately. A common approach that still provides confidentiality and prevents undetected data modification is to use group DH key exchange in combination with commitments. For example, SafeSlinger [11] and PairSonic [39, 40] employ a nested multi-value commitment, as visualized in Figure 1a, followed by a group DH key derivation, as depicted in Figure 1b, to ensure consistent key contribution among participants. Yet, fully secure key agreement cannot be achieved without an additional asymmetric resource [30].

2.1.2 Out-of-band (OOB) Authenticity Channel. OOB channels can provide authenticity and integrity guarantees that in-band channels lack [27]. Such channels exploit physical or perceptual constraints that prevent attackers from modifying transmissions without detection. Integrity codes realize this property by encoding data using OOK, where the presence of energy in on slots cannot be reliably removed by an adversary [6–8]. Combined with Manchester encoding, any attempt to flip a slot breaks the expected transition pattern and becomes detectable at the receiver. Figure 2 illustrates a minimal example of such an integrity coded transmission and an adversarial signal that invalidates it.

Prior work has demonstrated integrity codes in wireless systems. Secure pairing over IEEE 802.11 using energy-based signaling at the physical layer has been shown to work, but with constraints leading to slot durations of tens of microseconds [1, 18]. A similar concept for IEEE 802.15.4 networks comes with slot durations in the hundreds of microseconds, determined by the MAC/PHY standard [47]. In contrast, generating signals from raw IQ samples removes such protocol constraints. Our design uses 4 microsecond

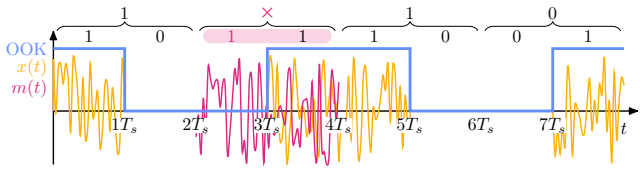


Figure 2: The data `1010` is sent as integrity coded signal $x(t)$. An interfering signal $m(t)$ flips a bit which can be detected.

slots, enabling significantly higher throughput than prior RF-based approaches.

2.1.3 User Interaction. Some SDP protocols incorporate user interaction to bind the pairing process to user intent and confirm that the correct devices participate [13, 32]. However, complex verification procedures increase user errors or reduce usability. Recent systems, therefore, aim to minimize cognitive effort while maintaining reliable confirmation. For example, PairSonic [39, 40] uses acoustic signals to assist users during verification.

2.2 Signal Manipulation Attacks

Integrity codes rely on the physical-layer assumption that an attacker cannot selectively remove energy from a legitimate transmission without detection. However, prior work shows that wireless signals can be manipulated under favorable conditions, challenging this assumption. Beyond simple jamming, adversaries can perform overshadowing, where a stronger crafted signal dominates the receiver, or cancellation/attenuation, where a phase-aligned waveform destructively interferes with the legitimate transmission [37]. Experimental studies demonstrate that such signal cancellation attacks are feasible with SDR platforms [33]. Related investigations show similar manipulation capabilities in wireless ranging and multi-carrier systems, particularly when attackers leverage directional antennas or advantageous geometry [28]. These findings directly undermine the principle of integrity codes, assuming strong attackers that achieve synchronization, have channel knowledge, and a favorable position [49].

Several works propose hardening mechanisms to mitigate such attacks. One direction introduces spatial diversity or cooperation between multiple devices to make simultaneous cancellation significantly harder [14–17, 22, 24]. Another direction attempts to prevent precise cancellation by randomizing the wireless channel properties, limiting an attacker's ability to track the signal [23, 36]. While the latter approach has been shown to be feasible on `BCM43` chips [9], the multi-antenna requirement does not suit our use case. Additionally, helper devices are not user-friendly and therefore not an option for seamless pairing. However, some approaches restrict the attacker model by leveraging propagation constraints, e. g., by defining a secure area based on distance and delay properties [38]. As our pairing system intends positional proximity, we adopt the assumption that an attacker must be outside a safe area to perform secure pairing with Pair-Fi (see Section 5.2 for analysis).

2.3 Repurposing Wi-Fi Chipsets

The first public insights into Broadcom's `BCM43` family came from the `bcm-v4 specification project`, which reconstructed register maps and firmware interfaces for early IEEE 802.11a/g/n chipsets

by analyzing closed-source drivers and interface traffic [31]. First modifications to the programmable state machine (PSM) microcode for the proprietary media access controller (MAC) of the BCM43 chips, named D11 core, were made possible through the publication of the b43-tools, mainly a disassembler and assembler for early revisions of the D11 architecture [5]. With the help of these tools, OpenFWWF developed and shipped a first fully open replacement microcode for these chip generations, demonstrating re-programmability [20].

Today's Wi-Fi chips embedded in smartphones, access points, and mini-PCs, however, use IEEE 802.11n/ac/ax/be variants of the BCM43 family on which a more complex fully equipped system-on-chip (SoC) is implemented, including an ARM core offloading most tasks that were previously performed by host drivers, hiding inner workings behind a closed firmware binary. Inspired by work enabling frame monitoring on such devices [2], Nexmon extended existing toolsets to chip revisions of that time, added a C-based firmware patching framework, and provided helper libraries that make firmware modifications practical on modern devices [41, 44, 45]. Despite that progress, debugging microcode modifications for the opaque D11 core remained tedious. The d11emu framework closed that gap with an emulator that can work closely with running devices for state-extraction and -injection [29].

Modifications to the ARM firmware and the D11 microcode allow development and evaluation of research ideas on COTS Wi-Fi devices. Numerous public applications came out of this opportunity in the past: on the receive side, several projects have revealed hidden measurement capabilities. For example, per-frame channel state information (CSI) computed by the hardware can be extracted and forwarded to user space [19, 21]. While invaluable for sensing, CSI is reported only once per Wi-Fi frame, far too coarse for the microsecond slot timing targeted by our OOB channel. A different line of work samples the receive signal strength hardware, allowing for cross-technology broadcast communication [3, 4] and RF-activity monitoring for testbeds [46]. While this is sufficient for energy detection, the sampler cannot provide the accuracy targeted by our project. On the transmit side, playback of arbitrary waveforms has been shown in practice in two different versions. A small sample play buffer (SPB) can be used to transmit a signal based on a limited number of samples in a loop, suited for simple carrier waves or jamming signals [42]. Alternatively, a larger memory, named buffer memory (BM) or TemplateRAM, can be used to store prepared IQ samples of longer waveforms, for example to implement a covert channel based on pre-coded Wi-Fi frames [43].

The existing ecosystem delivers prepared transmission or coarse reception, but never both in combination and not at the accuracy targeted by this work. Pair-Fi is, to the best of our knowledge, the first open implementation to demonstrate raw IQ sample reception on BCM43 chips and flexible arbitrary waveform transmission on recent IEEE 802.11ax Wi-Fi chips on smartphones.

3 Pair-Fi Group Pairing Protocol

Pair-Fi is an SDP scheme that allows two or more human-operated smartphones to exchange contact data securely without relying on any pre-shared key material or existing public key infrastructure. Pair-Fi uses a pairing protocol that is of the same family as is used by PairSonic [40], adapted to work with an RF OOB channel that uses integrity-coded signals. The scheme combines multi-value commitments, group DH key derivation, and physical-layer security for secure pairing. The protocol ensures that any active MitM attack is detectable and that user data is protected from passive attacks. Ensuring availability or preventing denial-of-service (DoS) attacks is not a goal of the system. Pair-Fi further aims to prevent user errors by minimizing interaction. We assume that all participants are in close proximity and that a potential physical-layer adversary is outside a reasonably small yet sufficiently safe area around all participants. We consider the devices of legitimate participants as secure. As all users should be close by, we further assume that no legitimate participant has malicious intentions.

Section 3.1 gives an overview of Pair-Fis main components and its system structure. It follows a detailed description of the protocol logic in Section 3.2.

3.1 System Structure

Pair-Fis main system components are visualized in Figure 3. The pairing group is split into two role sets: A single user takes the coordinator role (on the left), all other users are regular participants (on the right). On both sides, the Pair-Fi App (A) acts as interface between human operator and smartphone. It further implements the whole pairing protocol logic. The App can interact with a Wi-Fi chip through an application programming interface (API) and data tunnel (B) for two purposes: First, the major part of protocol traffic is exchanged over a high bandwidth channel (C), utilizing Wi-Fi Direct to connect all participants in peer-to-peer (P2P) mode. This channel is assumed insecure by default. Second, for Wi-Fi Direct connectivity bootstrapping, initial configuration, and verification sequence distribution, a dedicated low-bandwidth OOB channel (D) is used. This non-standard broadcast channel uses signals based on integrity codes and is thus assumed to provide integrity verification guarantees under certain conditions. The coordinator can transmit on the OOB channel, and participants can receive. The two channels can operate independently and in parallel. All users are assumed to be in short range over the whole pairing procedure with the possibility to interact with each other, e. g., by speech.

3.2 Group Pairing Protocol Logic

We use the following notation to describe the pairing protocol in detail, where a group of $\mathcal{N}$ users, each user labeled U_i, wants to securely share their data D_i using their smartphones M_i, with $i \in \{1 \ldots \mathcal{N}\}$:

- A and B denote user subsets of U.
- $A \rightarrow B$: A communicates to B over the in-band channel.
- $A \Rightarrow B$: A communicates to B using the OOB channel.
- $U_i \xrightarrow{UI} M_i$: A user interacts with the user interface (UI) of their own device.
- $\circlearrowleft_i$: A device waits for a condition, possibly timing out.
- $\mathcal{H}(\cdot)$ is a cryptographic hash function.
- $\{\cdot\}_k$ is a value encrypted using key k.
- $\|$ is the concatenation operator.
- $=_?$ checks whether two values are equal.

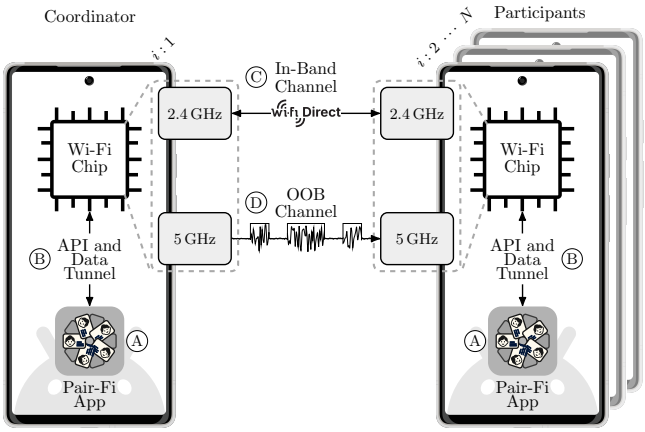


Figure 3: Overview of components involved in a typical pairing scenario using Pair-Fi. A single coordinator smartphone, on the left, and a number of participant smartphones, on the right, are in close range. They communicate over a Wi-Fi Direct in-band channel, while an integrity coded signal is sent from the coordinator to the participants over an out-of-band (OOB) channel that leverages the Wi-Fi chips hardware. An App implementing the pairing logic interfaces with both channels through an API and data tunnel.

The group pairing protocol is built around nested commitments that allow participants to reveal information stepwise, while ensuring that only intended users participate. With C_i, a participant commits to a fresh hash nonce H_{n_i}, a fresh group DH key G_i, and encrypted user data $E_i = \{D_i\}_{N_{m_i}}$. The hash nonce H_{n_i} is constructed based on two randomly chosen nonces N_{m_i} (match nonce) and N_{w_i} (wrong nonce) as follows:

$$\begin{aligned} H_{n_i} &= \mathcal{H}(H'_{m_i} \| H_{w_i}) \\ &= \mathcal{H}(\mathcal{H}(H_{m_i}) \| \mathcal{H}(N_{w_i})) \\ &= \mathcal{H}(\mathcal{H}(\mathcal{H}(N_{m_i})) \| \mathcal{H}(N_{w_i})) \end{aligned}$$

This structure allows each user to publish two unique value pairs that can be verified by other users against the prior received commitments. Publishing the pair (H_{m_i}, H_{w_i}) is used to indicate a successful verification by U_i during the pairing process - (H'_{m_i}, N_{w_i}) indicates failure. Once a pair is received and verified, no other pair from the publisher will be accepted. The match nonce N_{m_i} further serves as encryption key for the user data.

At the very beginning of the pairing procedure, the group votes a single user to take a special role as coordinator ($i = 1$). The coordinator is responsible for kicking off the process that can be separated into the five phases: *Initialization*, *Commitment*, *Distribution*, *Verification*, and *Secret Sharing*, as detailed below.

1) Initialization:

- $U_1 \xrightarrow{UI} M_1$: Select coordinator role and enter number of users $\mathcal{N}$.
- M_1: Spin up Wi-Fi Direct with randomized configuration $\mathcal{C}$, containing network name and passphrase. Choose a random number $\mathcal{R}$.
- $M_1 \Rightarrow *$: Broadcast configuration, number of users, and random number as $(\mathcal{C}, \mathcal{N}, \mathcal{R})$ in an endless loop.
- $M_1 \xrightarrow{UI} U_1$: Display a human-recognizable symbol based on random number $\mathcal{R}$.
- $\circlearrowleft_1$: Wait until $\mathcal{N} - 1$ devices are connected on Wi-Fi Direct with configuration $\mathcal{C}$.
- $U_i \xrightarrow{UI} M_i$, $i \neq 1$: Select participant role and select the same symbol displayed on M_1 from a given list of symbols. The symbol selection prevents listening on the OOB channel before M_1 starts transmitting, which would undermine the channels integrity verification guarantees.
- $M_1 \Rightarrow M_i$, $i \neq 1$: Receive bootstrapping info $\mathcal{C}$, $\mathcal{N}$, and $\mathcal{R}$. Proceed only if integrity is verified.
- M_i, $i \neq 1$: *Optionally compare $\mathcal{R}$ to symbol selection.* Connect to Wi-Fi Direct based on configuration $\mathcal{C}$.

At this point all participants are connected via Wi-Fi Direct and know the number of users, requiring only two UI interactions. The OOB channel stays busy to prevent any late injection of illegitimate signals.

2) Commitment:

- M_i: Randomly choose the two nonces N_{m_i} and N_{w_i}. Derive the nested hashes H_{m_i}, H'_{m_i}, H_{w_i}, and H_{n_i}.
- M_i: Generate group DH key $G_i = g^{n_i} \mod p$ where p is a fixed prime number and n_i a large, randomly chosen number functioning as DH private key.
- M_i: Perform encryption of the user data, $E_i = \{D_i\}_{N_{m_i}}$.
- M_i: Produce the commitment $C_i = \mathcal{H}(N_{m_i} \| G_i \| E_i)$.
- $M_i \rightarrow M_1$: Publish commitment C_i.
- $M_1 \rightarrow M_i$: Distribute all C_j for $j \neq i$.
- $\circlearrowleft_i$: Wait until each user has received $\mathcal{N} - 1$ commitments. Timeout otherwise.

Each participant has committed to their match nonce, DH public key, and encrypted data. All users possess the multi-value commitments of each other.

3) Distribution:

- $M_i \rightarrow M_1$: Publish the committed to values (H_{n_i}, G_i, E_i).
- $M_1 \rightarrow M_i$: Distribute (H_{n_j}, G_j, E_j) for $j \neq i$.
- M_i: Check $C_j =_? \mathcal{H}(H_{n_j} \| G_j \| E_j)$ for all $j \neq i$ to validate the received data. Abort on failure.
- $\circlearrowleft_i$: Wait until each user has received $\mathcal{N} - 1$ value triples. Timeout otherwise.

By now, any participant holds as many bound multi-value commitments and triples of the committed values as the number of participants. From this point on, if a pair (H'_{m_j}, N_{w_j}) for any $j \neq i$ is received by M_i such that $H_{n_j} =_? \mathcal{H}(H'_{m_j} \| \mathcal{H}(N_{w_j}))$, abort the pairing as this indicates verification failure at user j.

4) Verification:

- M_i: Compute verification hash over a concatenation of all $\mathcal{N}$ triples $T_x = (H_{n_x}, G_x, E_x)$ as $h_i = \mathcal{H}(T_a \| T_b \| T_c \| \ldots)$ where the Ts are uniquely ordered by the commitments C_x.
- M_1: Choose a random number $\mathcal{R}_v$.
- $M_1 \Rightarrow *$: Switch OOB channel transmission to verification code and random number tuple $(h_1, \mathcal{R}_v)$.
- $M_1 \xrightarrow{UI} U_1$: Display a human-recognizable symbol based on random number $\mathcal{R}_v$.

- $M_1 \Rightarrow M_i$, $i \neq 1$: Receive $(h_1, \mathcal{R}_v)$. Proceed only if integrity is verified, otherwise publish (H'_{m_i}, N_{w_i}) and abort.
- M_i, $i \neq 1$: Test own verification code against coordinator with $h_i =_? h_1$. If they differ publish (H'_{m_i}, N_{w_i}) and abort.
- $M_i \xrightarrow{UI} U_i$, $i \neq 1$: Display a human-recognizable symbol based on received random number $\mathcal{R}_v$.
- $\circlearrowleft_i$: Wait until all user devices show a symbol. Timeout otherwise.
- $U_i \xrightarrow{UI} M_i$: Confirm if all $\mathcal{N}$ devices display equal symbols. Deny otherwise.
- $M_i \rightarrow *$: On confirmation publish (H_{m_i}, H_{w_i}) to indicate success. Otherwise publish (H'_{m_i}, H_{w_i}) to indicate failure and abort.
- M_i: Verify that $H_{n_j} =_? \mathcal{H}(\mathcal{H}(H_{m_j}) \| H_{w_j})$ for all $j \neq i$. Abort on failure.
- $\circlearrowleft_i$: Wait until verified (H_{m_j}, H_{w_j}) for all $j \neq i$ are received. Timeout otherwise.

During this phase, all participants have verified and confirmed that they hold the exact same legitimate triples T_i for all $i = 1 \ldots \mathcal{N}$ with a single UI interaction. Because all triples, including the own, are used in the verification hash calculation and the integrity of the hash transmitted by the coordinator can be verified, each participant inherently confirms validity of the own published triple received by all other participants as well as validity of all received triples.

5) Secret Sharing:

- M_i: Derive group DH tree with K as the private key of the root node and encrypt nonce $\{N_{m_i}\}_K$.
- $M_i \rightarrow M_1$: Publish the encrypted nonce $\{N_{m_i}\}_K$.
- $M_1 \rightarrow M_i$: Distribute all $\{N_{m_j}\}_K$ for $j \neq i$.
- M_i: Decrypt $\{N_{m_j}\}_K$ and E_j to get user data D_j for all $j \neq i$.
- $U_i \xrightarrow{UI} M_i$: Select datasets to import and exit.

If not aborted, all participants finally hold the decrypted user data of all other participants with the guarantee that no impersonation on protocol level, data alteration, nor information exposure occurred.

4 Implementation on Smartphones

Implementing a functional proof of concept (PoC) of the Pair-Fi SDP scheme on actual hardware requires a platform on which the system components described in Section 3.1 can be realized. Querying the Wi-Fi Alliance database[3] for smartphones that support dual-band Wi-Fi and Wi-Fi Direct returns more than 9000 results. The real number of candidate models is lower, as there is no direct indicator of simultaneous band operation capabilities, which is one of the pairing system requirements. Additionally filtering for Wi-Fi Agile Multiband, a certification tightly coupled to parallel band use, reduces the number of candidates to around 1000. A subset of unknown size of the remaining smartphone models allows modifications of their Wi-Fi chip firmware, which is needed for implementing Pair-Fis OOB channel. We found that most Samsung Galaxy models since the `S8` and all Google Pixel models since the 7 series can work with Pair-Fi in theory. We pick the Google Pixel 7 model as target. It comes with a Broadcom `BCM43` family chipset, namely the `BCM4389c1` [25], which we can work with through reverse engineering efforts and firmware modification.

In the following, details on the implementation are presented in three parts: Section 4.1 describes the realization of the user space App. It follows Section 4.2 with details on the API and data tunnel. The last part, Section 4.3, is dedicated to the OOB channel and related platform insights gained through reverse engineering.

4.1 Application

The application implements UI and pairing protocol logic, and interfaces with the Wi-Fi chip. The UI is realized with the open-source and cross-platform framework Flutter and the Dart programming language. The protocol logic is written in the cross-platform language Kotlin, following the description in Section 3.2. Any system functionalities, such as configuration of Wi-Fi Direct, are done through the Android SDK, targeting Android 14 and newer.

Figure 4 shows an excerpt of the interface visible during a pairing procedure. At first, users can select a role, either as coordinator or participant. The coordinator enters the number of participating users and is forwarded to a widget that shows an icon. In the background, a configured Wi-Fi Direct network is created, and the coordinator stays on this screen until the expected number of participants is connected. In parallel, participants are asked to select the icon displayed on the coordinator's screen. The next visible screen for both parties is an icon that all users have to compare. Only if all users confirm equality of the displayed icon, user data is exchanged and ready for import, as visible on the final widget. If the user rejects the confirmation, message integrity on the OOB channel cannot be verified, or a wrong nonce is received, the app displays a security error warning of a possible attack. If at any of the relevant stages a timeout occurs, a respective message is displayed to the user. Each screen is kept simplistic, paired with a concise instruction to facilitate handling and minimize user errors.

The protocol logic requires two initial nonces, a symmetric crypto algorithm, and commitments derived by a hash function. The application uses AES GCM ECDH with a key size of 256 bit. As one of the nonces serves as encryption key, this also defines the size of the nonces. As hash function for commitment and verification code derivation, the application uses SHA3-512.

4.2 API and Data Tunnel

In order to perform the pairing protocol, the application must interface with the Wi-Fi chip for two purposes: First, to initialize and join a Wi-Fi Direct network. Second, configuring and accessing the OOB channel. While the former is possible through the Android system, the latter requires a custom implementation. On the application side, this can be done as part of the protocol logic. On the Wi-Fi chip, a custom handler must be integrated into the existing firmware. Prior work around the `BCM43` commonly leverages an existing input/output control (IOCTL) call system for that purpose. This method requires only a single function hook in the Wi-Fi chips' ARM microprocessor firmware and is thus simple to implement. However, performing IOCTLs under Android requires elevated privileges that the application does not possess.

Instead, control and data messages can be embedded into broadcast user datagram protocol (UDP) frames. For sending messages

[3] The Wi-Fi Alliance provides a product finder at https://www.wi-fi.org/product-finder.

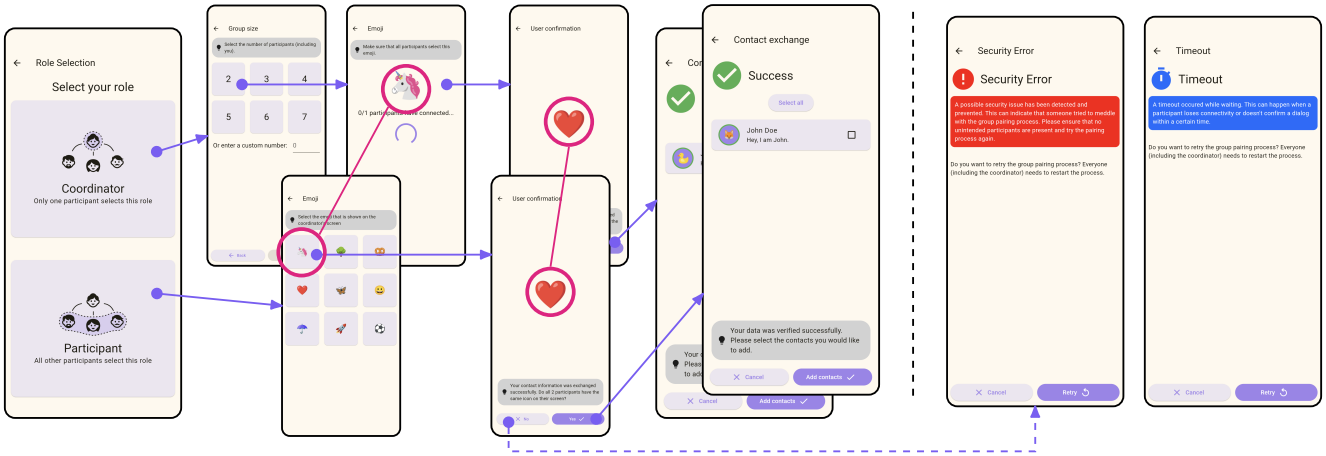


Figure 4: User workflow in the Pair-Fi App during the pairing procedure with two participating users.

from the Wi-Fi chip to the application, respective frames with the generic broadcast UDP address and a configured destination port can be crafted and pushed towards the host. The application can receive the messages by listening for broadcast frames on the configured port at a UDP socket. Sending messages from the application towards the Wi-Fi chip works similarly. The message can be transmitted on a UDP socket, destined to the broadcast address of the network that the Wi-Fi chip is currently configured with. This requires that the interface is configured. For the coordinator, this is true by default due to the Wi-Fi Direct network that is configured at the very beginning of the pairing process. This is not the case for the participants. They join the network after receiving data on the OOB channel. To workaround this, every participant configures its own Wi-Fi Direct interface at the beginning of the pairing process, which gets replaced when joining the coordinator's network.

Intercepting the UDP frames on the Wi-Fi chip requires some additional effort. On recent BCM43 devices, frames are no longer pushed as a whole through the ARM processor's memory, as was the case on older revisions described in prior work. Instead, a reduced header is given to the ARM firmware for inspection, and the actual frame is directly fetched through direct memory access (DMA) by the D11 MAC. Through reverse engineering, we discovered a special packet fetch engine that can be instructed by the ARM processor to pull in the missing frame content. The only thing left after fetching and processing is to free the frames as they are aired out otherwise.

For control and data exchange between the Wi-Fi chip (W) and the application (A) a common API must be defined. The messages might consist of a single identifier or contain structured data in form of type-length-values (TLVs). We define the following API messages, ordered by their type identifier:

- **0x1** $A \rightarrow W$: Start a transmission on the OOB channel. The message is a TLV holding the data to transmit.
- **0x2** $A \rightarrow W$: Stop an ongoing transmission.
- **0x3** $A \rightarrow W$: Start a reception on the OOB channel.
- **0x4** $W \rightarrow A$: Data received on the OOB channel as TLV.
- **0x5** $A \rightleftharpoons W$: Ping, including a sequence number.
- **0x6** $A \rightleftharpoons W$: Pong, including a sequence number.
- **0x7** $A \rightarrow W$: Configuration parameters as nested TLVs.

4.3 OOB Channel

Pair-Fi relies on the physical-layer security of an OOB channel, more specifically on verifiable integrity, during its bootstrapping phase and for distribution of a verification hash. In theory, integrity codes can be used for that purpose. But transmitting and receiving integrity-coded signals with decent entropy and short slot times in practice requires low-level access to transceiver hardware. Although there is barely any public technical information available on the targeted Wi-Fi chip, we discovered that it is possible to instantiate the chip's hardware through software changes, such that it enables transmission and reception of integrity codes. An introduction to the complex chip system is given in Section 4.3.1. While transmission of custom signals on the BCM43 chip family has been demonstrated [42, 43], it is still challenging to continuously transmit OOK with short slots and random samples. Section 4.3.2 describes extensions to prior transmission methods that enable fast and reliable integrity code transmissions. Theoretically, a receiver can decode integrity codes purely from power measurements, but for the targeted short OOK slots, this would require fine-grained values. Instead of relying on power estimates, we demonstrate that it is possible to capture pure signals as IQ samples on BCM43 chips. Section 4.3.3 provides insights on this feature and showcases it as receiver side of the OOB channel.

4.3.1 System Overview. The overall layout of BCM43 family SoCs has been analyzed and discussed in prior work. However, recent chips seem to differ in their modular architecture. An abstract representation with a focus on relevant components of the system layout of recent BCM43 chips is given by Figure 5. A single SoC is connected to a host system over a bus, e. g., PCIe. The host side is managed by a driver, e. g., bcmdhd. On the Wi-Fi chip, there are five main components: An ARM Cortex R4 or A7 with on-chip ROM and RAM, responsible for initialization and handling of non-time-critical tasks. The firmware for this processor is loaded by the driver from the host file system on bring-up. This firmware can be modified by writing C-based patches in combination with the Nexmon framework. The ARM processor is attached to an advanced extensible interface (AXI) backplane, which i. a. connects it to the D11 MAC, a proprietary microcontroller responsible for handling time-critical tasks. The

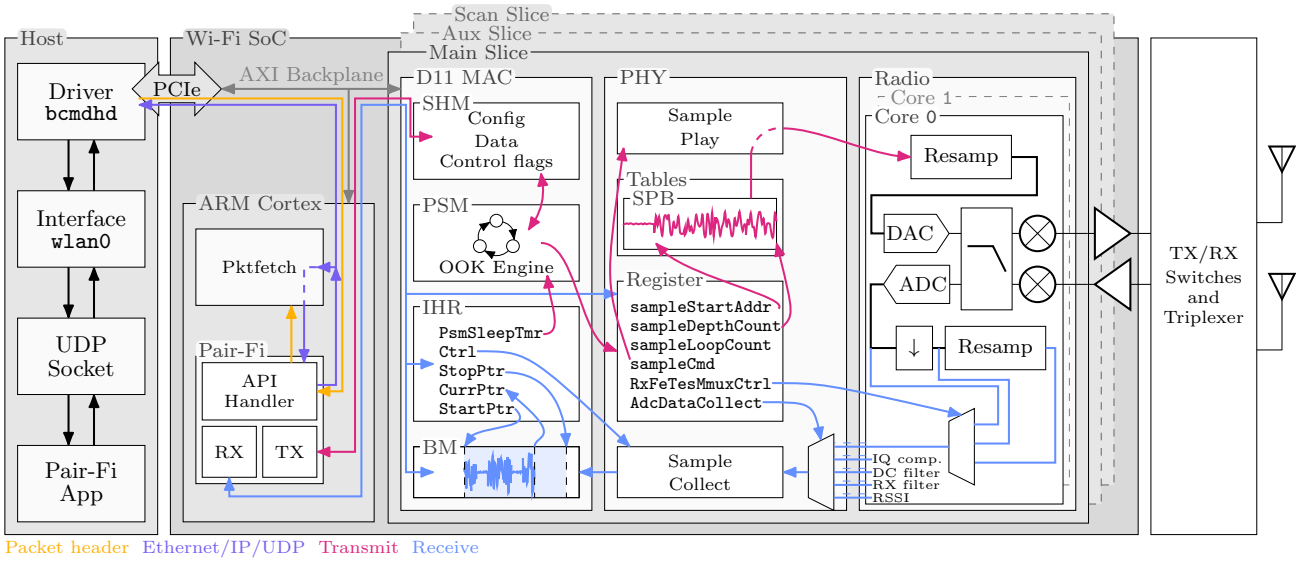


Figure 5: Pair-Fi system overview, layout, and integration into the architecture of recent BCM43 chips.

D11 runs a PSM optimized for flow control. The microcode running in this PSM is embedded in the firmware file of the ARM processor and can also be modified. However, the proprietary architecture makes this a challenging task. The D11 comes with multiple own memory areas, register banks, and hardware modules. It offers an interface through memory-mapped input/output (MMIO) registers. The interface includes pass-through access to components of the PHY block. The PHY is responsible for all the IEEE 802.11 physical-layer tasks, such as (de)modulation, equalization, beamforming, or frequency-time-domain conversions. It comes with its own registers and buffers (or tables) that control functionality, store data, or report statuses. However, they are mostly split up into bit fields with unknown meaning, making it hard to work with. The PHY is directly connected to the Radio, which handles digital-analogue-conversion, filtering, and mixing. Between Radio and antennas, an RF front-end (FE) provides amplification, transmit/receive switching, and antenna multiplexing.

A major difference from prior versions of the BCM43 chip layout is the redundancy of the D11, PHY, and Radio. While older models were able to tune in either the 2.4 GHz or 5 GHz band at a time, the newer architecture provides parallel operation in both bands — or even three bands since the introduction of 6 GHz for Wi-Fi. This feature is advertised as real simultaneous dual-band (RSDB) and "tri-band simultaneous" [25]. The redundant portions are called slices, where the main slice operates in the 5 GHz and 6 GHz bands, while the aux slice works in the 2.4 GHz band. On chips advertised as tri-band simultaneous, a third scan slice exists that can operate in all three bands, but seems to be 20 MHz bandwidth only. All three slices share the same antennas through di- or triplexers. This newer architecture is particularly suited for Pair-Fi as it allows for operating a fully functional Wi-Fi Direct on one slice, while running the custom OOB channel on another slice.

4.3.2 Transmitter. Prior work [42, 43] has demonstrated that transmission of custom signals from IQ samples can be realized on BCM43 hardware in two ways: First, a PHY table named sample play buffer (SPB) that can hold up to 512 samples can be used. Second, IQ samples can be provided from a larger buffer memory (BM), which on most platforms is part of the D11. The buffer size is platform-dependent and can vary from a few hundred kibibyte up to one mebibyte. The sampling rate is twice the bandwidth of the currently selected Wi-Fi channel. Both approaches allow to configure start and end pointers, and provide looping indefinitely or a specified number of times. We were able to verify that both functionalities still exist in modern BCM43 chips by reverse engineering methods related to self-calibration, such as IQ imbalance or power amplifier pre-distortion (PAPD), and a custom time-of-flight (ToF) implementation. In theory, both approaches sound promising for implementing the transmitter side of Pair-Fis OOB channel. A trivial approach would be to fill either buffer with random samples and then toggle transmission on and off according to the desired OOK slots. However, the turnaround times between on and off toggling are unpredictable and can have durations of hundreds of microseconds. For transmission from the BM, it would also be feasible to prepare the whole OOK modulated signal at once instead. But writing to the BM through D11 MMIO registers is slow compared to the rate at which the samples are fetched from the transmit hardware. Hence, rewriting random samples during transmission is not feasible and would lead to a repeating signal. This makes it trivial for an adversary to learn the signal and attack it on the physical layer. For the SPB, we discovered through experimental evaluation that start pointer (sampleStartAddr) and number of samples to transmit (sampleDepthCount) can be dynamically rewritten with very low latency during active transmission. This can be leveraged to transmit arbitrarily long OOK signals by dynamically switching between low amplitude or zero samples for off slots and random samples for on slots. However, the available number of samples is limited, reducing entropy of on slots. We workaround this by choosing a random offset inside the stored on slot samples for each slot. While a proper live source for random samples would be favorable, this approach still provides practically unpredictable on slots, which allows us to transmit integrity-coded signals.

A next challenge lies in the actual setting of the respective registers. To realize short slot durations, precise timing and low latency are required. Further, running either a long task or multiple scheduled tasks on the single-threaded ARM processor interferes with Wi-Fi Direct operations and is prone to unpredictable delays. Instead, running the on-off-switching as a dedicated state machine on one of the D11 cores solves these issues. Further, we discovered that the proprietary architecture supports a sleep instruction that can be configured in $^1/_8$ µs steps through an internal hardware register (IHR) (PsmSleepTmr), which is very precise compared to scheduling on the ARM (~1 ms). All initialization can be comfortably performed through the ARM core. This includes the conversion of the data bits into Manchester coded bits through a lookup table. These bits, the message lengths, as well as control signals, can be given from the ARM processor to the D11 through its shared memory. The ARM is also responsible for creating and writing the zero and random samples into the SPB. The hardware expects 32-bit wide samples from the buffer with 10-bit signed values for each I and Q part, where the lowest 10 bits are Q and the 10 bits above are I.

Another aspect of the transmission that must be considered is its power. The transmit power depends on the combination of multiple components of the Radio and FE, such as baseband multiplication and power amplifiers. Instead of configuring each component on its own, we can reuse an existing functionality that takes in an index and sets all relevant components at once. In regards to counter attacks on the physical layer, it seems beneficial to transmit with maximum power. Further, having a fixed transmission power makes the decoding on the receiver side easier. We therefore choose the maximum possible power by default.

As already discovered in prior work [43], it can happen that instead of the desired signal, only an unmodulated carrier wave is transmitted. Although Pair-Fi is a PoC, it is desirable to have a reliable system. We therefore include a mechanism that reads back the current transmit signal strength indicator (TSSI) and decides based on a threshold if the current transmission is the actual signal or carrier only. In the latter case, the transmission is restarted until the desired signal is aired or a configurable number of retries is reached. With a maximum retry count of 10, we reached a 100% success rate over 100 consecutive tries to kick off a transmission.

4.3.3 Receiver. During reverse engineering of numerous firmware blobs and through extensive practical evaluations, we discovered that BCM43 hardware is capable of capturing sampled data from several tapping points into the BM of the D11. Figure 5 shows an excerpt of this functionality, also called Sample Collect, in an abstract manner. Selecting a tap is mainly done through two multiplexers, which are controlled through PHY registers. The RxFeTesMmuxCtrl register can tap on early stages such as analog-to-digital converter (ADC), cascaded integrator–comb (CIC) decimator, or Farrow resampler outputs. While such values support calibration and debugging, they might come at a high sampling rate, and processing them in software does not scale well. This is different with the taps selectable by a bitfield of the AdcDataCollect register, which are values coming out of an IQ mismatch compensator, DC filter, RX filter, or received signal strength indicator (RSSI) measure. The sampling rates here are either double the current Wi-Fi channel bandwidth, e. g. 40 Msps at 20 MHz, or even equal to it. This is beneficial as reading from the BM is a critical bottleneck. On devices where the BM is part of the D11, accessing it from the ARM is only possible by reading four bytes at a time through an MMIO register. This also prevents streaming samples to the host. Thus, the sample collection works rather as a one-shot capture. The behavior of the sample collector can further be controlled through some D11 registers: A start and a stop pointer into the BM can be set. A control register allows to select if a single or continuous capture shall be performed. The latter works as the selected area in the BM functions as a ring buffer. Another register reports a pointer to the most recently written sample. All three available pointers expect indices representing four bytes each as input. It is also possible to sample interleaved from multiple radio cores or from a single core.

As samples can not be streamed out of the BM as fast as they are written, the size of the BM dictates the amount of data and thus the duration over which a signal can be captured. On the BCM4389, the BM is 384 KiB large. Assuming that the output of the RX filter is selected, the minimum sample rate by default is 20 MHz, and each sample is four bytes long, holding I and Q as signed two-byte long values. This allows to collect samples for 4.9152 ms. Pair-Fi sends at most 32 bytes over the OOB channel. Each bit in a message is Manchester encoded and thus takes two slots. Based on this, we choose a slot duration of 4 µs, which lets an OOB message fit twice in the BM, ensuring that no unwanted cutting occurs.

Once an OOB channel signal is captured, the contained message can be decoded in software on the ARM. The receiver first generates an envelope — already during readout for higher efficiency and lower memory footprint. Next, a configurable threshold is applied to get clear on and off slots. Start and end of a message are detected through cross correlation with a known integrity code preamble. Then, each slot is consecutively processed by selecting a possible window and checking for the expected Manchester transition. The self-clocking property of Manchester encoding allows for continuous synchronization. If the whole message can be correctly decoded, its integrity is inherently verified. Suppose any invalid slot is detected, the decoder aborts and signals an error to the host. Else, the decoded message is forwarded to the Pair-Fi application.

As the decoder requires a fixed threshold, preferably as close to the noise floor as possible, we further fix the receive gain to the maximum possible value. This prevents any uncertainties regarding signal power introduced by automatic gain control (AGC) and maximizes the peak signal-to-noise ratio (PSNR). As there is no need to demodulate any exact amplitude or phase, it is also not severe if the high gain leads to clipping every now and then.

5 Security Analysis

Pair-Fi builds on multiple established components that were shown to be secure in prior work. However, the mixture of these and our additions requires a fresh review. Section 5.1 discusses respective changes and security considerations. The pairing protocol relies on the integrity verification guarantee of the OOB channel. Although integrity codes are appropriate for this task, the potential for a sophisticated attack remains a concern. Section 5.2 elaborates on the feasibility of attacking the physical-layer security of Pair-Fi through practical evaluation.

5.1 Group Pairing Protocol

The pairing protocol relies on multi-value commitments, group DH key derivation, and hash verification. SafeSlinger [11] and PairSonic [39, 40] have extensively studied the security of SDP using these mechanisms. They showed that multi-value commitments and encryption prevent information leakage while offering contact authentication. Because the protocol relies on the actual number of users, it prevents the introduction of unauthorized 'virtual' participants. Replacing parties or performing MitM fails due to the hash verification.

However, we must consider that Pair-Fi uses an OOB channel for bootstrapping a pairing session and must meet certain conditions to prevent impersonation of the coordinator. An adversary trying to inject its own bootstrapping information will be detected because the coordinator starts transmitting integrity codes before the participants start receiving. This is enforced through the user interaction via joint icon selection. Thus, their signals would overlap. If the signals are equal, this has no further implications. If they differ, the decoder detects an invalid message and cancels the pairing. The same applies to the verification hash that the coordinator sends later on the OOB channel. To reduce user interaction, the coordinator never stops transmitting between the bootstrapping and verification hash message, inherently providing integrity and weak authenticity of the verification hash if its physical-layer integrity can be verified. Under the assumption that the guarantee of integrity codes holds, this means that the protocol modifications added security while reducing user interaction.

5.2 Physical-Layer Security

Integrity codes have been shown to hold an integrity verification guarantee under the condition that the energy of the on slots can not be canceled [6–8]. While randomness in the generated signal suggests that cancellation is not possible, it has been shown that a powerful attacker might cancel out parts of an unknown signal just through relaying [33, 37, 49]. We have to consider that such strong attackers might compromise Pair-Fis pairing scheme, as breaking the physical-layer security of integrity codes inherently destroys the pairing protocols security.

We harden our system against this attacker model. Every symbol with a duration of 50 ns ($\frac{1}{20\,\text{MHz}}$) is completely random and independent of previous or following symbols. Further, as we can transmit from raw IQ samples, these symbols do not follow any well-defined modulation. And, although we use Wi-Fi hardware, the samples are not mapped to orthogonally spaced carriers, resulting in a high-power noise-like wideband signal. This makes it practically infeasible to predict the remainder of a symbol. Nevertheless, we consider a worst-case adversary that has perfect knowledge of the channel between any two parties, can predict the remainder of a symbol during reception, and has zero processing delay. Similar to acoustic integrity codes, we can assume for our pairing scenario that an adversary is outside a safe area defined by the distance between legitimate sender and receiver $\overline{AB}$ and a safe radius r around either party [38]. This sets a lower bound on the amount an attacker can cancel out per symbol, given the propagation delay. The respective delay computes as $\tau = \frac{2r-\overline{AB}}{c}$. Simulating cancellation on actual Pair-Fi signals sent from a Google Pixel 7 and received

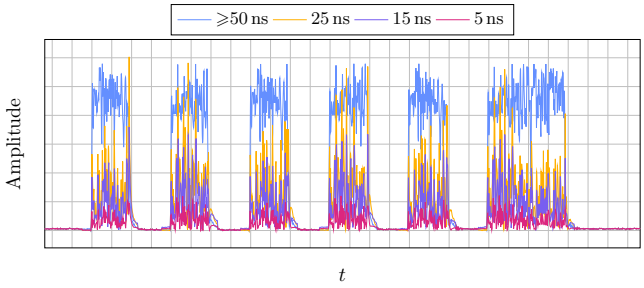


Figure 6: Overlay of a signal sent by Pair-Fi and versions of the same signal with simulated attenuation at three different propagation delays. Even signals under strong attenuation by a nearby attacker are still recognizable.

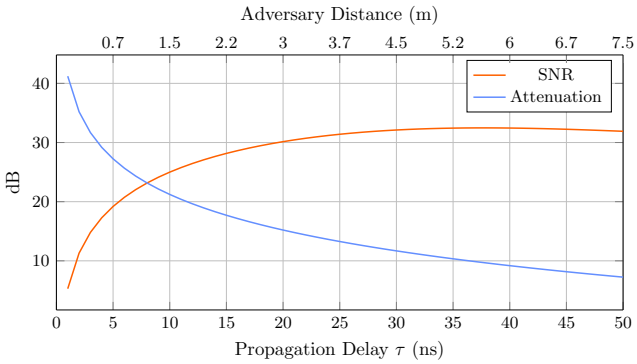


Figure 7: Attenuation and residual SNR of a signal sent by Pair-Fi against distance and inherent delay of a simulated relay attacker to sender and targeted receiver.

on a USRP SDR shows that our system can reliably work even under worst-case conditions. Figure 6 shows that due to the high signal-to-noise ratio (SNR) of our signals, even a strong attenuation leaves on slots recognizable. Figure 7 details the effect of distance and hence delay on attenuation and residual SNR, showing a severe effect yet indicating that full signal cancellation with the given parameters seems not feasible in practice. Even an attacker as close as 1 m could not fully cancel out the signal and thus modify the integrity-coded message.

6 Conclusion

In this paper, we demonstrate the feasibility of secure and efficient device pairing using smartphone Wi-Fi chips augmented with SDR-like capabilities. Our solution, Pair-Fi, overcomes the limitations of existing approaches by achieving unprecedented timing precision and flexibility. With this, it enables robust pairing performance even in challenging real-world environments, thus allowing for practical and user-friendly future wireless pairing solutions.

Acknowledgments

This work was supported by the Federal Ministry of Education and Research of Germany in the project Open6GHub (grant number: 16KISK014) and the LOEWE initiative (Hesse, Germany) within the emergenCITY center [LOEWE/1/12/519/03/05.001(0016)/72].

References

[1] 1999. IEEE Standard for Telecommunications and Information Exchange Between Systems - LAN/MAN Specific Requirements - Part 11: Wireless Medium Access Control (MAC) and Physical Layer (PHY) Specifications: High Speed Physical Layer in the 5 GHz Band. *IEEE Std 802.11a-1999* (Dec. 1999), 1–102. doi:10.1109/IEEESTD.1999.90606

[2] Andres Blanco and Matias Eissler. 2012. One Firmware to Monitor Em All. https://www.coresecurity.com/core-labs/publications/one-firmware-monitor-em-all

[3] Hannah Brunner, Rainer Hofmann, Markus Schuss, Jakob Link, Matthias Hollick, Carlo Alberto Boano, and Kay Römer. 2020. Cross-Technology Broadcast Communication between Off-The-Shelf Wi-Fi, BLE, and IEEE 802.15.4 Devices. In *International Conference on Embedded Wireless Systems and Networks (EWSN 2020)*. ACM, Lyon, France, 176–177.

[4] Hannah Brunner, Rainer Hofmann, Markus Schuß, Jakob Link, Matthias Hollick, Carlo Alberto Boano, and Kay Römer. 2021. Leveraging Cross-Technology Broadcast Communication to Build Gateway-Free Smart Homes. In *2021 17th International Conference on Distributed Computing in Sensor Systems (DCOSS)*. Institute of Electrical and Electronics Engineers, New York, NY, USA, 1–9. doi:10.1109/DCOSS52077.2021.00014

[5] Michael Büsch. 2008. b43-tools. https://git.bues.ch/git/b43-tools.git

[6] M. Cagalj, S. Capkun, and J.-P. Hubaux. 2006. Key Agreement in Peer-to-Peer Wireless Networks. *Proc. IEEE* 94, 2 (Feb. 2006), 467–478. doi:10.1109/JPROC.2005.862475

[7] M. Cagalj, S. Capkun, R. Rengaswamy, I. Tsigkogiannis, M. Srivastava, and J.-P. Hubaux. 2006. Integrity (I) Codes: Message Integrity Protection and Authentication over Insecure Channels. In *2006 IEEE Symposium on Security and Privacy (S&P'06)*. Institute of Electrical and Electronics Engineers, New York, NY, USA, 15 pp.–294. doi:10.1109/SP.2006.23

[8] Srdjan Čapkun, Mario Čagalj, Ramkumar Rengaswamy, Ilias Tsigkogiannis, Jean-Pierre Hubaux, and Mani Srivastava. 2008. Integrity Codes: Message Integrity Protection and Authentication over Insecure Channels. *IEEE Transactions on Dependable and Secure Computing* 5, 4 (Oct. 2008), 208–223. doi:10.1109/TDSC.2008.11

[9] Marco Cominelli, Shaghayegh Shahcheraghi, Jakob Link, Matthias Hollick, Federico Cerutti, Francesco Gringoli, and Arash Asadi. 2024. Physical-Layer Privacy via Randomized Beamforming Against Adversarial Wi-Fi Sensing: Analysis, Implementation, and Evaluation. *IEEE Transactions on Wireless Communications* 23, 12 (Dec. 2024), 19603–19617. doi:10.1109/TWC.2024.3485477

[10] D. Dolev and A. Yao. 1983. On the Security of Public Key Protocols. *IEEE Transactions on Information Theory* 29, 2 (March 1983), 198–208. doi:10.1109/TIT.1983.1056650

[11] 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 (MobiCom '13)*. Association for Computing Machinery, New York, NY, USA, 417–428. doi:10.1145/2500423.2500428

[12] Matthias Fassl and Katharina Krombholz. 2023. Why I Can't Authenticate — Understanding the Low Adoption of Authentication Ceremonies with Autoethnography. In *Proceedings of the 2023 CHI Conference on Human Factors in Computing Systems (CHI '23)*. Association for Computing Machinery, New York, NY, USA, 1–15. doi:10.1145/3544548.3581508

[13] Mikhail Fomichev, Flor Álvarez, Daniel Steinmetzer, Paul Gardner-Stephen, and Matthias Hollick. 2018. Survey and Systematization of Secure Device Pairing. *IEEE Communications Surveys & Tutorials* 20, 1 (2018), 517–550. doi:10.1109/COMST.2017.2748278

[14] Nirnimesh Ghose, Loukas Lazos, and Ming Li. 2017. HELP: Helper-Enabled In-Band Device Pairing Resistant Against Signal Cancellation. In *26th USENIX Security Symposium (USENIX Security 17)*. USENIX Association, USA, 433–450.

[15] Nirnimesh Ghose, Loukas Lazos, and Ming Li. 2018. Secure Device Bootstrapping Without Secrets Resistant to Signal Manipulation Attacks. In *2018 IEEE Symposium on Security and Privacy (SP)*. 819–835. doi:10.1109/SP.2018.00055

[16] Nirnimesh Ghose, Loukas Lazos, and Ming Li. 2018. SFIRE: Secret-Free-in-band Trust Establishment for COTS Wireless Devices. In *IEEE INFOCOM 2018 - IEEE Conference on Computer Communications*. 1529–1537. doi:10.1109/INFOCOM.2018.8486417

[17] Nirnimesh Ghose, Loukas Lazos, and Ming Li. 2022. In-Band Secret-Free Pairing for COTS Wireless Devices. *IEEE Transactions on Mobile Computing* 21, 2 (Feb. 2022), 612–628. doi:10.1109/TMC.2020.3015010

[18] Shyamnath Gollakota, Nabeel Ahmed, Nickolai Zeldovich, and Dina Katabi. 2011. Secure In-Band Wireless Pairing. In *20th USENIX Security Symposium (USENIX Security 11)*.

[19] Francesco Gringoli, Marco Cominelli, Alejandro Blanco, and Joerg Widmer. 2021. AX-CSI: Enabling CSI Extraction on Commercial 802.11ax Wi-Fi Platforms. In *Proceedings of the 15th ACM Workshop on Wireless Network Testbeds, Experimental Evaluation & CHaracterization (WiNTECH '21)*. Association for Computing Machinery, New York, NY, USA, 46–53. doi:10.1145/3477086.3480833

[20] Francesco Gringoli and Lorenzo Nava. 2009. OpenFWWF - Open FirmWare for WiFi Networks. http://netweb.ing.unibs.it/openfwwf/

[21] Francesco Gringoli, Matthias Schulz, Jakob Link, and Matthias Hollick. 2019. Free Your CSI: A Channel State Information Extraction Platform For Modern Wi-Fi Chipsets. In *Proceedings of the 13th International Workshop on Wireless Network Testbeds, Experimental Evaluation & Characterization (WiNTECH '19)*. Association for Computing Machinery, New York, NY, USA, 21–28. doi:10.1145/3349623.3355477

[22] Yaqi He, Kai Zeng, Long Jiao, Brian L. Mark, and Khaled N. Khasawneh. 2024. Swipe2Pair: Secure and Fast In-Band Wireless Device Pairing. In *Proceedings of the 17th ACM Conference on Security and Privacy in Wireless and Mobile Networks (WiSec '24)*. Association for Computing Machinery, New York, NY, USA, 196–206. doi:10.1145/3643833.3656127

[23] Yantian Hou, Ming Li, Ruchir Chauhan, Ryan M. Gerdes, and Kai Zeng. 2015. Message Integrity Protection over Wireless Channel by Countering Signal Cancellation: Theory and Practice. In *Proceedings of the 10th ACM Symposium on Information, Computer and Communications Security (ASIA CCS '15)*. Association for Computing Machinery, New York, NY, USA, 261–272. doi:10.1145/2714576.2714617

[24] Yantian Hou, Ming Li, and Joshua D. Guttman. 2013. Chorus: Scalable in-Band Trust Establishment for Multiple Constrained Devices over the Insecure Wireless Channel. In *Proceedings of the Sixth ACM Conference on Security and Privacy in Wireless and Mobile Networks (WiSec '13)*. Association for Computing Machinery, New York, NY, USA, 167–178. doi:10.1145/2462096.2462124

[25] Broadcom Inc. 2025. BCM4389 | Wi-Fi 6E and Bluetooth 5 Combo Chipset. https://www.broadcom.com/products/wireless/wireless-lan-bluetooth/bcm4389

[26] Synaptics Inc. 2025. Synaptics Veros™ Wireless Connectivity Solutions. https://www.synaptics.com/products/wireless

[27] Sameh Khalfaoui, Jean Leneutre, Arthur Villard, Jingxuan Ma, and Pascal Urien. 2021. Security Analysis of Out-of-Band Device Pairing Protocols: A Survey. *Wireless Communications and Mobile Computing* 2021, 1 (2021), 8887472. doi:10.1155/2021/8887472

[28] Patrick Leu, Martin Kotuliak, Marc Roeschlin, and Srdjan Capkun. 2021. Security of Multicarrier Time-of-Flight Ranging. In *Proceedings of the 37th Annual Computer Security Applications Conference (Acsac '21)*. Association for Computing Machinery, New York, NY, USA, 887–899. doi:10.1145/3485832.3485898

[29] Jakob Link, David Breuer, Francesco Gringoli, and Matthias Hollick. 2023. Rolling the D11: An Emulation Game for the Whole BCM43 Family. In *Proceedings of the 17th ACM Workshop on Wireless Network Testbeds, Experimental Evaluation & Characterization (WiNTECH '23)*. Association for Computing Machinery, New York, NY, USA, 88–95. doi:10.1145/3615453.3616520

[30] Ueli Maurer. 1997. Information-Theoretically Secure Secret-Key Agreement by NOT Authenticated Public Discussion. In *Advances in Cryptology — EUROCRYPT '97*, Walter Fumy (Ed.). Springer, Berlin, Heidelberg, 209–225. doi:10.1007/3-540-69053-0_15

[31] Andrew Miklas and Johannes Berg. 2016. bcm-v4. https://bcm-v4.sipsolutions.net/

[32] Shahab Mirzadeh, Haitham Cruickshank, and Rahim Tafazolli. 2014. Secure Device Pairing: A Survey. *IEEE Communications Surveys & Tutorials* 16, 1 (2014), 17–40. doi:10.1109/SURV.2013.111413.00196

[33] Daniel Moser, Vincent Lenders, and Srdjan Capkun. 2019. Digital Radio Signal Cancellation Attacks: An Experimental Evaluation. In *Proceedings of the 12th Conference on Security and Privacy in Wireless and Mobile Networks (WiSec '19)*. Association for Computing Machinery, New York, NY, USA, 23–33. doi:10.1145/3317549.3319720

[34] Moses Namara and Bart P. Knijnenburg. 2021. The Differential Effect of Privacy-Related Trust on Groupware Application Adoption and Use during the COVID-19 Pandemic. *Proc. ACM Hum.-Comput. Interact.* 5, CSCW2 (Oct. 2021), 405:1–405:34. doi:10.1145/3479549

[35] Sean Oesch, Ruba Abu-Salma, Oumar Diallo, Juliane Krämer, James Simmons, Justin Wu, and Scott Ruoti. 2022. User Perceptions of Security and Privacy for Group Chat. *Digital Threats* 3, 2 (Feb. 2022), 15:1–15:29. doi:10.1145/3491265

[36] Yanjun Pan, Yantian Hou, Ming Li, Ryan M. Gerdes, Kai Zeng, Md. A. Towfiq, and Bedri A. Cetiner. 2020. Message Integrity Protection Over Wireless Channel: Countering Signal Cancellation via Channel Randomization. *IEEE Transactions on Dependable and Secure Computing* 17, 1 (Jan. 2020), 106–120. doi:10.1109/TDSC.2017.2751600

[37] Christina Pöpper, Nils Ole Tippenhauer, Boris Danev, and Srdjan Capkun. 2011. Investigation of Signal and Message Manipulations on the Wireless Channel. In *Computer Security – ESORICS 2011*, Vijay Atluri and Claudia Diaz (Eds.). Springer, Berlin, Heidelberg, 40–59. doi:10.1007/978-3-642-23822-2_3

[38] 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 (WiSec '20)*. Association for Computing Machinery, New York, NY, USA, 31–41. doi:10.1145/3395351.3399420

[39] Florentin Putz, Steffen Haesler, and Matthias Hollick. 2024. Sounds Good? Fast and Secure Contact Exchange in Groups. *Proc. ACM Hum.-Comput. Interact.* 8, CSCW2 (Nov. 2024), 425:1–425:44. doi:10.1145/3686964

[40] Florentin Putz, Steffen Haesler, Thomas Völkl, Maximilian Gehring, Nils Rollshausen, and Matthias Hollick. 2024. PairSonic: Helping Groups Securely Exchange Contact Information. In *Companion Publication of the 2024 Conference on Computer-Supported Cooperative Work and Social Computing (CSCW Companion '24)*. Association for Computing Machinery, New York, NY, USA, 69–71. doi:10.1145/3678884.3681818

[41] Matthias Schulz. 2018. *Teaching Your Wireless Card New Tricks: Smartphone Performance and Security Enhancements Through Wi-Fi Firmware Modifications*. Ph. D. Dissertation. Technische Universität Darmstadt, Darmstadt. doi:10.26083/tuprints-00007243

[42] Matthias Schulz, Francesco Gringoli, Daniel Steinmetzer, Michael Koch, and Matthias Hollick. 2017. Massive Reactive Smartphone-Based Jamming Using Arbitrary Waveforms and Adaptive Power Control. In *Proceedings of the 10th ACM Conference on Security and Privacy in Wireless and Mobile Networks (WiSec '17)*. Association for Computing Machinery, New York, NY, USA, 111–121. doi:10.1145/3098243.3098253

[43] Matthias Schulz, Jakob Link, Francesco Gringoli, and Matthias Hollick. 2018. Shadow Wi-Fi: Teaching Smartphones to Transmit Raw Signals and to Extract Channel State Information to Implement Practical Covert Channels over Wi-Fi. In *Proceedings of the 16th Annual International Conference on Mobile Systems, Applications, and Services (MobiSys '18)*. Association for Computing Machinery, New York, NY, USA, 256–268. doi:10.1145/3210240.3210333

[44] Matthias Schulz, Daniel Wegemer, and Matthias Hollick. 2017. Nexmon: Build Your Own Wi-Fi Testbeds With Low-Level MAC and PHY-Access Using Firmware Patches on Off-the-Shelf Mobile Devices. In *Proceedings of the 11th Workshop on Wireless Network Testbeds, Experimental Evaluation & CHaracterization (WiNTECH '17)*. Association for Computing Machinery, New York, NY, USA, 59–66. doi:10.1145/3131473.3131476

[45] Matthias Schulz, Daniel Wegemer, and Matthias Hollick. 2017. Nexmon: The C-based Firmware Patching Framework. https://nexmon.org

[46] Markus Schuß, Carlo Alberto Boano, Kay Römer, Jakob Link, and Matthias Hollick. 2020. Towards an Automated Monitoring of RF Activity in Low-Power Wireless Testbeds. In *3rd Workshop on Benchmarking Cyber-Physical Systems and Internet of Things (CPS-IoTBench 2020)*. London, United Kingdom.

[47] Wenlong Shen, Bo Yin, Lu Liu, Xianghui Cao, Yu Cheng, Qing Li, and Wenjing Wang. 2016. Secure In-Band Bootstrapping for Wireless Personal Area Networks. *IEEE Internet of Things Journal* 3, 6 (Dec. 2016), 1385–1394. doi:10.1109/JIOT.2016.2604221

[48] Cypress Semiconductor Corporation/Infineon Technologies. 2019. *CYW43455 Single-chip 5G WiFi IEEE 802.11n/Ac MAC/Baseband/Radio with Integrated Bluetooth 5.0 Document Number: 002-15051 Rev. *O.*

[49] Nils Ole Tippenhauer, Kasper Bonne Rasmussen, and Srdjan Capkun. 2016. Physical-Layer Integrity for Wireless Messages. *Computer Networks* 109 (Nov. 2016), 31–38. doi:10.1016/j.comnet.2016.06.021

A Copyright