



# SPAWN: Seamless Proximity-Based Authentication by Utilizing the Existent WiFi Environment

Philipp Jakubeit<sup>1</sup>(✉) , Andreas Peter<sup>1,2</sup> , and Maarten van Steen<sup>1</sup> 

<sup>1</sup> University of Twente, Drienerlolaan 5, 7522 NB Enschede, Netherlands  
p.jakubeit@utwente.nl

<sup>2</sup> Carl von Ossietzky Universität Oldenburg, Ammerländer Heerstraße 114-118,  
26129 Oldenburg, Germany

**Abstract.** Our objective is to create a transparent authentication factor using existing hardware and information already present. Transparent authentication refers to not burdening the user with interaction, and therefore a transparent authentication factor is applicable not only at the beginning of a session but continuously during an authenticated session. We choose to utilize the WiFi environment of a user, as it is ubiquitous in terms of the presence of WiFi signals and user hardware. As we intend our contribution as an addition to stand-alone passwords or existing multifactor-authentication schemes, we decided to build on the concept of separated authentication channels used in state-of-the-art multifactor authentication. To do so, we require two devices. Measuring the WiFi environment from two points enables us to use the proximity of devices as the additional authentication claim. In this work, we demonstrate that it is feasible to use WiFi to identify the proximity of devices. We analyze two scenarios, a semi-densely populated apartment environment and a densely populated office environment in terms of WiFi access points. In the apartment scenario, we show that SPAWN provides at least as much entropy as a traditional password, while not requiring the user to retype a low-entropy token. In the office scenario, this amount of entropy can still be derived in 74% of the measures. By applying private set metrics, we investigate and demonstrate that a device's proximity can be employed as an authentication factor without compromising privacy.

## 1 Introduction

We are faced with the increasing prevalence of multifactor-authentication (MFA) systems on a daily basis. According to the Global Password Security Report [14], 57% of online businesses had adopted MFA schemes in 2019. The most common form of MFA in use is two-factor authentication (2FA), which combines a knowledge factor (the user's password) and a possession factor (a code sent via SMS or an app). NIST describes the use of MFA as a 'security enhancement' [22]. However, most 2FA schemes lack usability by requiring user input. This is confirmed by a 2023 MFA report, which shows that 33% of the users turn off

MFA due to its annoyance [18]. Furthermore, authentication is still only used at the beginning of an authenticated session. While some more secure systems such as banking applications use strict session timeouts, systems that continuously evaluate the user’s authenticity are virtually nonexistent.

We propose SPAWN, a seamless, proximity-based authentication mechanism that utilizes the user’s WiFi environment as an additional, transparent authentication factor. The concept is that when a user is required to multifactor authenticate, a second device is in close proximity. This is true for traditional two-factor authentication, as the second device must be accessible for the user to receive and retype the token. The main aspect of the multifactor scheme to date is to have two separate communication channels with the service to provide reassurance of the user’s authenticity. However, the user is kept in the loop to retype a one-time token. We bypass the user involvement in utilizing environmental measurements. Having a device nearby provides sufficient information to confirm that the entity is likely to be the user the entity claims to be. Therefore, SPAWN can be used as a transparent factor in MFA systems. A *transparent authentication* factor allows a user to be authenticated without the user’s participation in the authentication process. Furthermore, due to its transparency, SPAWN can be used as a continuous authentication factor. A *continuous authentication* factor allows the system to authenticate a user continuously, not only at the beginning of an authenticated session, but on demand. We can even use SPAWN for multiple devices in the user’s vicinity. For example, when an infant wants to access an online streaming service, it may not be sufficient for the smartphone to be close to the smart TV, but also for the smartphone of a legal guardian to be in proximity to allow for a valid authentication.

SPAWN works by measuring the WiFi environment of the user on at least two devices. We measure the user’s WiFi environment in terms of existing WiFi access points (AP) in the surroundings of the two devices and take advantage of the uniqueness demonstrated of a WiFi fingerprint [8]. These measurements are encrypted with a fresh key and sent to the service for comparison. The fresh key is derived from a session key and a session-specific identifier. The latter is communicated by the service, while the former is created and communicated via a software-spawned hotspot by one of the user devices. This mechanism allows SPAWN to provide the WiFi-based proximity authentication factor while protecting the confidentiality of the measurements. Our main contributions are:

- We investigate the feasibility of using the WiFi environment of a user to determine if devices are in close proximity.
- We demonstrate that it is feasible to use WiFi to identify the proximity of devices and investigate the guarantees that can be obtained.
- We show that the proximity of the device can be used as an authentication factor.
- We show that it is possible to assert proximity of devices without compromising privacy (each device only learns its own measurement, and the service only learns the number of APs per measurement and their intersection size).

## 2 Related Work

Different systems have been proposed for secure, transparent, and continuous authentication. One area of research is behavioral profiling, which utilizes user dynamics such as call and application usage, to detect irregularities [2]. Another field of study is the use of input devices, with keystroke dynamics being used to authenticate a user [1]. Modern input devices, such as touchscreens, have been used to develop biometric profiles with a classification success rate of 100% in less than four seconds [3]. Even the position that a user holds the device can be used for user recognition [19]. However, all of these methods involve some form of classification, which is time consuming and computationally expensive. In contrast, SPAWN requires only a comparison without prior training.

Since 2020, tracking infectious diseases has become a popular area of research. This involves measuring the proximity of devices of different users, usually using Bluetooth or WiFi. For example, WiFi network logs from enterprise networks are used to reconstruct device trajectories for contact tracing [23]. However, this is limited to enterprise environments and requires access to enterprise WiFi logs. Another approach is to require changes to existing routers [27]. Again, training and classification add to the costs of such systems.

Regarding the proximity of devices, several paths are researched. An agreed frequency can be used to measure proximity [4]. Synthesized signal-strength data on different channels is used to increase the entropy of proximity-based key generation [13]. WiFi signals and WiFi technical specifications can be used to determine the proximity of devices [16]. The channel state information (CSI) of the ambient WiFi signals is used to determine the proximity of devices [21]. Additional hardware in terms of backscatter tags is used [15] and auditory proximity detection [6]. However, SPAWN is a unique approach as it measures the environment of the two devices in terms of observable APs, regardless of the state of the WiFi connection. Note that we do not compare to distance-bounding protocols, as they aim to establish an upper bound on the physical distance, while we evaluate the similarity of the WiFi environment to infer proximity.

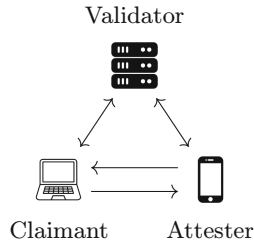
To compare observed APs and ensure the desired privacy guarantees, we employ a technique known as private set intersection cardinality (PSI-CA). There are several ways to implement PSI-CA. One way is to use multiparty computation combined with oblivious transfer [17]. Another approach is a generic transformation to construct PSI from key agreement protocols [20]. Lastly, functional encryption can be used to realize PSI and PSI-CA [24].

## 3 System Description

SPAWN has two interacting parties: a user who intends to conduct authentication and a service validating the user’s authentication attempt. Analogously to traditional 2FA schemes, we require at least three devices, two user devices, and the service. The service acts as a third party *validator*, verifying the authentication claims. The first device of the user is the *claimant*, which produces the claim

to be validated. All other user devices are *attesters*, supporting the claimant’s claim against the validator. Having only communication channels between the devices and the service could further improve the security of our authentication scheme, as it mitigates man-in-the-middle and eavesdropping attacks. However, this requires more data than is accessible via operating system (OS) abstractions (see Sect. 6). Therefore, we assume not only a separate communication channel with the validator but also an additional one-way connection between the user devices. In Fig. 1 we show the three devices and their connectivity.

SPAWN ensures that the user devices and the validator are familiar with each other and can communicate securely. Both user devices perform a WiFi measurement, and the validator evaluates the similarity of these measurements. For this evaluation, we utilize the degree of overlap in the measurements and the ratio of overlap to all observed APs. Considering both the number of shared APs and the proportion of overlap to observed APs, allows us to determine how much information is shared by the devices’ measurements and how similar the two measurements are. The establishment of asymmetrically secured communication between the devices and the validator creates an initial level of trust. Both the device and the validator know that their communication is confidential and authenticated. Further, the asymmetrically secured communication allows for non-repudiation, thus ensuring the integrity and accountability of the communication. Furthermore, symmetrically secured communication between the two user devices creates a secondary level of trust by providing confidentiality, integrity, and authentication based on the possession of the symmetric key. Additionally, the utilization of current environmental measurements is necessary to achieve a third level of trust due to showing an overlap in the measured WiFi environment, which upholds a previously specified amount of shared information and required similarity. This third level of trust guarantees the desired proximity claim for transparent authentication. Furthermore, including the measurements is crucial in preventing relay attacks, providing recency due to a service nonce, and proximity due to the overlap of APs in the measurements.



**Fig. 1.** This figure illustrates the communication channels between a laptop a smartphone, and the validator.

### 3.1 Identity Assertions and Security Foundations

The major components of SPAWN are the WiFi-capable user devices. We require at least  $n = 2$  user devices  $\{D_1, \dots, D_n\}$  to determine proximity. Additionally, we require the validator for verification. We assume that  $D_1$  is the claimant. *The claimant* collects WiFi measurements and communicates with the validator under encryption, for example, using a TLS-secured channel. In addition,

it can communicate with the second user device, the attester. Each device  $D_j$  for  $j \in \{2, \dots, n\}$  functions as an attester. *The attester* should confirm the authentication claim of the claimant. It collects WiFi measurements and communicates with the validator under encryption. In addition, it has a means of communication with the claimant. This *device-to-device* (D2D) communication in SPAWN is required to ensure up-to-dateness. We assume that the devices are capable of communicating wirelessly. We realize wireless communication using a *SoftSpot*, a software-spawned WiFi hotspot. Additionally, we assume means of direct communication for the initialization of a pre-shared key (e.g., NFC, cable connection). The *pre-shared key* (PSK) serves multiple purposes: A private PSK facilitates a trust relationship between user devices and guarantees the confidentiality and integrity of data exchanged between devices. Explicitly, the PSK protects against replay and man-in-the-middle attacks.

### 3.2 Conducting and Comparing Measurement

To make measurements, recall the devices  $D_i$  for  $i \in \{1, \dots, n\}$ . Each device is tasked with performing a WiFi measurement of the surrounding APs. We assume that the measurement  $\mathbf{M}_i$  for a device  $D_i$  includes the media access control (MAC) address and the service set identifier (SSID) of each observed AP, represented by  $MAC(AP)$  and  $SSID(AP)$ , respectively. These two pieces of information are available on all operating systems (OSs) without requiring privileged access rights. Therefore, the measurement is a set of tuples of the MAC address and SSID for each observed AP. We assume  $\mathbf{env}(D_i)$  to be the WiFi environment, measurable in the current position of the device  $D_i$ . The measurement is defined as the set:  $\mathbf{M}_i = \{(MAC(AP), SSID(AP)) | AP \in \mathbf{env}(D_i)\}$ .

To compare the measurements, the validator must validate the proximity of the claimant device and the attester device. To do so, the validator compares the set of APs measured by the devices. If only one device is required to be in proximity, the validator compares only two measurements. If more devices are required, the validator evaluates each attester’s measurement against the claimant’s measurement, for example,  $D_1$  and  $D_2$  as well as  $D_1$  and  $D_3$ .

*Requirements.* For each participating party, the validator, the claimant device  $D_1$ , and each attester device  $D_j$ , we require that each device  $D_i$  learns no more than its own measured set of APs  $\mathbf{M}_i$ . The validator learns none of the sets  $\mathbf{M}_i$ .

Outsourced private set intersection cardinality (PSI-CA) protocols fulfill these requirements by letting the participating user devices learn nothing more than their sets and the validator the intersection cardinality of the sets. PSI-CA is a cryptographic primitive in which only the cardinality of the intersection of two private input sets is learned. Contrary to most of the solutions in the literature, we require an outsourced evaluation in which neither of the parties learns cardinality [10, 24]. Only the third-party validator should acquire this knowledge. The second requirement in our use case is that the applied solution is either non-interactive or employs mechanisms to prevent cheating from the participating parties. We require this, as we apply PSI-CA to derive an authentication claim.

### 3.3 Authentication Process

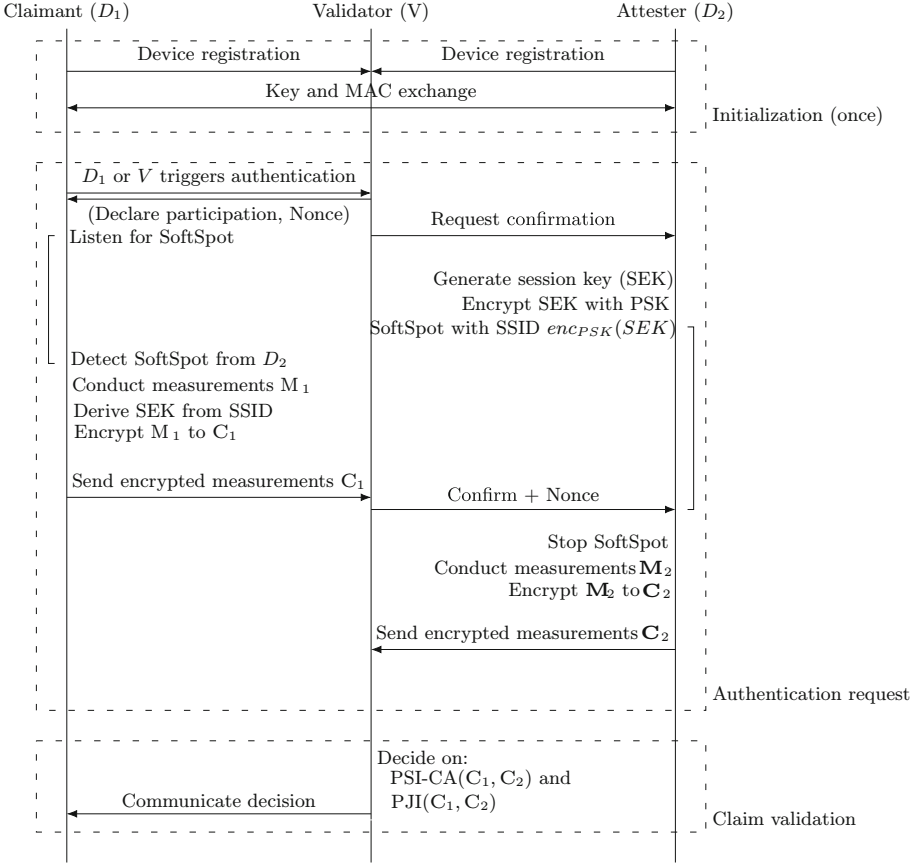
The authentication process is divided into three stages: initialization, authentication request, and validation of the authentication claim. We provide a visual representation of these stages in Fig. 2. In addition, we discuss user participation and session management.

**Initialization.** Initialization is carried out once to bind the devices to each other. Each claimant and attester device are registered at the validator to participate in the SPAWN-based authentication. We assume that this is handled by dedicated software “apps,” from the validator, running on the user devices. Furthermore, we assume that these apps provided by the validator communicate and establish an encrypted channel between the device and the validator. For the binding of the claimant and the attester, we assume previously mentioned close-proximity communication in the form of NFC or a cabled connection. The two apps will offer ways of generating and exchanging a shared secret between them. We call this randomly generated key the PSK. The key can be generated locally or by a key exchange forwarded by the validator. Local PSK generation involves either one device that takes the role of key generation or a multiparty approach. In the case of a lead device, the key is simply exchanged locally. Alternatively, a key exchange could be conducted through the validator. We discuss the security implications of these two scenarios in Sect. 5. Furthermore, the devices will store each other’s MAC address as an identifier.

**Authentication Request.** An authentication request can be triggered by the claimant device or the validator. The claimant and the validator have a one-round exchange about a request being scheduled. Additionally, the validator sends a fresh nonce to the claimant device. The claimant starts listening to the WiFi environment for the attester’s MAC address. In parallel, the validator requests confirmation from the attester device. In the following, the attester device does several things. The attester device generates a random 32-byte session key (SEK) and encrypts it with the PSK,  $enc_{PSK}(SEK)$ , before spawning a SoftSpot, with  $enc_{PSK}(SEK)$  as the SSID.

As soon as the claimant device measures the SoftSpot of the attester device (identified by the MAC address), it does three things. First, the claimant device decrypts the SEK by applying the PSK,  $dec_{PSK}(enc_{PSK}(SEK))$ . Second, the claimant,  $D_1$  conducts measurements  $\mathbf{M}_1$  (dropping the SoftSpot) and encrypts the observed AP representation set with a *key* derived from the SEK and the nonce,  $key = SEK || nonce$ . Third, it sends the ciphertext set to the validator. Given an encryption function  $enc_{key}(message)$  and the set of measurements  $\mathbf{M}_i$ , the set of ciphertext  $\mathbf{C}_i$  is defined as  $\mathbf{C}_i = \{enc_{key}(m) | m \in \mathbf{M}_i\}$ .

The validator communicates the nonce to the attester device and that it has received a claim. The attester device ( $D_2$ ) stops the SoftSpot, measures the WiFi environment, creates  $key = SEK || nonce$  and encrypts its measurements  $\mathbf{M}_2$  with *key* to receive  $\mathbf{C}_2$ .



**Fig. 2.** This figure shows the steps of the SPAWN authentication process with two user devices, one acting as the claimant and the other as the attester, and a service as the validator  $V$ . Furthermore, the three phases of the authentication process (initialization, authentication request, and claim validation) are depicted. Session management is not included in the figure, as it deals with when the authentication request and claim validation are used together. The two solid bars on the left and right in the authentication-request phase represent the time that the user device listens to the SoftSpot and must keep the SoftSpot active.

**Claim Validation.** The claimant has provided a set of ciphertext  $C_1$ , which the validator validates against the set of ciphertext received  $C_2$  from the attester device. As the sets are encrypted, the validator cannot access the content but can determine only the cardinality of each set,  $|C_1|$  and  $|C_2|$ . When comparing the encrypted elements of the sets, the validator can calculate the cardinality of the intersection of the two sets by computing  $PSI-CA(C_1, C_2) = |C_1 \cap C_2|$ . Knowing the cardinality of both sets and the cardinality of their intersection, the validator can then compute the cardinality of the union of the two sets applying

$\text{PSU-CA}(\mathbf{C}_1, \mathbf{C}_2) = |\mathbf{C}_1| + |\mathbf{C}_2| - \text{PSI-CA}(\mathbf{C}_1, \mathbf{C}_2)$ . We call this private set union cardinality (PSU-CA) analogously to PSI-CA. Validation is carried out on the basis of the following metrics: the number of shared APs and the proportion of overlap to all observed APs.

*Number of Shared APs.* The PSI-CA determines the amount of information that an adversary must guess. Depending on the data used for an AP, each AP contains a varying amount of information. The literature reports a min-entropy of the MAC address of 7.15-bits [9]. As we intend to also use the SSID we looked at Wigle [26], an online service that collects information about wireless hotspots around the world. Based on the 1.2 billion (short-scale) observed unique APs, we calculate a min-entropy of 5.84 bits for the SSID. Hence, we get a total summed min-entropy of at least 12.99 bits per AP. This implies that the number of observed APs times 12.99 determines the entropy that an adversary would have to guess to determine a set of APs without prior knowledge.

We aim to utilize SPAWN as an enhancement to a multifactor-authentication system. Therefore, we evaluate the factor strength in comparison to established and authorized multifactor methods, such as a two-digit code, a six-digit code, an eight-character password (ASCII characters 32-127), and a 128-bit key.

We observe that SPAWN is dependent on the number of APs in the intersection as the entropy is  $k \times 12.99$  bits for  $k$  APs. Therefore, PSI-CA determines the amount of information that an adversary needs to guess. Even with a matching AP, SPAWN provides approximately twice the entropy compared to a two-digit code (6.64 bits). A 6-digit code (19.9 bits) is surpassed by two APs. Five APs surpass an eight-character password (52.44 bits), and from ten APs onward the entropy of a 128-bit key (128 bits) is surpassed.

*Proportion of Overlap to All Observed APs.* The intersection-to-union ratio is known as the Jaccard index (JI) [7]. It is defined by the cardinality of the intersection divided by the cardinality of the union and is shown to be capable of fingerprinting a location based on WiFi [8]. In our setting, we compute the private JI (PJI) by  $\text{PJI}(\mathbf{C}_1, \mathbf{C}_2) = \frac{\text{PSI-CA}(\mathbf{C}_1, \mathbf{C}_2)}{\text{PSU-CA}(\mathbf{C}_1, \mathbf{C}_2)}$ . We utilize the PJI as a distance metric that is specific to the location and can be calibrated. Although PSI-CA establishes the minimum information required at the intersection, PJI determines the similarity between the measurements. To consider the location and the device used, we propose a dynamic approach. During calibration, the system calculates the standard PJI for the given location and operates transparently unless the PJI deviates significantly. In such cases, the user must decide on the legitimacy of the authentication attempt.

**User Involvement.** In general, SPAWN can be used as a transparent authentication factor. However, we require a user in the loop during three scenarios.

*Calibration of a Location.* WiFi is characterized by high variability in coverage and density, as shown in the literature [8]. Therefore, there cannot be a standard

solution, but there needs to be a calibration per location. We assume that a user is prompted with a confirmation request to determine the location’s PSI-CA and PJI. This happens the first couple of times defined by the specific application.

*An Uncertain SPAWN Claim.* Even with a location-specific calibration phase that involves a user, an attester measurement can differ too much from a claimant measurement. In this case, the attester device is prompted with an authentication request, and the claimant device is notified of the denied attempt. If the PCI-CA still contains sufficient APs for the desired security level, the expected PJI could be adapted according to the new and confirmed measurements.

*Device Compromise.* In case of noticing theft or loss of a device, the user is the fastest hope of detection. Therefore, the user is compelled by the user’s own security to notify the validator about a missing device. This is especially relevant, as proximity to the absent device would be enough to confirm a login.

**Session Management.** The session covers the entire period from beginning to end. The start of the session is the moment when authentication takes place. In certain contexts, such as banking or other more secure systems, a session is timed out. Although continuous authentication is not a novel concept [5], it is hardly ever deployed. SPAWN allows transparent and therefore continuous authentication. In practice, a validator can initiate proximity validation at any time during the session. The validator requests the two devices to measure their environment and thereby can dynamically detect proximity. As a result, the session can also be terminated due to a failed proximity claim validation.

### 3.4 Instantiation

The PSI-CA protocol proposed in [24] fits both requirements of SPAWN precisely, the requirement of an external validator and the noninteractiveness. In [24] the validator is called the evaluator with the requirement that only the evaluator learns PSI-CA and that the participating parties learn nothing more than their own set. The proposed noninteractive protocol is an efficient two-client functional encryption (2C-FE) based PSI and PSI-CA scheme.

For small set sizes, the system proposed by [24] is quite efficient. We presume to deal with small set sizes, with an estimated maximum of about 100 APs in a user’s WiFi environment [8]. We guarantee by the PSK and local SoftSpot that users indeed have the same matching key and utilize the nonce shared by the validator as a session-specific identifier. Therefore, the only remaining question is how to instantiate the PRF.

*Instantiating PRF.* The PRF can be constructed based on a block cipher, a keyed hash function, or a hash-based message authentication code [24]. We chose the latter and instantiated the PRF required by applying KMAC, the Keccak-based message authentication code [11]. By choosing KMAC, we can independently

adapt the key and output size. Furthermore, it is based on Keccak, the latest secure hash algorithm to be standardized by NIST [11].

### 2C-FE based PSI-CA [24]

---

*Assumptions:* A pseudo-random function PRF

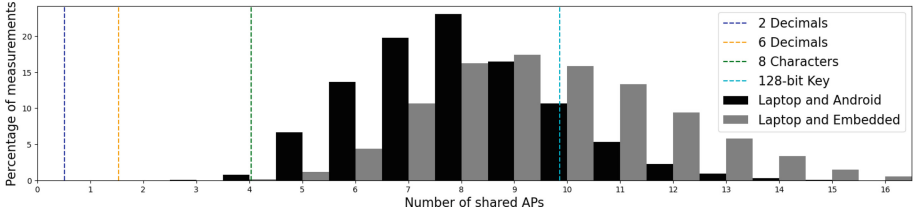
1. **Setup** Choose a PRF from an ensemble of PRFs and assume matching client keys.
  2. **Encrypt** Each client encrypts each set element together with a session-specific identifier by applying the chosen PRF for each set element and outputs the set of encrypted and identifier-tagged elements.
  3. **Evaluate** The evaluator computes the set intersection’s cardinality of the encrypted sets and derives the cardinality of the original set intersections.
- 

## 4 Results

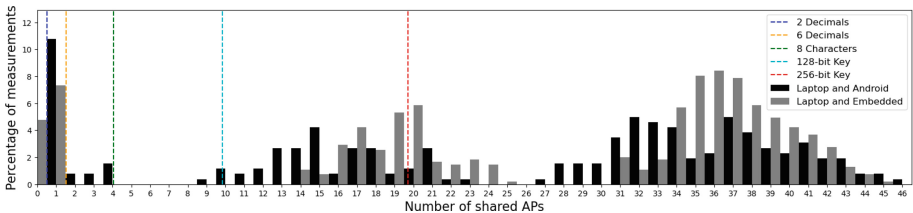
We decided to present two scenarios since considering all possible WiFi environments is infeasible: a semi-densely populated apartment environment with a maximum of 34 APs observed and a densely populated office environment with a maximum of 87 APs observed. We conducted measurements at both locations with three devices: an off-the-shelf laptop, with an Intel Alder Lake-P PCH CNVi WiFi chip, an Android device from Asus, the ZenFone 8, and an embedded WiFi chip on a Raspberry Pi 4, the Broadcom BCM4345<sup>1</sup>. Interestingly, at both locations, the maximum number of APs was measured by the embedded chipset. Figure 3 illustrates that, in the exemplary apartment environment, the measurements between a laptop and an Android device yield over 99% of instances where the entropy surpasses that of a password-based authentication factor. Similarly, when considering embedded hardware and the laptop, 100% of the measurements exhibit a number of shared APs such that the harvested entropy exceeds that of the minimum assumption for passwords. We conducted about 1800 measurements on the laptop, more than double on the embedded device, and 32 measurements on the Android device. Figure 4 illustrates that, in the office environment, the measurements between a laptop and an Android device yield nearly 74% of instances where the entropy surpasses that of a password-based authentication factor. Similarly, when considering embedded hardware and the laptop, almost 88% of the measurements exhibit a number of APs such that the harvested entropy surpasses that of the minimum assumption for passwords. We conducted about 3600 measurements on the laptop, about 340 measurements on the embedded device, and 14 measurements on the Android device.

---

<sup>1</sup> <https://gitlab.com/wifi-spawn/data>.



**Fig. 3.** This bar plot at an apartment illustrates the AP overlap between the laptop and the embedded device, as well as between the laptop and the Android device. Additionally, we use vertical dashed lines to represent the number of APs needed to achieve the entropy of other specified authentication methods, 2 and 6 digit numbers, 8-character passwords, and 128-bit keys. Everything to the right of a dashed line performs better.



**Fig. 4.** This bar plot at an office illustrates the AP overlap between the laptop and the embedded device, as well as between the laptop and the Android device. We use vertical dashed lines again to represent the number of APs needed to achieve the entropy of other specified authentication methods.

In addition to the overlap (i.e., intersection) of the APs measured by different devices, we determine the proximity to a measured location by applying the PJI. This can be done in two ways: dynamically or statically.

*Dynamically* determining proximity can be performed by letting the user confirm that the measured PJI is indeed sufficient and that the user wants to be authenticated in this setting. This allows us to tune the threshold at the validator based on the user’s feedback. Note that in the office scenario (Fig. 4) we observe an intersection of zero. This measurement cannot be used for transparent authentication, and in the situation of zero overlap, the system is required to prompt the user.

*Statically* determining proximity can be performed analogously to the threshold determined for location fingerprinting [8]. To achieve this, a close-by, *off-site* location is required during configuration. The threshold could then be the mean of the maximum similarity observed in the measurement at the off-site location and the minimum similarity observed in an on-site measurement [8]. We conducted 18 off-site measurements in front of the door to the apartment and the office. For the apartment, this implies that we would devalue 4.29% of the measurements. For the office, we conducted nine off-site measurements and observed

that we would devalue 26.53% of the measurements, while the remaining measurements accurately determine proximity with the statically determined threshold.

## 5 Security Analysis

Under normal operation, we consider the participation of two entities: the user and the validator. The user possesses a minimum of two devices and asserts the proximity of these devices, which is verified by the validator. During the initialization phase, device binding occurs, resulting in the user having an application (app) on each device. These applications are equipped with secure communication capabilities with the validator (e.g., TLS). Additionally, a pre-shared key is established between the user devices as part of the initialization phase. We assume that these security measures are in place.

We aim to obtain the same security goals as traditional, token-based, 2FA schemes. Explicitly resistance to impersonation attacks, parallel session attacks, guessing attacks, replay attacks, device loss, stolen-device attacks, insider attacks, and denial of service (DoS) attacks. We discuss DoS and a spoofing attack in the next paragraph on jamming and WiFi poisoning. After which we distinguish between an external adversary for most attacks, and the validator and the user as adversaries for insider attacks.

*Jamming and Poisoning the WiFi Environment.* As WiFi APs communicate wirelessly and can be created easily, jamming and poisoning are possible. If jamming attempts are conducted, the validator would notice a scarce presence of APs for authentication and would revert to a token-based 2FA. Poisoning the WiFi environment is also possible. However, it would only enrich the environment in terms of APs. To build an attack upon a poisoned environment without having access to the devices, an adversary would be required to capture the device to validator communication and hence be capable of decrypting the secure communication of the validator and the device. As we expect state-of-the-art security, such attacks will be infeasible if the scheme is implemented correctly.

*External Adversary Goals and Options.* The primary aim of an external adversary is to obtain unauthorized authentication. To acquire location-specific WiFi details, the adversary has two options. One option is to take control of a WiFi-enabled device near the user, while the other is to physically access the user's location. However, gaining physical access to a protected location, such as an apartment or an office to conduct a valid adversary measurement  $\mathbf{M}_A$ , is feasible only for a limited number of entities that are sufficiently close and authorized. Similarly, accessing WiFi-enabled hardware is also restricted to a limited number of entities who have the ability to gain access to a device near the user. However, both measurements are encrypted by the *key*, which includes a fresh nonce. Therefore, even an adversary that has a sufficiently similar WiFi measurement  $\mathbf{M}_A$  does not gain any additional advantage in relation to the objective as

both PCI-CA and PJI are computed on the encrypted set of measurements. The adversary has no means of computing  $C_A$  from  $M_A$  without the *key* or breaking the underlying encryption. The only benefit for an adversary who obtained a valid measurement  $M_A$  is the opportunity to extract location-specific data. However, this data extraction can also occur without the use of SPAWN.

To ensure that the validator deals with a genuine user, both user devices are preregistered. Preregistration entails a device-validator bound, implying that logging in can only be conducted by the claimant device. If both devices are compromised, the user’s account is protected by the built-in security mechanisms of the claimant device and the user’s password (assuming an MFA use case). If the attester device is compromised, the adversary’s capabilities are even more limited compared to the traditional second-factor token setting. This is because the claimant device is preregistered, and the adversary can only deny user access but cannot gain access using only the attester device. However, if the claimant device is compromised, the adversary can gain illicit authentication by bypassing the first authentication factor (e.g., password) and the security mechanisms of the claimant device. The only additional requirement is that the adversary must physically be close to the user with the attester device or poison the user’s environment and the environment of the attester device with the same fake AP. In summary, an adversary can circumvent SPAWN only if they have access to the claimant device *and* can successfully bypass the first authentication factor *and* be sufficiently close. This shows why device loss should be immediately communicated to the validator (see Sect. 3.3). Note that having access to the user device and bypassing the first authentication factor would also compromise traditional authentication schemes. 2FA might withstand such attempts if only the claimant device is controlled by the adversary, however, we consider requiring physical proximity sufficient as SPAWN is intended as an extension to traditional 2FA schemes.

*Corrupted Validator Goals and Options.* The objective is to learn sensitive private inputs from its users. This objective is independent of the security model chosen, either honest but curious, or malicious. In the case of an honest but curious model, the validator fulfills its duties and attempts to gather as much information as possible. In this scenario, the validator can be utilized to transmit key-exchange information, as it also carries out the transmission without the risk of a man-in-the-middle attack. In the case of a malicious model, the validator is not considered trustworthy and will take the necessary actions to obtain location-specific data. Therefore, a key exchange via the service should not be conducted to ensure that the user information is still secured by the PSK.

*Malicious User Goals and Options.* A malicious user may have the intention of misleading the system to prove proximity when there is no proximity by not providing the correct location information. Such behavior can also occur in traditional multifactor-authentication systems. For example, a user might ask their family or colleagues to forward a token for confirmation. Similarly, in the case of SPAWN, the user could share the set of ciphertexts or manipulate the WiFi

environment to deceive the system. However, executing such an attack to falsely claim proximity requires more advanced techniques than simply sharing a few digits, as in traditional 2FA schemes. It would involve reverse-engineering and modifying the validator’s app or creating an entirely artificial WiFi environment to accurately replay a request. Both are more complex endeavors than simply retyping a number in traditional multifactor-authentication schemes.

## 6 Discussion

*Multi-Device Proximity.* It is possible to use multiple devices as attesters instead of just one. When there are multiple attesters, each claimant and attester pair should follow the procedures outlined above for a single pair of devices. This approach is particularly advantageous when there is a second authority involved.

*Device-to-Validator only Communication Model.* We also considered developing a communication model for the devices that relies only on validator channels and eliminates the need for D2D communication. However, the abstractions of received packets in OSes do not provide sufficient detail. The only reason for D2D communication is to ensure the actuality of the data on both devices. By enabling the WiFi hardware to operate in ‘monitor mode’ with detailed, timestamped beacon frames, we can obtain actuality from the received packets only. However, constantly monitoring the WiFi environment would require dedicated hardware, as it continuously blocks the WiFi antenna for packet reception. Alternatively, we could introduce a new element to the beacon frames that the OSes would be required to pass on to the user. This new element could include something as simple as timestamps or even new features as provided by WiFi direct [25].

## 7 Conclusion

We showed that the user’s WiFi environment can be used to determine proximity between devices. The measurement capabilities of different devices vary, however, the information present is sufficiently distinctive to conclude proximity, and therefore provides an additional authentication factor. We showed that guessing the correct set of WiFi APs is between 74% and 100% of the cases harder than guessing a random eight-character password. We further showed that with off-site measurements SPAWN can be calibrated to recognize sufficiently close devices statically. The only additional aspect that we require is the validator’s software on the claimant device. We assume an app for this paper; however, a simple browser extension would suffice. With minor changes such as WiFi timestamps or other unique information, we could remove the D2D part and further optimize the usability and security of SPAWN. However, without additional requirements except for an installed app, SPAWN provides transparent and thus continuous authentication capabilities. This elevates usability while increasing security by being continuously applicable throughout the session.

## References

1. Crawford, H., Ahmadzadeh, E.: Authentication on the go: assessing the effect of movement on mobile device keystroke dynamics. In: Thirteenth Symposium on Usable Privacy and Security, pp. 163–173 (2017)
2. Crawford, H., Renaud, K.: Understanding user perceptions of transparent authentication on a mobile device. *J. Trust Manage.* **1**, 1–28 (2014)
3. Dee, T., Richardson, I., Tyagi, A.: Continuous transparent mobile device touchscreen soft keyboard biometric authentication. In: 32nd International Conference on VLSI Design, pp. 539–540. IEEE (2019)
4. Ghose, N., Gupta, K., Lazos, L., Li, M., Xu, Z., Li, J.: ZITA: zero-interaction two-factor authentication using contact traces and in-band proximity verification. *IEEE Trans. Mob. Comput.* **23**, 6318–6333 (2023)
5. Richard, P.: Security: active authentication. *IT Professional* **15**(4), 4–7 (2013)
6. Han, D., Chen, Y., Li, T., Zhang, R., Zhang, Y., Hedgpeth, T.: Proximity-proof: secure and usable mobile two-factor authentication. In: Proceedings of the 24th Annual International Conference on Mobile Computing and Networking, pp. 401–415 (2018)
7. Jaccard, P.: Étude comparative de la distribution florale dans une portion des alpes et des jura. *Bull. Soc. Vaudoise Sci. Nat.* **37**, 547–579 (1901)
8. Jakubeit, P., Peter, A., van Steen, M.: The measurable environment as nonintrusive authentication factor on the example of WiFi beacon frames. In: Saracino, A., Mori, P. (eds.) International Workshop on Emerging Technologies for Authorization and Authentication, pp. 48–69. Springer, Cham (2022). [https://doi.org/10.1007/978-3-031-25467-3\\_4](https://doi.org/10.1007/978-3-031-25467-3_4)
9. Jakubeit, P., Peter, A., van Steen, M.: LocKey: location-based key extraction from the WiFi environment in the user’s vicinity. In: Proceedings of the Eighteenth International Conference on Information Security Practice and Experience (2023)
10. Jolfaei, A.A., Mala, H., Zarezadeh, M.: EO-PSI-CA: efficient outsourced private set intersection cardinality. *J. Inf. Secur. Appl.* **65**, 102996 (2022)
11. Kelsey, J., Chang, S., Perlner, R.: Nist special publication 800–185: sha-3 derived functions: cshake, kmac, tuplehash and parallelhash. Tech. Rep., National Institute of Standards and Technology, Gaithersburg, MD (2016)
12. König, R., Renner, R., Schaffner, C.: The operational meaning of min-and max-entropy. *IEEE Trans. Inf. Theory* **55**, 4337–4347 (2009)
13. Li, Z., Wang, H., Fang, H.: Group-based cooperation on symmetric key generation for wireless body area networks. *IEEE Internet Things J.* **4**(6), 1955–1963 (2017)
14. LastPass by LogMeIn. The 3rd annual global password security report. Tillgänglig (2019). <https://lp-cdn.lastpass.com/lporcamedia/document-library/lastpass/pdf/en/LMI0828a-IAM-LastPass-State-of-the-Password-Report.pdf>
15. Luo, Z., Wang, W., Qu, J., Jiang, T., Zhang, Q.: ShieldScatter: improving IoT security with backscatter assistance. In: Proceedings of the 16th ACM conference on Embedded Networked Sensor Systems, pp. 185–198 (2018)
16. Pierson, T.J., Peters, T., Peterson, R., Kotz, D.: Proximity detection with single-antenna IoT devices. In: The 25th Annual International Conference on Mobile Computing and Networking, pp. 1–15 (2019)
17. Pinkas, B., Schneider, T., Weinert, C., Wieder, U.: Efficient circuit-based PSI via cuckoo hashing. In: Nielsen, J.B., Rijmen, V. (eds.) EUROCRYPT 2018. LNCS, vol. 10822, pp. 125–157. Springer, Cham (2018). [https://doi.org/10.1007/978-3-319-78372-7\\_5](https://doi.org/10.1007/978-3-319-78372-7_5)

18. Prove. State of MFA Report (2023). <https://www.prove.com/blog/prove-identity-2023-state-of-mfa-report-consumer-attitudes-multi-factor-authentication>
19. Primo, A., Phoha, V.V., Kumar, R., Serwadda, A.: Context-aware active authentication using smartphone accelerometer measurements. In: Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition Workshops, pp. 98–105 (2014)
20. Rosulek, M., Trieu, N.: Compact and malicious private set intersection for small sets. In: Proceedings of the 2021 ACM SIGSAC Conference on Computer and Communications Security, pp. 1166–1181 (2021)
21. Shah, S.W., Kanhere, S.S.: Wi-Auth: Wifi based second factor user authentication. In: Proceedings of the 14th EAI International Conference on Mobile and Ubiquitous Systems: Computing, Networking and Services, pp. 393–402 (2017)
22. NIST special Publication 800-79-2. Guidelines for the authorization of personal identity (2022). <https://www.nist.gov/itl/applied-cybersecurity/back-basics-multi-factor-authentication-mfa>
23. Trivedi, A., Zakaria, C., Balan, R., Becker, A., Corey, G., Shenoy, P.: WiFiTrace: network-based contact tracing for infectious diseases using passive WiFi sensing. Proc. ACM Interact. Mob. Wearable Ubiquitous Technol. **5**(1), 1–26 (2021)
24. van de Kamp, T., Stritzl, D., Jonker, W., Peter, A.: Two-client and multi-client functional encryption for set intersection. In: Jang-Jaccard, J., Guo, F. (eds.) ACISP 2019. LNCS, vol. 11547, pp. 97–115. Springer, Cham (2019). [https://doi.org/10.1007/978-3-030-21548-4\\_6](https://doi.org/10.1007/978-3-030-21548-4_6)
25. WiFi Alliance: WiFi Direct (2023). <https://www.wi-fi.org/discover-wi-fi/wi-fi-direct>
26. Wigle: WiFi Network Database (2022). <https://wigle.net/>
27. Yuen, B., et al.: Wi-fi and bluetooth contact tracing without user intervention. IEEE Access **10**, 91027–91044 (2022)