FastZIP: Faster and More Secure Zero-Interaction Pairing

Mikhail Fomichev
mfomichev@seemoo.tu-darmstadt.de
Technical University of Darmstadt

Julia Hesse
jhs@zurich.ibm.com
IBM Research Europe - Zurich

Lars Almon
lalmon@seemoo.tu-darmstadt.de
Technical University of Darmstadt

Timm Lippert
timm.lippert@gmail.com
Technical University of Darmstadt

Jun Han
junhan@comp.nus.edu.sg
National University of Singapore

Matthias Hollick
mhollick@seemoo.tu-darmstadt.de
Technical University of Darmstadt

ABSTRACT

With the advent of the Internet of Things (IoT), establishing a secure channel between smart devices becomes crucial. Recent research proposes *zero-interaction pairing (ZIP)*, which enables pairing without user assistance by utilizing devices' physical context (e.g., ambient audio) to obtain a shared secret key. The state-of-the-art ZIP schemes suffer from three limitations: (1) prolonged pairing time (i.e., minutes or hours), (2) vulnerability to brute-force offline attacks on a shared key, and (3) susceptibility to attacks caused by predictable context (e.g., replay attack) because they rely on limited entropy of physical context to protect a shared key. We address these limitations, proposing *FastZIP*, a novel ZIP scheme that significantly reduces pairing time while preventing offline and predictable context attacks. In particular, we adapt a recently introduced *Fuzzy Password-Authenticated Key Exchange (fPAKE)* protocol and utilize *sensor fusion*, maximizing their advantages. We instantiate *FastZIP* for intra-car device pairing to demonstrate its feasibility and show how the design of *FastZIP* can be adapted to other ZIP use cases. We implement *FastZIP* and evaluate it by driving four cars for a total of 800 km. We achieve up to *three times shorter* pairing time compared to the state-of-the-art ZIP schemes while assuring robust security with adversarial *error rates below 0.5%*.

CCS CONCEPTS

• **Security and privacy** → **Security services**; • **Computer systems organization** → *Embedded and cyber-physical systems.*

KEYWORDS

Pairing, Zero-interaction, Internet of Things, fPAKE, Sensor fusion

ACM Reference Format:
Mikhail Fomichev, Julia Hesse, Lars Almon, Timm Lippert, Jun Han, and Matthias Hollick. 2021. FastZIP: Faster and More Secure Zero-Interaction Pairing. In *The 19th Annual International Conference on Mobile Systems, Applications, and Services (MobiSys '21), June 24-July 2, 2021, Virtual, WI, USA.* ACM, New York, NY, USA, 14 pages. https://doi.org/10.1145/3458864.3467883

MobiSys '21, June 24-July 2, 2021, Virtual, WI, USA

ACM ISBN 978-1-4503-8443-8/21/06…$15.00
https://doi.org/10.1145/3458864.3467883

1 INTRODUCTION

The proliferation of the Internet of Things (IoT) urges the need to secure wireless communication between smart devices to protect data they exchange (e.g., sensor readings). Such protection is crucial to ensure user privacy and trustworthiness of IoT systems [17, 26, 27]. To secure wireless communication, unassociated devices need to establish a shared secret key—a process known as *secure pairing*. This shared key is used by devices to provide encryption and authentication. In recent years, numerous pairing schemes have been proposed, most of which rely on user assistance (e.g., entering a password) [8, 11, 30]. However, many IoT devices are not equipped with user interfaces, making user-assisted pairing impractical [17, 26]. In addition, a rapid increase in the number of smart devices limits scalability of user-assisted schemes [13, 36].

To address this problem, recent research proposes *zero-interaction pairing (ZIP)* utilizing devices' context to derive a shared secret key without user involvement [14, 17, 28, 37]. Such context is represented as a set of *sensor modalities* (e.g., audio, acceleration) collected by devices from their ambient environment. ZIP schemes utilize *colocated* devices residing in an enclosed physical space such as a car to observe similar context compared to devices outside. Specifically, the colocated devices record their context and translate it to sequences of bits called *fingerprints*, which are input to a key agreement protocol to establish a shared cryptographic key. Thus, the security of ZIP schemes relies on the unpredictability of context, which depends on the intensity and variety of ambient activity (e.g., sound, motion) occurring in the environment.

To date, a number of ZIP schemes utilizing various sensor modalities to capture context have been proposed [17, 18, 27–29, 36, 37]. These state-of-the-art schemes have three major limitations: (1) prolonged pairing time, (2) vulnerability to offline attacks, and (3) susceptibility to attacks caused by predictable context (e.g., replay). First, state-of-the-art ZIP schemes suffer from *prolonged pairing time* requiring minutes and hours of context data to establish a shared key [13, 17]. This happens because they use a cryptographic primitive called *fuzzy commitments* [22], where the entropy of a shared key *is equal to* the entropy of fingerprint bits, input to the protocol. Thus, these ZIP schemes need to obtain at least 128 bits of entropy from context to ensure that a shared key provides adequate security [2]. Obtaining these bits takes a prolonged time because many contexts change slowly. Second, state-of-the-art ZIP schemes are by design vulnerable to *offline attacks*, namely an adversary can mount a brute-force attack on a shared key by repeatably guessing the used fingerprints. These schemes can *only* withstand offline

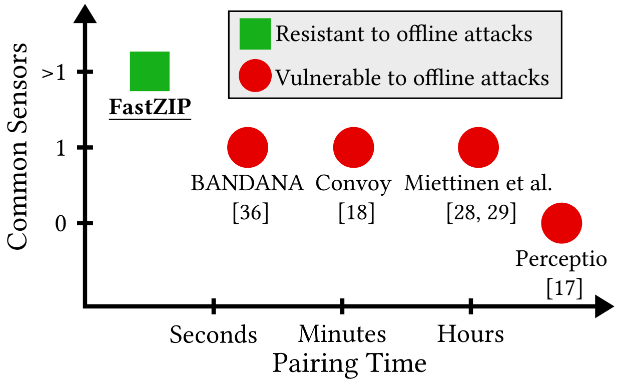


Figure 1: Design space of *FastZIP*: it provides shorter pairing time and improved security compared to state-of-the-art ZIP schemes utilizing more common sensors.

attacks if they use (1) long fingerprints (i.e., >128 bits) of (2) high entropy. However, recent works find severe entropy biases (e.g., bit patterns) in fingerprints of state-of-the-art ZIP schemes [3, 13], exposing them to offline attacks. Third, state-of-the-art ZIP schemes are susceptible to context replay, inference, or monitoring attacks due to *predictable context* [3, 13, 18]. Frequently, context becomes predictable because it relies on a single sensor modality (e.g., acceleration). Thus, an adversary can obtain similar context in comparable environments, use better hardware, or employ video analysis.

The above three limitations impair practicality and security of ZIP schemes, hindering their real-world deployment. To overcome these limitations, we propose *FastZIP*, a novel ZIP scheme that achieves shorter pairing time and improved security by addressing the following two challenges. Figure 1 compares *FastZIP* and state-of-the-art ZIP schemes in terms of paring time[1], resistance to offline attacks, and the number of common sensors required for pairing.

First, to shorten pairing time, we need to reduce the number of fingerprint bits while being robust to offline attacks. This is challenging because fewer bits means less entropy in a shared key, easing an offline attack. To address this challenge, we adapt a recently introduced *Fuzzy Password-Authenticated Key Exchange (fPAKE)* protocol [10]. fPAKE establishes a shared key from low-entropy secrets (e.g., short passwords) and is resistant to offline attacks. While fPAKE is an existing protocol, adapting it to ZIP schemes is not trivial. Specifically, we find that fPAKE protection against offline attacks is not always guaranteed in realistic ZIP settings, namely when colocated devices do not yield highly similar fingerprints from context. Thus, we analyze how to set fPAKE parameters to withstand offline attacks even in such settings (cf. Section 4). To the best of our knowledge, we are the first to implement fPAKE and demonstrate that it shortens pairing time and improves security of ZIP schemes using real-world data.

Second, to defend against predictable context attacks (e.g., replay attacks), we propose a simple form of *sensor fusion* by concatenating fingerprints derived from different sensors, each capturing distinct ambient activity. Applying sensor fusion is not straightforward, as we require a generic method to extract fingerprint bits of sufficient entropy from heterogeneous sensor signals (cf. Section 4). Existing methods rely on scenario-specific characteristics of sensor signals (e.g., peak occurence), thus cannot be directly reused, and they often produce fingerprints with entropy biases [3, 13]. We demonstrate that sensor fusion not only prevents predictable context attacks (e.g., replay) but also assists fPAKE in shortening paring time, as we obtain more bits from context, accumulating entropy faster. Sensor fusion is feasible because smart devices have multiple sensor modalities often *integrated* in one chip, for example, an inertial measurement unit (IMU) contains an accelerometer, gyroscope, and magnetometer [46], while a camera has light and RGB sensors, and a wireless chipset hosts both Wi-Fi and Bluetooth [32].

We demonstrate the advantages of *FastZIP* by evaluating a novel use case of *intra-car device pairing* (cf. Section 6), which is inspired by the growing number of smart devices inside modern cars. For example, the increasing popularity of carsharing and self-driving rides urges the need to pair multiple user devices (e.g., smartphone, earbuds) with infotainment systems of different cars to enable such services as customized driving experience [19, 35]. Furthermore, Electronic Control Units (ECUs) require pairing with wireless third-party components (e.g., tire pressure monitor) to enable travel efficiency and safety [7, 9, 43]. In both examples, the growing number of devices hinder manual pairing, requiring pairing solutions without user intervention. Despite focusing on in-car pairing, we show how the design of *FastZIP* can generalize to other ZIP use cases (e.g., smart home) to improve pairing time and security in Section 7.

Through our real-world experiments, we demonstrate the feasibility of leveraging the context of a moving car to pair devices inside it. Such context is affected by road and traffic conditions, car characteristics such as suspension, and driving patterns, and it can be captured by accelerometer, gyroscope, and barometer sensors [6, 18, 34, 45] that are ubiquitous in user devices (e.g., smartphone) and modern cars. We evaluate *FastZIP* by collecting sensor data from four cars driven over 800 km on different road types, including urban, rural, and highways. In our evaluation, we assume that pairing devices can start measuring context simultaneously by receiving a broadcast command from the car's infotainment system, which is not compromised. *FastZIP* achieves up to *three times* faster pairing compared to state-of-the-art ZIP schemes, shows error rates *below 0.5%* in the presence of a powerful adversary, and runs efficiently on off-the-shelf IoT devices. In summary, we make the following contributions:

- We design *FastZIP*, a novel ZIP scheme utilizing fPAKE and sensor fusion to reduce pairing time and improve security.
- We implement *FastZIP* for intra-car device pairing and evaluate it by collecting real-world driving data, demonstrating the effectiveness of *FastZIP*.
- We publicly release the collected data, source code of our evaluation stack, and the first implementation of fPAKE.

2 BACKGROUND

We first explain the working principle and shortcomings of fuzzy commitments—a cryptographic protocol used by state-of-the-art ZIP schemes to share a secret key. Then, we detail the fPAKE protocol [10], addressing these shortcomings, that we utilize in *FastZIP*.

[1] We use the shortest pairing time reported in the original publication for each scheme.

ZIP Based on Fuzzy Commitments. Prior work on ZIP relies on fuzzy commitments or vaults [21, 22] to exchange a key K between two devices holding similar fingerprints f, f' [17, 18, 27–29, 36, 37]. Specifically, Device A chooses a 128-bit key K and sends a commitment $c \longleftarrow \text{ECC.Encode}(K) \oplus f$ to Device B, which can recover $K \longleftarrow \text{ECC.Decode}(\text{c} \oplus \text{f}')$ if the fingerprint mismatch $f' \oplus f$ is within the error correction capability of the error correction code (ECC). While conceptually simple, this approach has two disadvantages in the case of ZIP. First, it inherently requires fingerprints f, f' to be at least 140 bits, since they are XORed to an expanded encoding of the 128-bit key[2]. Second, an eavesdropping adversary can capture the commitment c and try decoding it with arbitrarily many fingerprint guesses to obtain the key K. This constitutes an *offline attack* on K, which can only be defended against if the fingerprints have high entropy (i.e., they are hard to guess). In practice, state-of-the-art ZIP schemes already require multiple minutes or even hours to obtain fingerprints >128 bits from context [17, 18, 28, 29, 36]. Even worse, an in-depth entropy analysis reveals that fingerprints of these schemes contain bit patterns or predictable distributions of 0- and 1-bits [3, 13]. Thus, an adversary can more easily guess the fingerprints, exposing state-of-the-art schemes to offline attacks.

fPAKE Protocol. fPAKE used by *FastZIP* allows reducing the number of required fingerprint bits, hence shortening pairing time, while providing resilience to offline attacks. In essence, fPAKE is also a fuzzy commitment, but instead of creating the commitment from fingerprint f, fPAKE adds an interactive *entropy amplification* phase that turns fingerprints f, f' into high entropy keys $\mathbb{k}, \mathbb{k}'$ with a similar mismatch pattern as f, f' (cf. Figure 2). In entropy amplification, fPAKE leverages an established cryptographic primitive called password-authenticated key exchange (PAKE) [1], which allows two parties to exchange a secure (i.e., 128-bit and uniform) key from a shared short string, such as a password, or even a bit. The PAKE protocol is secure against offline attacks, meaning that the best possible adversarial strategy is to guess the short string and engage in the key exchange. In fPAKE, PAKE is used to amplify the entropy of individual fingerprint bits as follows: Devices A and B run multiple standard PAKE [1] protocols on the individual fingerprint bits in parallel, obtaining key vectors $\mathbb{k}$ and $\mathbb{k}'$, where $\mathbb{k}_i = \mathbb{k}'_i$ if the i-th fingerprint bits matched. Next, Device A chooses a 128-bit secret s and sends a fuzzy commitment $com \longleftarrow \text{ECC.Encode}(s) \oplus \mathbb{k}$ to Device B, which decodes it with $\mathbb{k}'$. Afterwards, Devices A and B confirm to each other that they know s by sending each other hash values $H(s||0)$ and $H(s'||1)$ of the secret. Finally, if the hash check succeeds, Devices A and B derive a shared key k_{AB} from s using a key derivation function (KDF).

Advantages of fPAKE in ZIP. By using high entropy keys in the fuzzy commitment phase (cf. Figure 2), fPAKE prevents an eavesdropping adversary from mounting an offline attack because the adversary only knows $c \oplus \mathbb{k}$, which is a secure encryption of c under a (by the guarantees of PAKE) secure key $\mathbb{k}$. Moreover, even an active adversary (e.g., malicious Device B) can try *exactly one* fingerprint guess f' as input to the interactive entropy amplification phase. If that one guess is too far (i.e., f and f' are dissimilar), even the unbounded adversary cannot recover s, making the offline attack *impossible*. Otherwise, if the guess is "close enough" (cf. Section 4), the active adversary can attempt an offline attack. However, Device A waits for the key confirmation h' within a short timeout (e.g., a few seconds), allowing the adversary only this amount of time to perform the attack. We note that for standard fuzzy commitments the key confirmation upon timeout cannot similarly limit the offline attack, as the adversary does not participate in the protocol.

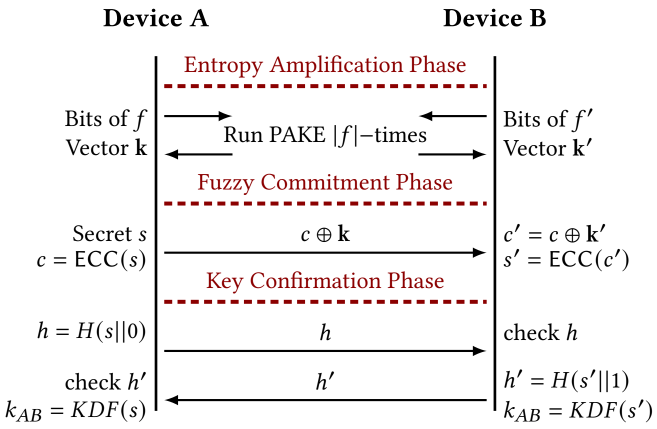


Figure 2: Detailed flow diagram of the fPAKE protocol.

In Section 4, we demonstrate how to leverage the strong security of fPAKE against offline attacks to reduce the required fingerprint sizes *well below* 128 bits in many settings. Also, we empirically show that additional communication overhead of fPAKE (i.e., entropy amplification phase) is negligible compared to up to three times faster pairing time when using fPAKE instead of fuzzy commitments.

3 SYSTEM AND THREAT MODELS

We introduce our system model, describing the goal, requirements, and assumptions of *FastZIP*, and our threat model, detailing adversary's goals and capabilities.

System Model. The main *goal* of *FastZIP* is to establish a shared secret key between colocated devices within a trusted boundary (e.g., inside a car) based on the perceived context. We design *FastZIP* to fulfill the following *requirements*: (1) be free of user interaction during pairing (*usability*), (2) have short pairing time (*practicality*), and (3) work on commodity devices equipped with off-the-shelf sensors (*deployability*). To achieve the main goal while satisfying the requirements, we make the following *assumptions*: (1) devices running *FastZIP* do not have any pre-shared secrets, nor any jointly trusted third party, (2) they communicate using Wi-Fi or Bluetooth, and share a common set of sensors such as an accelerometer, gyroscope, and barometer, and (3) they begin measuring context upon receiving the "start" command from the car's infotainment system, which is assumed to be non-compromised.

Threat Model. We consider an adversary whose *goal* is to establish a shared secret key with a legitimate device while residing outside the trusted boundary. In particular, the adversary attempts to either *impersonate* one of the legitimate devices or acts as a *man-in-the-middle* between a pair of devices. The adversary can neither

[2] The 140 bits are for an expanded encoding allowing up to 10% mismatching fingerprint bits. To allow for 30% mismatch in fingerprints, 205 bits are required.

compromise legitimate devices nor break cryptographic primitives, however, they fully control a wireless channel, are equipped with the same sensing hardware as legitimate devices, and have four attack capabilities. In an *injection attack*, the adversary attempts to pair with legitimate devices using self-chosen context readings. In a *replay attack*, the adversary replays precollected context readings. In a *similar-context attack*, the adversary tries to actively match their context with legitimate devices. In the intra-car pairing, the adversary launching a replay attack replays the precollected context data from a route driven by a victim car carrying legitimate devices, while in a similar-context attack, they actively follow the victim car to capture similar context such as the road bumpiness. The first three attacks require the adversary to participate in the pairing protocol, while in an *offline attack*, they record a successful pairing session and try to compute a shared key from it by repeatedly guessing fingerprints used by legitimate devices.

4 SYSTEM DESIGN

We present the architecture of *FastZIP*, describing its modules: *activity filter*, *quantization*, and *key exchange*.

System Overview. The main goal of *FastZIP* is to share a symmetric key between a pair of devices utilizing their context. In a moving car the context encompasses road turns, bumpiness, and speed changes [6, 34, 45], and it can be perceived by accelerometer, gyroscope, and barometer sensors that are ubiquitous in smart devices. *FastZIP* works as follows (cf. Figure 3): Devices A and B capture their context using a set of common sensors. The resulting sensor readings are input to the *activity filter* to discard low-entropy context, which can be predicted by an adversary. Afterwards, the filtered sensor readings are input to the *quantization* translating them into a sequence of fingerprint bits. Each device constructs its fingerprint by concatenating sub-fingerprints derived from different sensors (i.e., sensor fusion). These fingerprints are input to the *fPAKE* protocol, which outputs a shared symmetric key if the fingerprints have a sufficient number of similar bits.

Activity Filter. The security of any ZIP scheme relies on the unpredictability of context from outside a trusted boundary (e.g., car interior). The low-entropy context undermines security of ZIP schemes, allowing an adversary to guess fingerprints derived from it [13]. *FastZIP* utilizes the *activity filter* to ensure that fingerprints are obtained from context data with sufficient entropy.

To estimate the entropy of a sensor signal, we analyze its strength relative to noise and variation. For that, we employ three metrics: *average power*, *signal-to-noise ratio (SNR)*, and the *number of prominent peaks*, which are used to characterize signal's quality [23, 27, 45]. The average power and SNR are applicable to all sensors, while prominent peaks is a complementary metric for rapidly changing modalities (e.g., acceleration), ensuring their sufficient variation. We compute the average power P_s in dB of a discrete sensor signal $s(t)$ as follows:

$$P_{s(dB)} = 10 \cdot \log_{10}\left(\frac{1}{T}\sum_{t=1}^{T} s^2(t)\right)$$

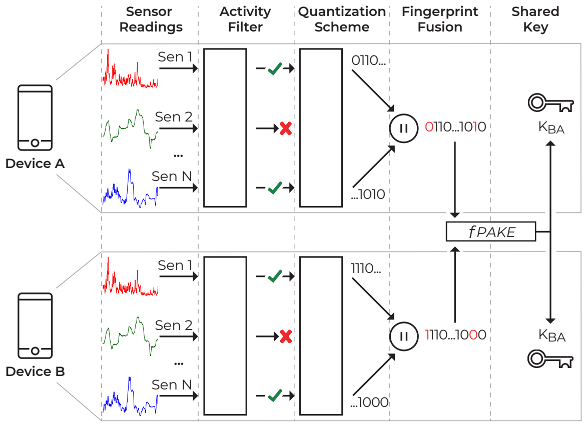


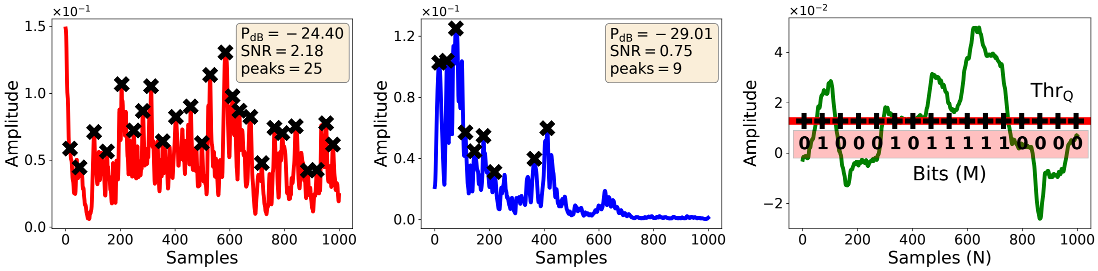
Figure 3: System overview. *FastZIP* takes as input a set of sensor readings from two devices. The readings are quantized to *similar* fingerprints and afterwards input to the fPAKE protocol to share a symmetric key, K_{BA}.

We cannot compute SNR as the ratio of signal to noise power, as we do not have the estimate of the latter; estimating noise power will impose additional processing overhead. Thus, we use an alternative definition of SNR as the ratio of mean to standard deviation of a signal: $SNR = \frac{\mu}{\sigma}$ [4]. To find prominent peaks in a signal, we count peaks that have sufficient height relative to the highest peak, while being within minimum distance Δ_P from each other. Figures 4a and 4b show activity filter metrics computed for two acceleration signals (prominent peaks marked with ✖). We see that the former signal captures continuous activity, exhibiting sufficient entropy, while the latter signal contains noise in its right half, which is reflected in the computed metrics.

After computing the metrics, we check them against fixed thresholds to discard signals with insufficient entropy, which, in turn, may reduce availability of *FastZIP*. To avoid this, we apply the activity filter on a continuous stream of sensor data using an overlapping sliding window. Thus, parts of the signal containing sufficient entropy are considered in both preceding and following timeslots, making it possible to retain them.

Quantization. *Quantization* translates a sensor signal (e.g., acceleration) to fingerprint bits used in a key agreement protocol. To ensure security, the produced fingerprints must be sufficiently unpredictable. Prior work [3, 13] finds that quantization methods of state-of-the-art ZIP schemes generate fingerprints with patterns (e.g., containing more 0-bits). We design the quantization of *FastZIP* with three goals in mind: it must (1) generate fingerprints that are random, (2) apply across various sensors, and (3) reveal minimum information about the input sensor signal. The last goal seeks to reduce adversary's knowledge about the input sensor signal leaked by quantization (e.g., signal range [16]).

We quantize short sensor signals of several seconds, producing fingerprints of a few dozen bits. However, our method generalizes to longer signals and fingerprints. The advantage of using short input signals is twofold: (1) it forces an adversary to guess context captured by a sensor within a precision of a few seconds, (2) it requires less processing, improving the runtime performance of *FastZIP*. Our quantization takes a sensor signal S of length N samples as input and outputs a fingerprint f of M bits (cf. Figure 4c). Specifically, we first find a quantization threshold Thr_Q (solid red



(a) Signal with sufficient entropy **(b) Signal with insufficient entropy** **(c) Signal quantized to bits**

Figure 4: Activity filter applied to acceleration signals (a) and (b); Quantization applied to a gyroscope signal (c).

line in Figure 4c) that splits the signal horizontally into upper and lower parts. The threshold is computed from the median of the signal, ensuring that the same number of samples lie above and below it. This way of selecting Thr_Q leads to improved randomness of the fingerprints (cf. Section 6.5) and is efficient to implement. Second, we place the quantization points $p_1, \ldots, p_M$ (marked as ✚ in Figure 4c) equidistantly onto the threshold line within distance $\Delta_Q = \lceil \frac{N}{M} \rceil + \varepsilon$ from each other, covering the signal completely. The number of quantization points and Δ_Q are public parameters customized for each sensor modality (cf. Section 6.1). Using public parameters has the advantage of (1) fewer communication rounds, as they do not need to be exchanged during the pairing protocol, and (2) not leaking information about a specific input signal, as the parameters are derived from the class of signals of the same modality, and thus are general. Third, we obtain a bit in the fingerprint $f(i)$ by comparing a signal value at the quantization point $S(p_i)$ with the quantization threshold Thr_Q:

$$f(i) = \begin{cases} 1, & S(p_i) > Thr_Q \\ 0, & \text{otherwise} \end{cases}$$

Key Exchange. The main challenge of adapting fPAKE for the use in *FastZIP* is to find which minimum fingerprint sizes[3] are required to securely exchange a 128-bit key, protecting against offline attacks. With fPAKE we can choose *arbitrarily small* fingerprints, which are then amplified to match the size of the encoded secret (cf. Figure 2). By reducing the number of required fingerprint bits, we are able to shorten pairing time, while providing sufficient security.

Before calculating the required fingerprint sizes, we determine sufficient security levels for *FastZIP*. Specifically, we consider two levels: (1) the minimal probability P with which an offline attack is eliminated and (2) the average complexity C of an offline attack. We set $P = 1 - 2^{-20}$, namely an adversary *actively participating* in a million pairing sessions can mount an offline attack in at most one of them (without even learning which one). This level of security is considered adequate for ZIP [29]. We set $C = 2^{60}$, demanding that an offline attack has an average complexity of at least 2^{60} AES decryptions. We consider this complexity sufficient given that attack time is limited to few seconds due to the key confirmation timeouts that we augment the fPAKE protocol with (cf. Figure 2).

Table 1: Offline protection and brute-force complexity for *FastZIP* computing 128-bit keys for different choices of similarity thresholds and fingerprint sizes. Gray boxes mark sufficient security levels. T is described in text.

Similarity Threshold	Fingerprint Bits	Offline Attack $1-P$	Brute-force Complexity C
95%	40	$< 2^{-23}$	$\approx 2^{37}T$
90%	60	$< 2^{-20}$	$\approx 2^{32}T$
85%	80	$< 2^{-12}$	$\approx 2^{60}T$
80%	100	$< 2^{-7}$	$\approx 2^{63}T$
75%	120	≈ 1	$\approx 2^{64}T$
70%	140	≈ 1	$\approx 2^{60}T$

We explain how to calculate the required fingerprint sizes, satisfying our chosen security levels using a 95% similarity threshold as an example. The similarity threshold defines the amount of common bits in two fingerprints needed to obtain a shared secret key. One might think to set the fingerprint size $|f|$ such that the probability of guessing at least $0.95 \cdot |f|$ bits correctly is smaller than 2^{-20}, since the key should be undecodable otherwise. Unfortunately, this is not true: an ECC (used in fPAKE) correcting 5% mismatch between the fingerprints *leaks some information about the encoded secret* until up to $2 \cdot 5\% = 10\%$ mismatch. Thus, an active adversary guessing less than 90% of the fingerprint correctly learns nothing about the secret. However, if the guessed fingerprint is "close enough" (i.e., 90–95% of the bits), the adversary cannot immediately decode the secret but obtains an *ambiguous encoding* from which the secret can be brute-forced. Taking this "security gap" inherent to ECCs into account, we set $|f|$ such that the probability $\sum_{i=m}^{n} \binom{n}{i} / 2^n$ of guessing $m = (2 \cdot Thr - 1) \cdot |f|$ out of $n = |f|$ bits correctly is smaller than 2^{-20}, where Thr is the target similarity threshold.

We note that $|f|$ goes to infinity when Thr approaches 75%, since $2 \cdot 25\% = 50\%$ of a random bitstring is easy to guess. Thus, full protection against offline attacks with probability at least $1 - 2^{-20}$ is only possible for thresholds over 75%, requiring short fingerprints of 40–60 bits for thresholds above 90% (cf. Table 1). Below 90%, the security of *FastZIP* relies on our other security level measuring brute-force complexity of the offline attack. For estimating this complexity, we

[3] Here, we assume fingerprints to be uniformly random bitstrings; entropy biases will increase fingerprint sizes (cf. Section 6.5).

think of ECC encodings as consisting of $n = |f|$ parts which are correct or wrong depending on whether the corresponding fingerprint bit was correct or not. We use the following brute-force method to decode an ambiguous encoding: randomly guess which m parts of the encoding are correct, decode only them, and set the secret to be the first result that appears twice. Considering that we do not know how many parts i of the codeword are actually correct, the conditional probability of guessing m out of i correct parts in the n parts long encoding is given by the hypergeometric distribution as $i^{\underline{m}}/n^{\underline{m}}$, which finds its maximum at $i = (n-m)/2$. The complexity of the offline attack is thus lower bounded by $n^{\underline{m}}/(n-m)^{\underline{m}}T$. Here, T is the complexity of ECC.Decode, which is larger than the complexity of one AES decryption.

Table 1 shows the calculated fingerprint sizes providing sufficient security for *FastZIP* based on fPAKE for a range of similarity thresholds. In Section 7, we elaborate that these findings are generic, thus can be directly reused by other ZIP schemes.

5 INTRA-CAR DEVICE PAIRING

We present *intra-car device pairing*—an exemplary use case of *FastZIP* to pair devices inside a moving car. It enables novel vehicular applications such as pairing user devices for customized driving experience or pairing ECUs for travel efficiency [9, 35]. We first provide the case overview followed by implementation details.

Case Overview. There is a growing number of on-board smart devices in modern cars, including devices of drivers and passengers (e.g., smartphone, earbuds) as well as ECUs and infotainment systems [9, 20]. The prohibitive user effort to pair these devices, many of which lack user interfaces, justifies the use of *FastZIP* for intra-car device pairing. *FastZIP* utilizes four sensor modalities to capture the context of a moving car: vertical and horizontal acceleration, gyroscope sky-axis, and barometer. Our review of prior work shows that acceleration of a moving car can be decomposed into *vertical* and *horizontal components*, with the former capturing road conditions (i.e., bumpiness), while the latter—driving patterns and traffic conditions (i.e., acceleration/deceleration) [6]. A *gyroscope* measures car's turns and steering directions [45], while a *barometer* captures altitude changes when a car moves along the road [34].

5.1 Implementation

Data Collection. We develop an Android app to collect accelerometer, gyroscope, and barometer data at fixed sampling rates (i.e., 100 Hz for accelerometer and gyroscope; 10 Hz for barometer). We convert accelerometer and gyroscope data to the world coordinates, eliminating the effect of device orientation. Before data collection, we perform an Network Time Protocol (NTP) update on smartphones, ensuring consistent data timestamps, which we use to synchronize the start of sensor recordings of colocated devices.

Data Processing. We process the collected sensor data before feeding it into the activity filter. Prior to any processing, we resample the data to the set sampling rates, eliminating the effect of *sampling rate instability* [38]. To decompose acceleration into vertical and horizontal components, we (1) remove the Earth's gravity from the accelerometer data applying a non-overlapping 5-second sliding window and (2) use the estimated Earth's gravity to perform the decomposition [6]. For the gyroscope data, transformed to the world coordinates by our app, we select a Z-axis that is perpendicular to the road surface. We convert the barometer data p to altitude h_{alt} in meters using a standard pressure-height formula [34]:

$$h_{alt} = 44330 \cdot \left(1 - \left(\frac{p}{1013.25}\right)^{\frac{1}{5.255}}\right)$$

After converting the sensor data to a required format, we perform signal smoothing and noise reduction in two steps: (1) applying them on the whole data and (2) on signals of several seconds, partitioning these data. To remove low-frequency noise and smooth the whole data without distorting it (e.g., keep peak locations), we use a Savitzky-Golay (SG) filter with a window length 3 and degree 2 polynomial. Afterwards, we apply a Gaussian filter with a sigma of 1.4 to reduce high-frequency noise.

We use the same sequence of filters on sensor signals of several seconds: the SG filter has a window length 5 and degree 3 polynomial for finger-grained smoothing, while the Gaussian filter stays the same. For the acceleration signals, we afterwards apply an exponentially weighted moving average (EWMA) filter to smooth them further, while keeping their significant changes; the EWMA alpha is set to 0.16 and 0.2 for vertical and horizontal acceleration, respectively. For the altitude signals, we perform mean subtraction before filtering. This helps to (1) remove offset between barometer sensors caused by hardware and temperature variation [14], (2) eliminate atmospheric pressure, accentuating altitude changes in the signal. We adapt filter parameters for signal smoothing and noise reduction from related work [6, 17, 27].

Activity Filter. The activity filter applies to a processed sensor signal of several seconds. We implement it by computing the average power and SNR for all modalities, and counting prominent peaks for vertical and horizontal acceleration (cf. Section 4). To pass the activity filter, a signal must have the average power, SNR, and optionally the number of prominent peaks higher than a predefined threshold, which we find empirically. The signal that passes the activity filter is input to the quantization, otherwise it is discarded.

Quantization. We implement quantization, converting a sensor signal to fingerprint bits, as described in Section 4. Its parameters (i.e., signal length, number of output bits) are set empirically for each modality (cf. Section 6.1). We compute the quantization threshold as a median of the sensor signal; for vertical and horizontal acceleration, we add small Δ to the median, reducing the effect of sensor noise on quantization. We concatenate bits quantized from the sensor signal with bits derived from other modalities likewise before inputting them as one fingerprint to the fPAKE protocol.

***FastZIP* Prototype.** We implement *FastZIP* to evaluate its runtime performance on off-the-shelf IoT devices (cf. Section 6.6). We focus on the fPAKE protocol, as the underlying functionality takes either constant (e.g., sensing) or negligible time (e.g., quantization). The *FastZIP* prototype allows two devices with similar fingerprints to establish a shared symmetric key. Our implementation is modular and agnostic to the fingerprint derivation, making it directly reusable by other ZIP schemes. To implement the fPAKE protocol, we use primitives from a Python cryptography library [39]. For the ECC, we utilize Shamir's secret sharing scheme in its error-correcting variant (i.e., introducing redundancy by adding more point-value pairs of the polynomial) [10]. For the PAKE component, we use the Encrypted Key Exchange (EKE) protocol [1], built as Diffie-Hellman

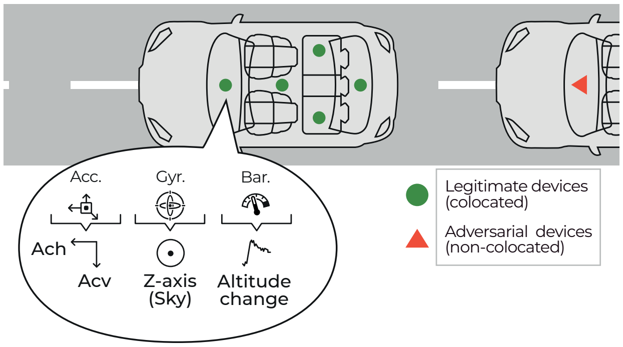

Acc.
Gyr.
Bar.
Ach
Acv
Z-axis (Sky)
Altitude change
● Legitimate devices (colocated)
▲ Adversarial devices (non-colocated)

Figure 5: Experiment setup. Smartphones are placed at five various spots inside each of two cars driving the same route.

key exchange symmetrically encrypted with passwords. Our fPAKE implementation supports two security levels, generating keys of 128- and 244-bits. We enable communication between devices utilizing IP sockets and data serialization [40, 41], allowing us to run the *FastZIP* prototype in real-time. To benchmark our implementation, we employ a Python time module [42].

6 EVALUATION

We present a comprehensive evaluation of *FastZIP* based on the real-world data we collect.

Experiment Setup. We collect accelerometer, gyroscope, and barometer data from four cars driven in a number of scenarios: within a *city*, on *country* roads, on a *highway*, and inside a *parking* garage for a total of 800 km. To evaluate the suitability of *FastZIP* for various cars, we collect these data using (1) two *similar cars* (Opel Astra wagons; 400 km of driving) and (2) two *different cars* (Škoda Octavia sedan and Volkswagen Golf hatchback; 400 km of driving). In both experiments, we equip two cars with five smartphones each, covering spots where smart devices are typically found [6, 13]: on a dashboard, between front seats, behind driver and passenger seats, and inside a trunk (cf. Figure 5). Then, we collect sensor data from two cars driven as such: (1) one car starts a predefined route, followed by another car after a 10–15 minute lag, (2) two cars drive one after another, changing the distance between each other and the role of a leading vehicle. We cover a similar number of kilometers for *city*, *country*, and *highway* driving. In the *parking* scenario, cars leave an underground garage and return back to it multiple times. To collect the sensor data, we utilize Nexus 5X and Nexus 6P smartphones. After data processing (cf. Section 5.1), we use vertical and horizontal acceleration (labeled as *Acv* and *Ach*, respectively), gyroscope sky-axis (*Gyr*), and altitude computed from barometer (*Bar*) in our evaluation.

Reproducibility and Reusability. We release the collected sensor dataset along with the driven routes map and the source code of our data collection app, evaluation stack, and *FastZIP* prototype [12].

6.1 Methodology

We evaluate *FastZIP* using several criteria: (1) security and usability, (2) pairing time, and (3) runtime performance. To assess security, we compute False Acceptance Rate (FAR) and evaluate entropy of our fingerprints. A false acceptance occurs when non-colocated devices in different cars (cf. Figure 5) pair because their fingerprints are similar enough. We assess usability by computing True Acceptance Rate (TAR), showing the rate of successful pairings between colocated devices inside the same car. For a detailed analysis of FARs and TARs, we compute them on the *full* data of an experiment (e.g., *similar cars*) and on the subsets of data corresponding to driving in one of our scenarios: *city*, *country*, *highway*, and *parking*. To evaluate pairing time, we find the amount of context data (in seconds) required to pair securely, while for runtime performance we benchmark the *FastZIP* prototype on the Raspberry Pi.

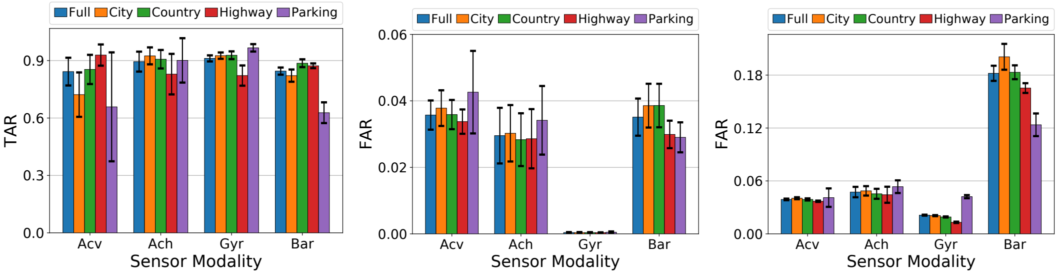
System Parameters. We use the collected sensor data to find configuration parameters for *FastZIP*'s modules: activity filter, quantization, and fPAKE yielding the best trade-off between security and short pairing time. To find the *length of sensor signal* to derive fingerprint bits, we examine how much sensor data is required to capture typical ambient activity (e.g., car turn by *Gyr*). Our results show that 10 seconds of *Acv*, *Ach*, and *Gyr* data capture typical road bumpiness, acceleration patterns, and car turns, while 20 seconds of *Bar* data is enough to record altitude changes. We set these signal lengths as input to the activity filter and to quantization, using them to empirically find thresholds for activity filter metrics for each sensor modality.

To choose the number of *fingerprint bits* output by quantization, we investigate (1) the good ratio between high TAR and low FAR and (2) modality variation. The latter helps us understand how many uncorrelated bits can be extracted from the sensor signal. Based on our findings, we set the number of fingerprint bits to 24 for both *Acv* and *Ach*, 16 for *Gyr*, and 12 for *Bar*. A *similarity threshold* defines the level of similarity between two fingerprints required to establish pairing. To select similarity thresholds, we study how many bits typically differ in the fingerprints of colocated devices. We set the following thresholds, balancing high TAR and low FAR, to be used in the fPAKE protocol: 70.8% (*Acv*), 75% (*Ach*), 93.7% (*Gyr*), and 91.7% (*Bar*).

6.2 Pairing between Colocated Devices

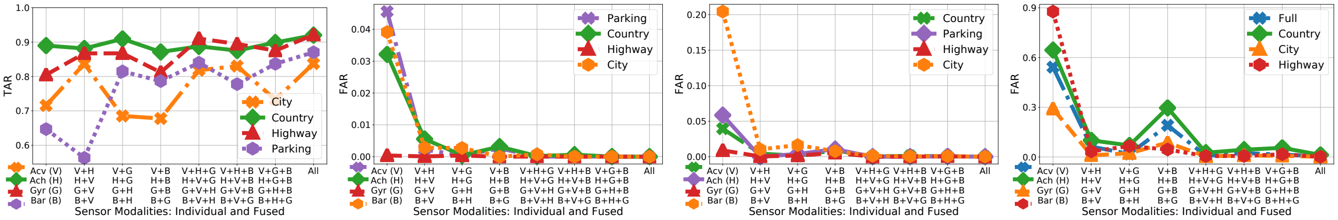
We compute TARs between each pair of colocated devices inside the same car, providing the average TAR. First, we present TARs for individual sensors (e.g., *Acv*) followed by the evaluation of sensor fusion. Our results are consistent across the *similar* and *different cars* experiments, indicating generalizability of *FastZIP* to various cars. In the following, we provide typical TARs.

TARs of Individual Sensors. Figure 6a depicts TARs for the first car in the *similar cars* experiment. We see that *full* TARs range between 0.84 and 0.91, showing that the individual sensors alone achieve relatively high success rates. However, the TARs of scenarios (e.g., *city*) have higher variation: while *Ach* and *Gyr* exhibit fairly consistent TARs, *Bar* and especially *Acv* show a wider spread of TARs. For *Acv*, the TAR spread is caused by diverse bumpiness perception inside a car affected by such factors as car suspension (e.g., front vs. rear) and surface on which bumpiness is measured (e.g., plastic vs. fabric). These factors become important when a car moves slowly, reducing TARs as in the *city* and *parking*, while higher speed leads to more profound bumpiness, increasing TARs as in the *country* and *highway*. For *Bar*, higher speed causes profound



(a) TARs (b) FARs: injection attack (c) FARs: replay attack

Figure 6: True Acceptance Rates (TARs) and False Acceptance Rates (FARs) for individual sensors.



(a) TARs (b) FARs: injection attack (c) FARs: replay attack (d) FARs: similar-context attack

Figure 7: Effect of sensor fusion on True Acceptance Rates (TARs) and False Acceptance Rates (FARs) for representative subsets of scenarios; we connect markers for readability, the plots do not show a time series.

altitude changes, improving TARs (e.g., *highway*), while there are few such changes when traveling short distances, reducing TARs (e.g., *parking*). In contrast, *Ach* and *Gyr* show lower TARs when a car moves at constant speed (e.g., *highway*). These sensors benefit from non-monotonic driving with many stops, leading to distinct acceleration patterns (*Ach*) and sharp turns (*Gyr*), as in the *city* and *parking*. Thus, no sensor outperforms the others in all the scenarios, and they show potential for complementing each other.

We analyze TAR deviation inside a car, finding that longer distance between devices leads to lower TARs. This happens because context signals (e.g., road bumpiness) can be attenuated or perceived with varying intensity at distant spots. We find that rapidly changing sensors (i.e., *Acv*, *Ach*) can have up to 20 percentage points of TAR difference between farthest devices, while for gradually changing sensors (i.e., *Gyr*, *Bar*) it is below five percentage points.

TARs with Sensor Fusion. We fuse sensors by concatenating sub-fingerprints of different modalities derived in the same timeframe. Thus, we obtain more fingerprint bits in less time, speeding up pairing. We explore the fusion of two, three, and all of our sensors. Our findings show that sensor fusion generally increases TARs, while reducing their deviation between devices. This happens because sensors can reinforce each other in the following way: error correction bits unused in the sub-fingerprint of highest similarity allow fixing extra errors in another sub-fingerprint, making the fused fingerprint exceed the similarity threshold, improving the TAR. Such reinforcing effect leads to the fused TAR to be either close to the highest TAR in sensor combination or even exceed it. The latter outcome is typical for sensor combinations including *Ach* and *Gyr*, which often capture co-occuring ambient activity (e.g., decelerate when turning).

Figure 7a shows a subset of fused TARs for the second car in the *similar cars* experiment. We see that by adding more sensors TARs steadily increase from left to right: ranging from (0.65, 0.89) for individual modalities to (0.85, 0.93) when fusing all of them. With the TAR of 0.9 colocated devices would need 1.1 pairing attempts on average to pair successfully. In few cases, sensors do not reinforce each other, namely combinations including *Acv* and *Bar*, and *Acv* and *Gyr* in the *parking* and *city*, leading to reduced TARs. For *Acv* and *Bar*, both have lowest TARs in these scenarios (cf. Figure 6a), so combining them increases the number of mismatching bits in the fused fingerprint. We find that *Acv* and *Gyr* often capture disjoint ambient activity (e.g., high speed: intense bumpiness but no turns), explaining lower potential for reinforcing each other.

6.3 Resilience to Attacks

We compute FARs between each pair of non-colocated devices in different cars, presenting the average FAR under *injection*, *replay*, and *similar-context* attacks (cf. Section 3). Similar to TAR, we first provide FARs for individual sensors and then evaluate their fusion, obtaining consistent results across *similar* and *different cars* experiments. In the following, we provide typical FARs.

Injection Attack. We use sensor data collected inside a parked car capturing noise to pair with legitimate devices. Figure 6b depicts FARs of individual sensors computed for the first car in the *similar cars* experiment. We see that three out of four modalities show FARs above 0.03, making this low-effort attack practical. However, injecting sensor noise does not work on *Gyr* because car turns result in distinct up and down peaks in the signal (cf. Figure 4c) that are not common for noise. With sensor fusion, FARs drop below half a percentage point using two modalities, converging to zero by adding more sensors (cf. Figure 7b). This result is the opposite of the reinforcing effect in TARs, showing that with more sensors differences between non-colocated fingerprints grow, reducing FARs.

We also try injecting sensor signals that are collected in a moving car but do not pass the activity filter. In this case, FARs grow by an extra percentage point for *Acv*, *Ach*, and *Bar*, while for *Gyr* they increase by order of magnitude: up to 0.005. Thus, low-entropy sensor signals from a moving car slightly improve the attack, while sensor fusion has the same effect as in Figure 7b.

Replay Attack. We replay sensor signals passing the activity filter from one car to pair with devices in another car; both cars have driven the same route. In the first case, we do not synchronize such replayed signals. Figure 6c depicts FARs of individual sensors for the first car in the *similar cars* experiment. Compared to injection attack, FARs show a fourfold increase for *Gyr* and *Bar*, remaining similar for *Acv* and *Ach*. The altitude change (*Bar*) on a given route has least variation, allowing successfully replay (i.e., FAR of up to 0.2), while other sensors are less affected. We can reach zero FARs by fusing more than two sensors (cf. Figure 7c).

In the second case, we replay sensor signals from periods when both cars drive the same part of the route (e.g., in a city) using a rough timeline of their travel. We see an extra twofold increase in FARs of *Gyr* and *Bar* peaking at 0.07 and 0.38, respectively, while for *Acv* and *Ach* the growth is 1–3 percentage points. Thus, all sensors have FAR above 0.05, making this attack alarming. The sensor fusion leads to zero FARs as in Figure 7c, showing its importance to prevent replay attacks.

Similar-context Attack. We use sensor signals passing the activity filter from one car to pair with devices in another car when two cars drive one after another (cf. Figure 5). We grant the adversary an unfair advantage of matching a single sensor (e.g., *Acv*). It means that they always "guess" the closest fingerprint to the legitimate one; the adversarial and legitimate fingerprints are derived from temporally close sensor signals. Figure 7d depicts the best achievable FARs for this attack. We see that none of individual sensors can prevent the similar-context attack alone, showing FARs between 0.3 and 0.9 (leftmost of the graph). As in the replay, *Bar* that has least variation is the most vulnerable followed by *Ach* and *Acv*. For *Ach* and *Acv*, FARs are caused by shared road conditions such speed limits leading to consistent decelerations (*Ach*) and road cracks resulting in similar bumpiness (*Acv*). *Gyr* is the most robust to this attack because it captures human-specific steering behavior, which varies between drivers.

We see that fusing two sensors cannot prevent the similar-context attack, especially when combining low-varying *Bar* with other modalities (cf. peak in the middle of Figure 7d). By adding three and more sensors, we achieve nearly zero FARs, emphasizing the necessity for sensor fusion to mitigate advanced attacks in ZIP.

6.4 Pairing Time

We compare pairing time of *FastZIP* utilizing fPAKE and state-of-the-art ZIP schemes based on fuzzy commitments. To enable a fair comparison, we assume the number of fingerprint bits and time to derive them to be the same (cf. Section 6.1) and evaluate pairing time on the level of the cryptographic protocol, namely fPAKE vs. fuzzy commitments. First, we calculate how much time it takes to obtain enough fingerprint bits to provide security against offline attacks for fPAKE and fuzzy commitments. For the former, we use findings in Table 1, while for the latter we target a 128-bit fingerprint, accounting for entropy loss due to error correction [29]:

$$|f|_{entropy\_loss} = |f|_{target} + 2 \cdot (1 - thr) \cdot |f|_{target}$$

Here, thr denotes a similarity threshold. Table 2 shows the resulting pairing times, demonstrating that *FastZIP* requires 20–40 seconds to pair in the majority of cases, while state-of-the-art schemes need 1.5–3 times longer time under the same conditions.

Second, we evaluate the time required to accumulate fingerprint bits for fPAKE and fuzzy commitments in Table 2 by traversing our collected sensor data with an overlapping sliding window using a 5-second step (cf. Activity Filter in Section 4 for reasoning). Figure 8 gives pairing times obtained on the *full* data of the first car in the *different cars* experiment, confirming the 1.5–3 faster pairing time of *FastZIP*. The calculated and obtained from our data pairing times are close to each other; the latter pairing times for *Bar* and its fusion combinations are even smaller, as the length of the *Bar* signal (i.e.,

Table 2: Calculated pairing times for *FastZIP* (fPAKE) and state-of-the-art ZIP schemes (Fuzzy commitments: F. com.).

Sensor	Sim.	Fingerprint Bits		Pairing Time (s)	
(Fusion)	Thr.	fPAKE	F. com.	fPAKE	F. com.
Acv (V)	70.8%	140	203	60	90
Ach (H)	75.0%	120	192	50	80
Gyr (G)	93.7%	50	145	40	100
Bar (B)	91.7%	60	147	100	260
V+H	72.9%	130	198	30	50
V+G	80.0%	100	180	30	50
V+B	77.8%	110	185	80	120
H+G	82.5%	90	173	30	50
H+B	80.5%	100	178	60	100
G+B	92.9%	55	147	40	120
V+H+G	78.1%	110	185	20	30
V+H+B	76.7%	120	188	40	80
V+G+B	82.7%	90	173	40	80
H+G+B	84.6%	80	168	40	80
All	80.2%	100	179	40	60

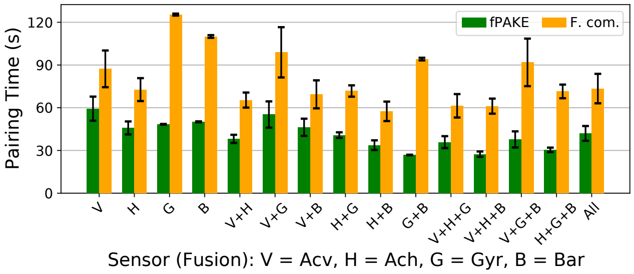
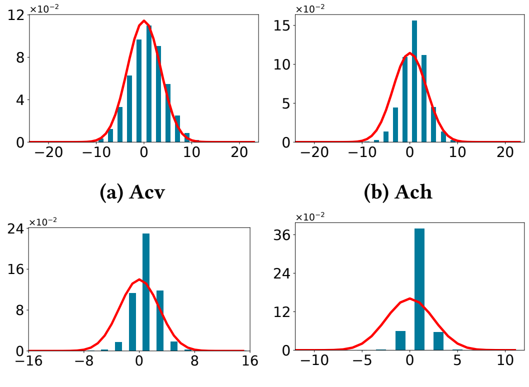


Figure 8: Pairing times obtained from our sensor data for *FastZIP* (fPAKE) and state-of-the-art ZIP schemes (Fuzzy commitments: F. com.).



(a) Acv

(b) Ach

(c) Gyr

(d) Bar

Figure 9: Distribution of fingerprint random walks for different sensors. Expected binomial distribution in red.

20 seconds) is significantly bigger than the sliding window step. We see that pairing times obtained from our data shorten in the case of profound ambient activity (e.g., *Acv* on *highway*), and pairing time consistency inside a car depends on device location for *Acv* and *Ach*, while it is stable for *Gyr* and *Bar*.

Above, we have shown how much reduction in pairing time can be achieved by using fPAKE instead of fuzzy commitments. *FastZIP* also utilizes sensor fusion as compared to a single sensor modality used by state-of-the-art ZIP schemes. The effect of sensor fusion on pairing time can be seen in Table 2 and Figure 8. The maximum reduction of pairing time is proportional to the number of used sensors (e.g., three times shorter with three sensors as compared to one), assuming each sensor obtains the same number of bits in the same amount of time. From Table 2 (cf. fPAKE pairing time), we see that for *Acv*, *Ach*, and *Gyr* (each requires 10 seconds to produce one fingerprint) the reduction in pairing time is 2–3 times when fusing all three sensors. It is smaller in some cases than the maximum possible reduction because each sensor outputs a different number of bits after error correction due to varying similarity thresholds. In our setting, sensor fusion allows *FastZIP* to shorten pairing time by an extra 2–3 times in addition to what is gained by fPAKE. Combining more sensors would further reduce pairing time.

6.5 Entropy of Fingerprints

To evaluate entropy of fingerprints produced by *FastZIP*, we (1) examine them for biases (e.g., bit patterns) and (2) estimate their min-entropy. To identify biases, we represent our fingerprints as random walks, with 1- and 0-bits showing steps in positive and negative directions [3, 13]. The result follows a binomial distribution if fingerprints are uniformly random. We also study bit transition probabilities, interpreting each bit position in a fingerprint as a state in a Markov chain. Figure 9 depicts the results of random walks for individual sensors. The distributions for all sensors are centered around the mean, indicating that overall fingerprints have the equal number of 0- and 1-bits. We see that more unique fingerprints can be generated from modalities with higher variation (e.g., *Acv*). The Markov property is close to 0.5 for all sensors, showing that the probability of each bit in a fingerprint to be 0 or 1 is equal. These findings reveal no biases in our fingerprints, indicating that our quantization achieves its design goals (cf. Section 4).

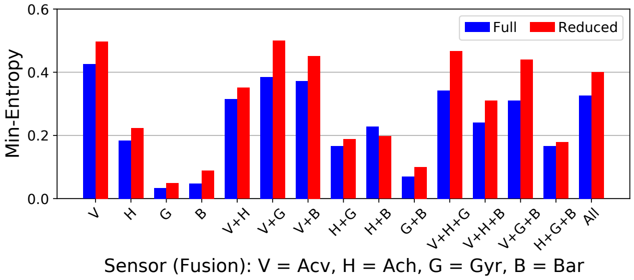


Figure 10: Min-entropy of *FastZIP* fingerprints estimated by NIST SP800-90B test suite (entropy in 1 bit).

To assess min-entropy, we apply the NIST *SP 800-90B* test suite [31, 44]. It consists of ten entropy estimators and is widely used [5, 25, 49]. Figure 10 shows the estimated min-entropy for fingerprints of individual and fused sensors. We obtain 0.43 bits of entropy for *Acv*, 0.19 bits for *Ach*, and below 0.05 bits for both *Gyr* and *Bar*, confirming our findings in Figure 9. Sensor fusion has a positive impact on min-entropy, which either stays close to the highest min-entropy in the combination or exceeds it. The fact that min-entropy increases when combining different sensors, indicates that they are uncorrelated, preventing the adversary from inferring one sensor signal from another. We consider the obtained entropy results to be conservative because the SP 800-90B suite is known to underestimate min-entropy [50], and it makes a fair assessment given $>10^6$ data samples, which we do not have. We find that dependency between consecutive bits is a decisive factor in lowering min-entropy of our fingerprints. To check if this is caused by quantization parameters, we halve the number of bits in our fingerprints (cf. *Reduced* in Figure 10), seeing only a modest increase in min-entropy. Thus, min-entropy in our fingerprints is restricted by the lack of entropy in the sensor data. In Section 7, we elaborate on attainable entropy from our sensor data.

Figure 10 shows that the majority of fused fingerprints have 30–40% of truly random bits. Thus, we need to collect more data to provide security, increasing pairing time of *FastZIP* by 2.5–3 times to 75–120 seconds. For state-of-the-art ZIP schemes, pairing time grows by 7 times, reaching several minutes, as they are more affected by non-random bits due to longer fingerprints.

6.6 Prototype Performance

We benchmark our *FastZIP* prototype on the Raspberry Pi 3 Model B, recording its performance in terms of computation and communication overhead. Specifically, we randomly sample 2000 fingerprints for each fusion combination, deploying them on two Raspberry Pis (i.e., 1000 fingerprints on each) connected via a Wi-Fi router. We measure the execution time to establish a 128-bit symmetric key on each device, showing the average performance in Figure 11[4]. We observe a maximum time of around 4.4 seconds for two sensors (i.e., *Acv* + *Ach*), growing to 8.2 seconds when fusing all of them. The

[4] We use fingerprints from our evaluation. Accounting for entropy loss in the fingerprints (cf. Table 2) will increase the execution time by a few seconds.

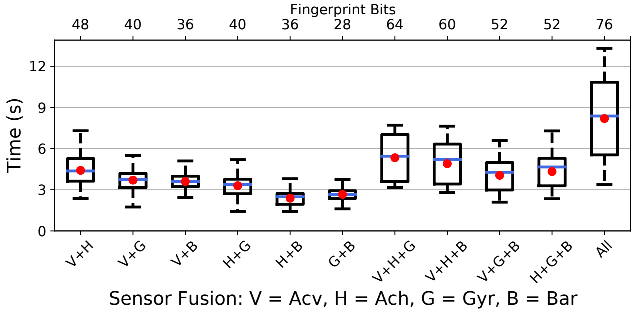


Figure 11: Performance of *FastZIP* for 128-bit key output.

execution time depends on the fingerprint size, and its deviation from the average performance increases with a lower similarity threshold (e.g., *Acv* + *Ach* vs. *Gyr* + *Bar*). We see that 60–80% of the execution time accounts for the communication overhead, which can be reduced using a direct link between devices. Rising the output key size from 128 to 244 bits proportionally increases the execution time. Overall, *FastZIP* runs efficiently on off-the-shelf IoT devices, imposing only a few seconds of overhead. Our prototype is Python-based without performance optimization techniques. Its building blocks (cf. Section 5.1) can be reimplemented in C to deploy *FastZIP* on more constrained devices.

7 DISCUSSION

We provide relevant discussion points for *FastZIP*.

Generalizability. We show how to adapt the building blocks of *FastZIP*: activity filter, quantization, and fPAKE to be used for ZIP in other use cases (e.g., smart home, wearables). Our activity filter utilizes generic metrics: average power, SNR, and number of prominent peaks that can be computed on any sensor signal. To find metrics thresholds, excluding low-entropy signals, we examine metrics of sensor signals of fixed length (e.g., 10 seconds), capturing strong and weak ambient activity. With this approach, we obtain thresholds suitable for different cars and road types. Similar results are reported for the average power threshold of audio signals recorded in different places [23]. Hence, thresholds for activity filter metrics can be determined once per use case and sensor type. Our activity filter can be easily adapted by wearable ZIP utilizing human gait captured by the acceleromter. Specifically, all three metrics (prominent peaks can mark gait cycles) are computed on gait signals of chosen length (e.g., 30 seconds), while metrics thresholds can be derived using public gait data of moving and still users [36].

FastZIP quantization has worked well on four sensor modalities. To apply it for other ZIP use cases, two parameters need to be adjusted: (1) length of input sensor signal and (2) number of output fingerprint bits. These parameters are set empirically based on the duration and variation of scenario-specific ambient activity captured by the sensor signal. For example, in smart home, a door knock event lasting a few seconds can be recorded by the microphone and accelerometer [17]. The former signal has higher variation, thus our quantization can be set to output more fingerprint bits from it.

Our fPAKE findings (cf. Table 1) are generic for a given similarity threshold and security level, hence directly reusable by other ZIP schemes. For a different choice of similar threshold/security level, the required fingerprint size in bits providing protection against offline attacks can be computed, as explained in Section 4.

Entropy of Sensor Data. The min-entropy presented in Figure 10 results from sensor data collected on high-quality flat roads, giving the lower bound of attainable entropy. We do not cover gravel, forest, or mountain roads that have profound bumpiness (*Acv*) and sharp turns (*Gyr*). Also, we do not have representative data from hectic metropolis driving, which should reveal distinct acceleration patterns (*Ach*) as well as from hilly regions with rapidly changing altitude (*Bar*). These different road and traffic conditions have high potential for increasing entropy in sensor data [18]. Another way of obtaining more entropy from sensor data is customized quantization. Our quantization focuses on (1) extracting bits from heterogeneous sensors and (2) reducing entropy biases in fingerprints, hence it may not be optimal in the amount of attainable entropy. Prior work explores various quantization methods [3, 15] some of which can be adapted to *FastZIP*.

Deployment Considerations. To deploy *FastZIP* in a real car setting a few points need to be considered. First, devices are expected to continuously sense their context before they establish pairing, eliminating the need for time synchronization [17, 29] (i.e., each device extracts fingerprints bits from common parts of context passing the activity filter). In other words, devices observe common context events (e.g., road bump) in the same timeline (i.e., similar to [17]) and can buffer them, tolerating clock offset between devices. For this to work, devices must maintain the same sampling rate of context measurements and start them simultaneously (e.g., upon a broadcasted command). For example, a major component of the car (e.g., infotainment system) can broadcast such a command when a car is started. Since devices in the same car are located nearby, they will receive this command almost at the same time. To further eliminate the effect of different devices receiving the broadcasted command at negligibly different times and account for overhead to trigger sensing, devices can start measuring context upon the command reception after a short pause (e.g., 5 seconds); for this they do not need synchronized clocks as well. Furthermore, *FastZIP* extracts much fewer bits from context signals (e.g., 24 bits from 10 seconds) as compared to existing ZIP schemes (e.g., 128 bits from 5 seconds in [48] or 512 bits from 6 seconds in [37]), making *FastZIP* less susceptible to several millisecond offsets between these signals. Specifically, we try injecting 5–7 millisecond offsets between context signals that we used to evaluate *FastZIP*, finding that it would reduce TARs of individual sensors (cf. Figure 6a) by maximum 10% for *Acv*, 7% for *Ach*, and below 5% for *Gyr* and *Bar*, while FARs remain the same. We consider this reduction to be acceptable for a proof-of-concept *FastZIP*, however, further research can investigate how to eliminate the effect of synchronization errors in real deployments. Since *FastZIP* requires a few dozen seconds to pair, collecting context for this time using low-power sensors will not impose much overhead. Second, each device is expected to learn parameters of the scheme (e.g., quantization) prior to pairing: in *FastZIP* it can happen upon the scheme installation, as commonly assumed in ZIP [17, 36, 47]. Before trying to pair, each device can advertise its desired security level (e.g., 128- or 244-bits in fPAKE), pairing with those devices that support the same security level.

Limitations. We evaluate *FastZIP* using devices fixed inside a car

Table 3: Comparison with state-of-the-art ZIP schemes.

Scheme	Use Case	Time (s)	(FAR, FRR)	Bias
Schürm. & Sigg [37]†	In-car	120	(0.10, 0.10)	Low
Miettinen et al. [28]†	In-car	1280	(0.23, 0.23)	High
Convoy [18]	In-car	300	-	-
Miettinen et al. [29]	Home	5640	(0.03, 0.02)	-
Perceptio [17]	Home	8280	-	-
BANDANA [36]	Wearables	96	-	High
FastZIP	In-car	20	(0.0, 0.06)	Low

† evaluated in [13]. We show best achievable results for each scheme.

interior, covering the likely use case of pairing between a mounted user device (e.g., smartphone) and an infotainment system. However, users may interact with their devices, affecting accelerometer and gyroscope readings. Differentiating between human and vehicle motion in the sensor data collected inside a moving car is an open research question [6]. We envision that predicting sensor data resulted from human motion [47] and filtering it afterwards [36] can help address this question.

8 RELATED WORK

To date, a number of ZIP schemes utilizing various sensors (e.g., microphone, accelerometer) to capture context have been proposed [17, 18, 27–29, 36, 37]. The state-of-the-art ZIP schemes rely on the fuzzy commitments cryptographic primitive [22] to establish a shared secret key. Other cryptographic alternatives include customized extensions of fuzzy commitments [36] or the EKE protocol [15]. However, these extensions do not have proven security guarantees. The majority of proposed ZIP schemes rely on a single common sensor to capture context. The existing schemes utilizing fuzzy commitments and context based on a single sensor modality suffer from (1) prolonged pairing time, (2) vulnerability to offline attacks, and (3) attacks caused by the predictable context (e.g., replay). *FastZIP* overcomes these limitations by a novel design, namely combining the fPAKE protocol [10] and multi-sensor context constructed by combining multiple sensor modalities (i.e., sensor fusion).

Table 3 compares *FastZIP* and prominent state-of-the-art ZIP schemes in terms of pairing time, error rates, and entropy biases in the fingerprints. We note that this comparison is indicative, as we use the information reported in the original publication for each ZIP scheme. *FastZIP* has the shortest pairing time among the schemes, including those that are used for in-car pairing, while achieving low error rates. This shortest pairing time is due to the combination of fPAKE and sensor fusion, which can together give a 3–9 reduction in pairing time (cf. Section 6.4). However, pairing time also highly depends on the used context (e.g., continuous gait [36] vs. infrequent knock [17]) and quantization method (e.g., in [28] one bit is derived from two minutes of sensor data.)

The schemes [37] and [28] utilizing ambient audio and noise levels, respectively, are evaluated for in-car pairing [13], showing error rates above 0.1. Despite audio and noise level context varying significantly in a running car, the fingerprints of those schemes contain entropy biases (e.g., more 0-bits). *Convoy* that uses road bumpiness captured by the accelerometer for pairing is vulnerable to the context replay attack [18], however the resulting FAR is not reported. A similar work bears the same weakness as *Convoy* but does not state the pairing time [24]. ZIP schemes for pairing smart home devices [17, 29] may achieve comparable error rates to *FastZIP*, requiring, however, at least two orders of magnitude longer time. This time will further increase in the case of entropy biases, which are not evaluated by the considered schemes. We note that the longest pairing time of *Perceptio* [17] is a tradeoff, as the scheme enables pairing between devices with heterogeneous sensors (e.g., microphone and accelerometer). For ZIP schemes targeting wearables such as BANDANA [36], utilizing human gait captured by the accelerometer, the pairing time is closest to *FastZIP*. However, such schemes often show bit patterns in their fingerprints and are vulnerable to video-based attacks [3].

Our review of related ZIP work reveals important results: entropy biases of various level of severity exist in fingerprints of all schemes. This is worrying, as the state of the art relies on fuzzy commitments, where high entropy of fingerprints is imperative to prevent offline attacks. Also, none of the works explicitly accounts for entropy biases (e.g., by saying how many more bits need to be collected). The impact of entropy biases is less severe in fPAKE, as it limits the offline attack in time and number of attempts. We notice that many ZIP schemes use previous versions of NIST statistical tests [33] to find entropy biases, reporting results for only passed tests, without further investigation [26, 27, 48]. Thus, we urge researchers to scrutinize the entropy of fingerprints derived from context with recent NIST tests [44] and additional tools such as in [3, 13].

9 CONCLUSION

In the age of the Internet of Things (IoT) securing wireless communication of smart devices is crucial to protect their data. Zero-interaction pairing (ZIP) allows establishing a shared secret key between devices based on their physical context (e.g., ambient audio). We propose *FastZIP*, a novel ZIP scheme that significantly reduces pairing time, while providing stronger security than state-of-the-art ZIP schemes. The main contribution of *FastZIP* is its innovative design combining the Fuzzy Password-Authenticated Key Exchange (fPAKE) protocol and sensor fusion. We implement and empirically evaluate *FastZIP* in the exemplary use case of intra-car device pairing, demonstrating that *FastZIP* (1) reliably pairs devices inside the same car, achieving up to three times faster pairing than state-of-the-art ZIP schemes, (2) is secure against various attacks, and (3) runs efficiently on off-the-shelf IoT devices.

ACKNOWLEDGMENTS

We thank our shepherd and our anonymous reviewers for their insightful comments that helped to improve this paper. We also thank Max Maass and Arne Brüsch for their assistance in conducting this research. This work has been co-funded by the Research Council of Norway as part of the project Parrot (311197) as well as the German Federal Ministry of Education and Research and the Hessian Ministry of Higher Education, Research, Science and the Arts within their joint support of the National Research Center for Applied Cybersecurity ATHENE. Jun Han and Julia Hesse are co-corresponding authors of this work.

REFERENCES

[1] S. M. Bellovin and M. Merritt. Encrypted key exchange: password-based protocols secure against dictionary attacks. In Proceedings 1992 IEEE Computer Society Symposium on Research in Security and Privacy, pages 72–84, 1992.

[2] BlueKrypt. Cryptographic Key Length Recommendation, 2020. https://www.keylength.com/en/4/.

[3] A. Brüsch, N. Nguyen, D. Schürmann, S. Sigg, and L. Wolf. Security properties of gait for mobile device pairing. IEEE Transactions on Mobile Computing, 19(3):697–710, 2019.

[4] J. T. Bushberg and J. M. Boone. The essential physics of medical imaging. Lippincott Williams & Wilkins, 2011.

[5] C. Camara, H. Martín, P. Peris-Lopez, and M. Aldalaien. Design and Analysis of a True Random Number Generator Based on GSR Signals for Body Sensor Networks. Sensors, 19(9):2033, 2019.

[6] K.-Y. Chen, R. C. Shah, J. Huang, and L. Nachman. Mago: Mode of Transport Inference Using the Hall-Effect Magnetic Sensor and Accelerometer. Proceedings of the ACM on Interactive, Mobile, Wearable and Ubiquitous Technologies, 1(2):8, 2017.

[7] K.-T. Cho, Y. Kim, and K. G. Shin. Who killed my parked car? arXiv preprint arXiv:1801.07741, 2018.

[8] M. K. Chong, R. Mayrhofer, and H. Gellersen. A survey of user interaction for spontaneous device association. ACM Computing Surveys (CSUR), 47(1):8, 2014.

[9] T. Claburn. Newsflash: Car cyber-security still sucks, 2018. https://www.theregister.co.uk/2018/01/26/car_hacking_wireless/.

[10] P.-A. Dupont, J. Hesse, D. Pointcheval, L. Reyzin, and S. Yakoubov. Fuzzy password-authenticated key exchange. In Annual International Conference on the Theory and Applications of Cryptographic Techniques, pages 393–424. Springer, 2018.

[11] M. Fomichev, F. Álvarez, D. Steinmetzer, P. Gardner-Stephen, and M. Hollick. Survey and Systematization of Secure Device Pairing. IEEE Communications Surveys Tutorials, 20(1):517–550, 2018.

[12] M. Fomichev, J. Hesse, L. Almon, T. Lippert, J. Han, and M. Hollick. Index of Supplementary Files from "FastZIP: Faster and More Secure Zero-Interaction Pairing", 2021. https://doi.org/10.5281/zenodo.4777836.

[13] M. Fomichev, M. Maass, L. Almon, A. Molina, and M. Hollick. Perils of Zero-Interaction Security in the Internet of Things. Proceedings of the ACM on Interactive, Mobile, Wearable and Ubiquitous Technologies, 3(1):10, 2019.

[14] M. Fomichev, M. Maass, and M. Hollick. Zero-interaction security—towards sound experimental validation. GetMobile: Mobile Computing and Communications, 23(2):16–21, 2019.

[15] B. Groza, A. Berdich, C. Jichici, and R. Mayrhofer. Secure Accelerometer-Based Pairing of Mobile Devices in Multi-Modal Transport. IEEE Access, 8:9246–9259, 2020.

[16] B. Groza and R. Mayrhofer. SAPHE: simple accelerometer based wireless pairing with heuristic trees. In Proceedings of the 10th International Conference on Advances in Mobile Computing & Multimedia, pages 161–168. ACM, 2012.

[17] J. Han, A. J. Chung, M. K. Sinha, M. Harishankar, S. Pan, H. Y. Noh, P. Zhang, and P. Tague. Do you feel what I hear? Enabling autonomous IoT device pairing using different sensor types. In 2018 IEEE Symposium on Security and Privacy (SP), pages 836–852. IEEE, 2018.

[18] J. Han, M. Harishankar, X. Wang, A. J. Chung, and P. Tague. Convoy: Physical context verification for vehicle platoon admission. In Proceedings of the 18th International Workshop on Mobile Computing Systems and Applications, pages 73–78. ACM, 2017.

[19] Hertz Blog. Connected Cars: Improve Your Driving Experience with the Internet of Things (IoT), 2020. https://www.hertz.com/blog/automotive/improve-your-driving-experience-with-internet-of-things.

[20] Ironpaper. Smart Car Statistics – The Increasingly Digital Experience of the Connected Vehicle, 2018. https://www.ironpaper.com/webintel/articles/smart-car-statistics-the-increasingly-digital-experience-of-the-connected-vehicle/.

[21] A. Juels and M. Sudan. A fuzzy vault scheme. Designs, Codes and Cryptography, 38(2):237–257, 2006.

[22] A. Juels and M. Wattenberg. A fuzzy commitment scheme. In Proceedings of the 6th ACM conference on Computer and communications security, pages 28–36, 1999.

[23] N. Karapanos, C. Marforio, C. Soriente, and S. Capkun. Sound-Proof: Usable Two-Factor Authentication Based on Ambient Sound. In USENIX Security Symposium, pages 483–498, 2015.

[24] Y. S. Kim. Secure and Safe In-Vehicle Device Pairing Using Accelerometer Sensor. In 2019 IEEE 90th Vehicular Technology Conference (VTC2019-Fall), pages 1–2. IEEE, 2019.

[25] D. Kreiser, Z. Dyka, S. Kornemann, C. Wittke, I. Kabin, O. Stecklina, and P. Langendörfer. On Wireless Channel Parameters for Key Generation in Industrial Environments. IEEE Access, 6:79010–79025, 2018.

[26] K. Lee, N. Klingensmith, S. Banerjee, and Y. Kim. VoltKey: Continuous Secret Key Generation Based on Power Line Noise for Zero-Involvement Pairing and Authentication. Proceedings of the ACM on Interactive, Mobile, Wearable and Ubiquitous Technologies, 3(3):93, 2019.

[27] Q. Lin, W. Xu, J. Liu, A. Khamis, W. Hu, M. Hassan, and A. Seneviratne. H2B: heartbeat-based secret key generation using piezo vibration sensors. In Proceedings of the 18th International Conference on Information Processing in Sensor Networks, pages 265–276. ACM, 2019.

[28] M. Miettinen, N. Asokan, T. D. Nguyen, A.-R. Sadeghi, and M. Sobhani. Context-based Zero-Interaction Pairing and Key Evolution for Advanced Personal Devices. In ACM Conference on Computer and Communications Security (CCS), pages 880–891. ACM, 2014.

[29] M. Miettinen, T. D. Nguyen, A.-R. Sadeghi, and N. Asokan. Revisiting context-based authentication in IoT. In Proceedings of the 55th Annual Design Automation Conference, pages 1–6, 2018.

[30] S. Mirzadeh, H. Cruickshank, and R. Tafazolli. Secure device pairing: A survey. IEEE Communications Surveys & Tutorials, 16(1):17–40, 2014.

[31] National Institute of Standards and Technology. EntropyAssessment, 2019. https://github.com/usnistgov/SP800-90B_EntropyAssessment.

[32] J. Ruge, J. Classen, F. Gringoli, and M. Hollick. Frankenstein: Advanced Wireless Fuzzing to Exploit New Bluetooth Escalation Targets. In 29th {USENIX} Security Symposium ({USENIX} Security 20), pages 19–36, 2020.

[33] A. Rukhin, J. Soto, J. Nechvatal, M. Smid, and E. Barker. A statistical test suite for random and pseudorandom number generators for cryptographic applications. Technical report, Booz-allen and hamilton inc mclean va, 2001.

[34] K. Sankaran, M. Zhu, X. F. Guo, A. L. Ananda, M. C. Chan, and L.-S. Peh. Using mobile phone barometer for low-power transportation context detection. In Proceedings of the 12th ACM Conference on Embedded Network Sensor Systems, pages 191–205. ACM, 2014.

[35] A. Sanz. IoT Connected car 3: Seat and Volvo Customer Experience, 2016. http://blogs.icemd.com/blog-iot-and-digital-marketing/iot-connected-car-seat-volvo-customer-experience/.

[36] D. Schürmann, A. Brüsch, S. Sigg, and L. Wolf. BANDANA—Body area network device-to-device authentication using natural gAit. In 2017 IEEE International Conference on Pervasive Computing and Communications (PerCom), pages 190–196. IEEE, 2017.

[37] D. Schürmann and S. Sigg. Secure Communication Based on Ambient Audio. IEEE Transactions on mobile computing, 12:358–370, 2013.

[38] A. Stisen, H. Blunck, S. Bhattacharya, T. S. Prentow, M. B. Kjærgaard, A. Dey, T. Sonne, and M. M. Jensen. Smart devices are different: Assessing and mitigatingmobile sensing heterogeneities for activity recognition. In Proceedings of the 13th ACM Conference on Embedded Networked Sensor Systems, pages 127–140. ACM, 2015.

[39] The pyca/cryptography team. pyca/cryptography, 2020. https://github.com/pyca/cryptography.

[40] The Python Software Foundation. pickle — Python object serialization, 2020. https://docs.python.org/3/library/pickle.html?highlight=pickle.

[41] The Python Software Foundation. socket — Low-level networking interface, 2020. https://docs.python.org/3/library/socket.html?highlight=socket.

[42] The Python Software Foundation. time — Time access and conversions, 2020. https://docs.python.org/3/library/time.html#module-time.

[43] S. A. Trikutam. Driving the Connected Car Revolution, 2019. https://www.cypress.com/blog/corporate/driving-connected-car-revolution.

[44] M. S. Turan, E. Barker, J. Kelsey, K. A. McKay, M. L. Baish, and M. Boyle. Recommendation for the entropy sources used for random bit generation. NIST Special Publication, 800(90B), 2018.

[45] C. Vaas, M. Juuti, N. Asokan, and I. Martinovic. Get in line: Ongoing co-presence verification of a vehicle formation based on driving trajectories. In 2018 IEEE European Symposium on Security and Privacy (EuroS&P), pages 199–213. IEEE, 2018.

[46] G. Wetzstein. Inertial Measurement Units I, 2019. https://stanford.edu/class/ee267/lectures/lecture9.pdf .

[47] Y. Wu, Q. Lin, H. Jia, M. Hassan, and W. Hu. Auto-Key: Using Autoencoder to Speed Up Gait-based Key Generation in Body Area Networks. Proceedings of the ACM on Interactive, Mobile, Wearable and Ubiquitous Technologies, 4(1):1–23, 2020.

[48] W. Xu, G. Revadigar, C. Luo, N. Bergmann, and W. Hu. Walkie-talkie: Motion-assisted automatic key generation for secure on-body device communication. In 2016 15th ACM/IEEE International Conference on Information Processing in Sensor Networks (IPSN), pages 1–12. IEEE, 2016.

[49] C. T. Zenger, M. Pietersz, J. Zimmer, J.-F. Posielek, T. Lenze, and C. Paar. Authenticated key establishment for low-resource devices exploiting correlated random channels. Computer Networks, 109:105–123, 2016.

[50] S. Zhu, Y. Ma, T. Chen, J. Lin, and J. Jing. Analysis and improvement of entropy estimators in NIST SP 800-90B for non-IID entropy sources. IACR Transactions on Symmetric Cryptology, pages 151–168, 2017.