Fingerprinting Smartphones From Embedded Transducers

A Thesis Submitted for obtaining
the Scientific Title of PhD in Engineering
from
Politehnica University Timișoara
in the Field of
Computer and Information Technology
by

Ing. Adriana-Maria BERDICH

PhD Committee Chair:	Prof. Univ. Dr. Ing. Marius-George MARCU
PhD Supervisor:	Prof. Univ. Dr. Ing. Bogdan GROZA
Scientific Reviewers:	Prof. Univ. Dr. Dana PETCU
	Prof. Univ. Dr. Ing. Alin-Dumitru SUCIU
	Prof. Univ. Dr. Ing. Gheorghe-Daniel ANDREESCU

Date of the PhD Thesis Defense: 21-June-2023

2

Acknowledgement

This thesis has been elaborated during my activity at the Faculty of Automation and Computing of the Politehnica University Timişoara, Romania.

I express special thanks to my PhD supervisor, Prof. Dr. Ing. Bogdan Groza, for supporting my research activities during my PhD study. I want to thank him for his calmness, for always motivating me during my research, for always being available in times of need and for sharing his knowledge in mobile security and other interesting topics, such as automotive security. His skills made me much more competitive and ambitious and gave me more motivation to work and publish more research articles.

I am also grateful to Prof. René Mayrhofer from Johannes Kepler University Linz for supporting my research works with his extensive experience on mobile systems security. I would also like to thank Prof. Yuval Elovici, Prof. Asaf Shabtai and Efrat Levy from Ben-Gurion University for their valuable experience in machine learning.

Timişoara, February 2023 Adriana-Maria BERDICH

Berdich, Adriana-Maria

Fingerprinting Smartphones From Embedded Transducers

Keywords:
smartphone fingerprinting, accelerometer, loudspeaker, microphone, camera, in-vehicle infotainment unit

Abstract:
Due to the significant privacy risks that smartphones present and due to their importance in device authentication and forensics investigations, fingerprinting smartphones have become increasingly popular. This thesis is focused on accelerometers, loudspeakers, microphones and camera sensors as potential fingerprint sources for smartphone embedded transducers. While there is little user knowledge regarding the privacy dangers, the output of these transducers, which convert one form of energy into another, leaks across numerous channels, like social networks, mobile apps and cloud services. Several signal processing techniques are used to extract characteristics and various traditional machine-learning algorithms are employed to fingerprint different and identical sensors. This thesis also proposes a system for device pairing based on accelerometer data collected from several transportation modes, i.e., tram, train, car, bike, walk and shake. In addition to smartphone sensors fingerprinting, ECU (Electronic Control Unit) fingerprinting is discussed as an extension.

Contents

Chapter 1

Introduction

1.1 Motivation

The number of smartphone users worldwide is expected to increase from 6 billion in 2021 to 7.6 billion in 2027 according to recent industry reports [1]. Smartphones are used for many daily tasks, including social networking, tracking user activity, bank transfers, NFC payments, controlling smart household appliances, etc. A newly emerged area where smartphones are increasingly used is the automotive industry which is undergoing a fast evolution. The traditional mechanical vehicles from several decades ago are transformed into electronic devices that can communicate with other objects, e.g., vehicles, smart parking areas, traffic lights, etc., and are equipped with several sensors, e.g., cameras, microphones, accelerometers, etc. In parallel to these, several car manufacturers are proposing the use of smartphones as car keys. Therefore, secure smartphone authentication toward a vehicle, without any human interaction, is a topic of high interest for secure vehicle-to-smartphone (V2S) communications. In this context, many of the experiments in this thesis are performed in an automotive context with the help of an infotainment unit. Figure 3.1 depicts a car with an infotainment unit and a smartphone that communicates with it.

Nowadays smartphones have a staggering amount of processing and memory capabilities. They are also equipped with various sensors, including loudspeakers, microphones, accelerometers, magnetometers and radio frequency sensors (such as NFC, UWB and GPS). These are generally referred to as transducers, components that convert one form of energy into another. Due to the manufacturing process, each transducer has unique properties that have the potential to be used as a fingerprint for the mobile device. Fingerprints based on software can also be used, but in this thesis, the focus is on hardware-based fingerprints. This is because they rely on the characteristics of transducers, which are embedded in the circuit board and are more challenging to replace. This makes the fingerprint more difficult to falsify.

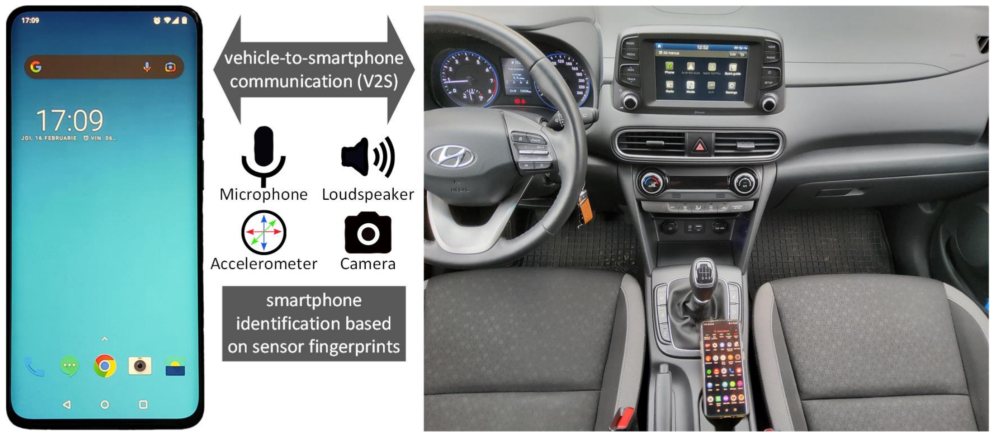

vehicle-to-smartphone communication (V2S)
Microphone
Loudspeaker
Accelerometer
Camera
smartphone identification based on sensor fingerprints

Figure 1.1: The smartphone-vehicle ecosystem

In Figure 1.2 a generic smartphone is illustrated with the sensors and transducers that can be subject to fingerprinting. Since the beginning of the 2000s, circuit identification using physical characteristics has been studied [2]. Later, Physically Unclonable Functions (PUFs), based on distinctive and erratic properties of the circuits, were proposed in [3] for security applications such as device authentication. Device-to-device (D2D) authentication is common in IoT scenarios. Using the characteristics of the device to ensure authentication is one way to eliminate user interaction, which is especially beneficial for embedded devices that lack user interfaces or inside vehicles where accessibility to the interface may be limited.

Fingerprinting smartphones is a topic that has triggered a lot of interest in the last decade, as proved by the high amount of work recalled in several surveys published throughout the years. In 2015, the authors in [4] surveyed several works that discuss smartphone fingerprinting based on the transport layer, IP and ICMP packets, application layer, browser and mobile apps. Two years later, in 2017, the authors in [5] discuss smartphone fingerprinting based on physical characteristics. Several fingerprinting techniques are presented based on the signals transmitted by smartphones, i.e., clock offset, Medium Access Control (MAC) and radio frequency physical. The authors also discuss smartphone fingerprinting based on their sensors, i.e., magnetometer, microphone and camera. A survey that depicts the algorithms used in smartphone fingerprinting was published in 2017 [6]. In 2019 the authors in [7] published a short study that discusses smartphone classification based on their cameras, accelerometers, loudspeakers and wireless transmitters. Another study on smartphone identification based on the serial number, IMEI, MAC, internal circuits, sensors and PUFs (Physical Unclonable Function) was published in 2020 in [8]. Two years ago, in 2021, a study focused on IoT (Internet of Things) device fingerprinting was published [9].

In the same context, as expected, user privacy is becoming increasingly important and the fingerprints can have a significant impact on this by leaking the identity of the

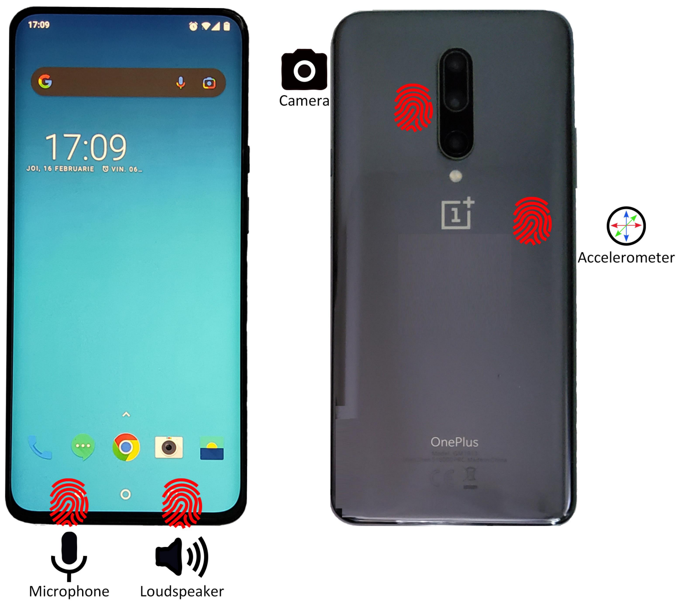


Figure 1.2: A generic smartphone with sensors and transducers subject to fingerprinting

device [10] and user localization [11]. One privacy attack recently explored in literature is an eavesdropping attack which consists in the reconstruction of the sounds played by the smartphone loudspeaker using the accelerometer sensor [12, 13, 14]. Data collected from accelerometers can also be used to detect activities in metro stations [15], track passengers in the metro [16] and detect the user's walking direction [17]. User activity recognition based on accelerometer and gyroscope data [18], or based on multiple sensors, e.g., accelerometer, gyroscope, magnetometer, barometer, proximity, humidity sensors, etc., were also discussed in the literature [19].

1.2 Research objectives

This thesis aims to fingerprint smartphones based on their sensors, i.e., accelerometers, loudspeakers, microphones and camera sensors and also briefly investigate such fingerprinting techniques for in-vehicle ECU. More specifically, the main objectives of this thesis can be summarized as follows:

1. Surveying the existing literature that addresses smartphone fingerprinting based on embedded sensors;

2. Collecting data from accelerometers, loudspeakers, microphones and camera sensors of different and identical smartphones to create comprehensive datasets;

3. Analyzing the collected data and finding the more reliable characteristics;

4. Fingerprinting smartphones based on accelerometer, loudspeakers, microphones and camera sensor characteristics which are the main four transducers used inside modern smartphones;

5. Analyze distinct classification algorithms and show that traditional machine learning algorithms may have better results than neural network algorithms;

6. Analyze and test how fingerprinting smartphone techniques can be extended to other components, using in-vehicle ECUs as an example.

1.3 Major contributions

In this thesis, several smartphone transducers, i.e., accelerometers, loudspeakers, microphones and camera sensors are fingerprinted. In addition to smartphone sensors, ECU fingerprinting is also analyzed. The contributions of this thesis can be summarized as follows:

1. Several comprehensive datasets were built containing:

    - Accelerometer data collected in different transportation modes: tram, train, car, bike, walk and shake [20];
    - 3,000 samples collected with 28 smartphones loudspeakers [21];
    - 19,200 samples collected with 32 smartphones microphones [22];
    - 300 dark photos collected with 6 identical smartphone cameras [23];

2. Several classification algorithms were used and their performance was analyzed in various scenarios, also using some signal processing techniques when needed [21], [22], [23], [24], [25];

3. Identification of smartphones from identical and different models of transducers (accelerometers, loudspeakers, microphones and cameras) was performed [21], [22], [23], [24];

4. Sensor identification in the presence of different types and levels of noise (additive white Gaussian noise or environmental noise) was performed [21], [22];

5. Device-to-device and in-vehicle authentication scenarios were addressed as applications for smartphone identification [20].

These major contributions are reflected by the following publications in relevant ISI journals and conferences. In [20] the author explored smartphone pairing based on accelerometer data collected from different transportation environments. For this, several accelerometer measurements were collected using smartphones in a train, tram, car and bike and later analyzed for the design of the protocol. Smartphone fingerprinting based on accelerometer data was analyzed in [24]. Experiments with 5 identical and 5 different smartphones were done in order to fingerprint them based on characteristics extracted from the accelerometer. In [21] smartphone fingerprinting based on loudspeaker characteristics is addressed. A dataset was built, containing records from 16 identical and 12 different smartphones, that play a linear sweep signal and it publicly released to serve for future works. Smartphones were identified based on the roll-off characteristics of the emitted sounds. Also, recurrent neural networks were used for a more accurate classification. In [22], microphone fingerprinting is addressed. A dataset was built, containing experiments with 16 identical smartphones that record locomotive, barrier, horn and tier sounds played by a high-fidelity audio system. The dataset also contains live recordings of a car honk, hazard lights and wiper sounds recorded with 16 different smartphones. The power spectrum of each signal was extracted from the recorded sounds and used as input for several machine learning classifiers to separate the smartphones. This dataset was also made public to serve for future investigations. Smartphone fingerprinting based on camera characteristics was discussed by the author in [23]. The characteristics extracted from 50 images collected using 6 identical smartphones were used as input for several classification algorithms in order to fingerprint the smartphones. The machine learning algorithms used for smartphone identification in the previously mentioned papers were also used in [26] to fingerprint in-vehicle ECUs based on an existing dataset. The author also contributed to other research papers focused on vehicle-to-smartphone interaction which, although they are not part of the main body of this work, provided a great opportunity for the author to gain even more insights into the security of the smartphone-vehicle ecosystem. These works discuss car to smartphone interaction [27], vehicle access rights based on cloud services [28], smartphone based access to vehicles [29] and audio-visual key exchange between smartphone and vehicle [30].

To sum up, the author has contributed to 11 papers on mobile system security and their presence within the in-vehicle environment, out of which the first 7 form the main body of the current thesis:

1. A. Berdich, B. Groza, R. Mayrhofer, E. Levy, A. Shabtai, and Y. Elovici, “Sweep-to-unlock: Fingerprinting smartphones based on loudspeaker roll- off characteristics,” *IEEE Transactions on Mobile Computing*, 2021.

2. B. Groza, A. Berdich, C. Jichici, and R. Mayrhofer, “Secure accelerometer based pairing of mobile devices in multi-modal transport,” *IEEE Access*, vol. 8, pp. 9246–9259, 2020.

3. A. Berdich, B. Groza, E. Levy, A. Shabtai, Y. Elovici, and R. Mayrhofer, "Fin-

gerprinting smartphones based on microphone characteristics from environment affected recordings," *IEEE Access*, vol. 10, pp. 122 399–122 413, 2022.

4. A. Berdich and B. Groza, "Smartphone camera identification from low-mid frequency dct coefficients of dark images," *Entropy*, vol. 24, no. 8, p. 1158,x 2022.

5. S. Murvay, A. Berdich, and B. Groza, "Physical layer intrusion detection and localization on CAN bus," Machine Learning and Optimization Techniques for Automotive Cyber-Physical Systems, *Springer*, 2023, (**accepted for publication**).

6. A. Berdich, B. Groza, and R. Mayrhofer, "A survey on fingerprinting technologies for smartphones based on embedded transducers," (**under submission**).

7. A. Berdich, P. Iosif, C. Burlacu, A. Anistoroaei, and B. Groza, "Fingerprinting smartphone accelerometers with traditional classifiers and deep learning networks," *IEEE 17th International Symposium on Applied Computational Intelligence and Informatics (SACI)*, 2023, (**accepted for publication**).

8. B. Groza, H. Gurban, L. Popa, A. Berdich, and S. Murvay, "Car-to- smartphone interactions: Experimental setup, risk analysis and security technologies," in *5th International Workshop on Critical Automotive Applications: Robustness & Safety*, 2019.

9. A. Berdich, A. Anistoroaei, B. Groza, H. Gurban, S. Murvay, and D. Iercan, "Antares-anonymous transfer of vehicle access rights from external cloud services," in *2020 IEEE 91st Vehicular Technology Conference (VTC2020-Spring)*, IEEE, 2020, pp. 1–5.

10. B. Groza, T. Andreica, A. Berdich, P. Murvay, and E. H. Gurban, "Prestvo: Privacy enabled smartphone based access to vehicle on-board units," *IEEE Access*, vol. 8, pp. 119 105–119 122, 2020.

11. A. Anistoroaei, A. Berdich, P. Iosif, and B. Groza, "Secure audio-visual data exchange for android in-vehicle ecosystems," *Applied Sciences*, vol. 11, no. 19, p. 9276, 2021.

The author is also passionate about vehicles and automotive engineering which determined her to also contribute to other research papers focused on automotive security. She has eight years of industry experience in automotive, with a primary focus on torque structure, fuel supply, injection, ignition and vehicle motion functions. This experience allowed her to also address subjects related to in-vehicle security, increasing the list of her contributions with new topics of high interest including the design in Simulink of several algorithms for different in-vehicle components and ECUs, e.g., ABS (Anti-lock Brake System), engine management systems, transmission controls, etc. [31, 32]. The design, implementation and validation in Simulink of several countermeasures for attacks on the CAN bus is another contribution of the author in the automotive area [33]. She also did reverse engineering on several signals collected from a passenger car in order to determine the signal functions [32]. Another contribution includes the simulation of several attacks on CAN buses for the adaptive cruise control and autonomous braking systems using scenarios from the European New Car Assessment Program (Euro NCAP) [25]. The author also analyzed the recently released regulations required for vehicle homologation and applied them in her research. She investigated and determined the safety impact based on ISO 26262 in the case of several attacks on the CAN bus for adaptive cruise control and autonomous braking systems [33]. Last but not least, the author's contribution to automotive security includes a threat analysis and risk assessment for the autonomous braking system based on ISO 21434 [25]. These contributions have been submitted for publication in journals as follows:

1. L. Popa, A. Berdich and B. Groza, "Cartwin—development of a digital twin for a real-world in-vehicle can network," *Applied Sciences*, vol. 13, no. 1, p. 445, 2022.

2. A. Berdich and B. Groza, "Secure by design autonomous emergency braking systems in ac- cordance with iso 21434," Machine Learning and Optimization Techniques for Automotive Cyber-Physical Systems, *Springer*, 2023, **(accepted for publication)**.

3. A. Berdich and B. Groza, "Cyberattacks on adaptive cruise controls and emergency braking systems: Adversary models, impact assessment and countermeasures," **(under submission)**.

4. C. Jichici, A. Berdich, A. Musuroi and B. Groza, "Control system level intrusion detection on J1939 heavy-duty vehicle buses," **(under submission)**.

1.4 Organization

The rest of this thesis is structured as follows. Chapter 2 discusses the background of smartphones transducers and survey papers which discuss device fingerprinting based on different sensors, i.e., accelerometers, loudspeakers, microphones and camera sensors. Chapter 3 proposes a system for device pairing in distinct transportation modes, i.e., tram, train, car, bike, walk and shake, based on accelerometer data. Chapter 3 also discusses smartphone fingerprinting based on accelerometer data. Chapter 4 presents a method for smartphone fingerprinting based on loudspeaker roll-off characteristics. Microphone identification using several environmental recordings is discussed in Chapter 5. Chapter 6 analyzes smartphone camera identification using the low and mid frequency of DTC coefficients from dark images. As an extension, Chapter 7 discusses ECU identification. Finally, Chapter 8 concludes this thesis.

Chapter 2

Background and literature review

In this chapter, the author presents the background and literature review related to fingerprinting smartphones based on several sensors characteristics. The content of this chapter is included in a survey article that is under submission [34].

2.1 Background

This section contains a description of the sensor fingerprinting process as well as information on the most popular feature extraction methods, classification algorithms and assessment measures. Also, various application scenarios are suggested.

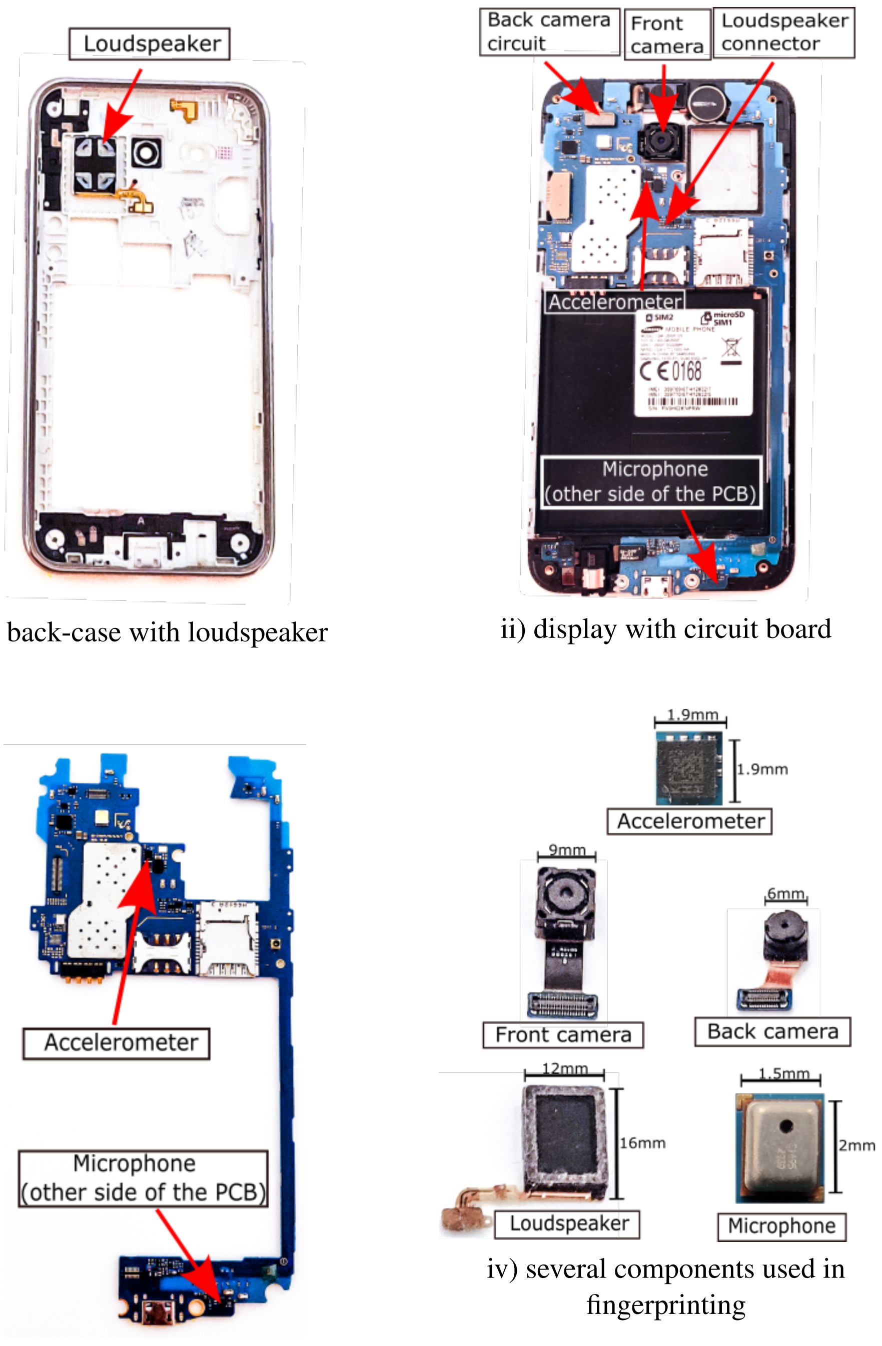
A commonly popular mid-range smartphone, the Samsung Galaxy J5, is dismantled in Figure 2.1. This gadget served as an example for several sensors, including the front and rear cameras, microphone, accelerometer and one actuator, the loudspeaker. The term "transducer", which refers to a device that converts one type of energy into another, is used to refer to both sensors and actuators.

2.1.1 Operation principles for smartphone transducers

What follows is a quick discussion of how the smartphone transducers mentioned above, i.e., accelerometers, loudspeakers, microphones and camera sensors, work.

Operation principle of accelerometer sensor: MEMS (Micro-Electromechanical Systems) accelerometer sensors are a component of smartphones. The MEMS accelerometers' working principle is illustrated in Figure 2.2. The accelerometer has a moving beam structure with a mass on springs and a fixed solid plane. The capacitance between the stationary plane and the moving beam changes when an acceleration is applied because the mass is moving.

Operation principle of smartphone loudspeaker: Figure 2.3 shows the essential elements of a smartphone MEMS loudspeaker. A MEMS loudspeaker includes a sieve cover



i) back-case with loudspeaker

ii) display with circuit board

iii) main circuit board

iv) several components used in fingerprinting

Figure 2.1: A disassembled Samsung Galaxy J5: (i) the case with loudspeaker, (ii) display with circuit board, (iii) main circuit board and (iv) five transducers: accelerometer, front and back camera, loudspeaker and microphone

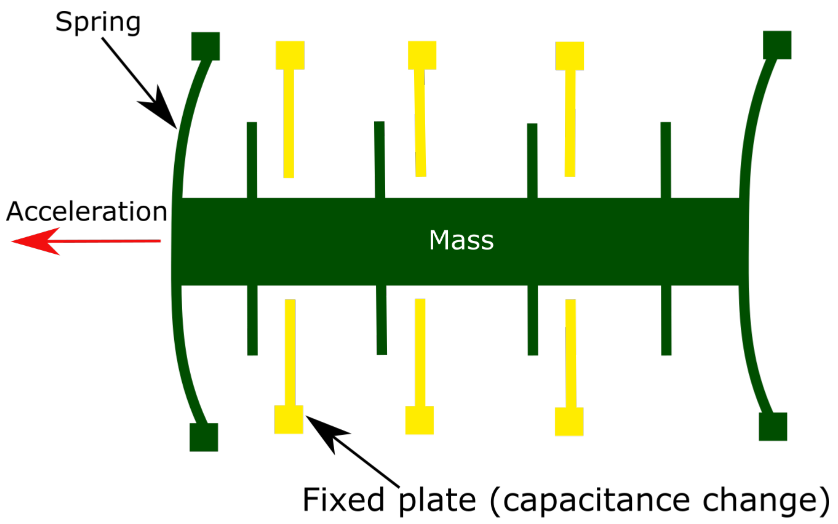


Figure 2.2: Operation principle of MEMS accelerometer

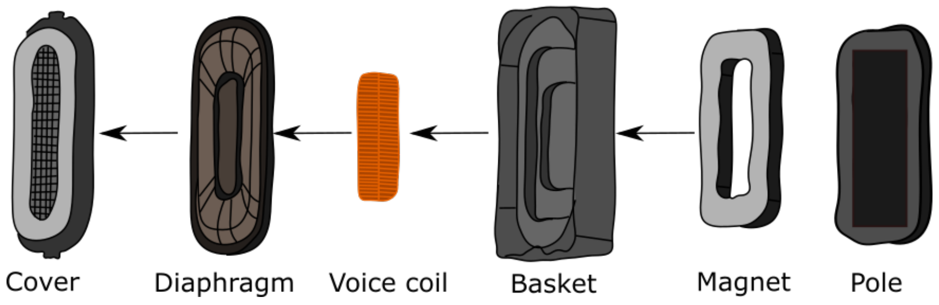


Figure 2.3: Operation principle of MEMS loudspeaker

that protects the diaphragm. The suspension, anchored to the casing and made of flexible material, allows the diaphragm to move. The diaphragm is often made of plastic, but it can also be made of paper or aluminum. A voice coil is present behind the diaphragm that is installed in the loudspeaker's main case. This is followed by a pole and magnet that, when combined with the voice coil, provide a magnetic force that causes the voice coil to vibrate and the diaphragm to make a sound.

Operation principle of smartphone microphones: MEMS microphones are used in smartphones because of their compact size, low cost and low power consumption. The parts of a MEMS microphone are shown in Figure 2.4. The case that contains the microphone has a small gap that makes it easier to hear sounds. The two main parts of the case are an ASIC (Application-Specific Integrated Circuit), which amplifies the signal received from the transducer and performs the functions of the ADC (Analog Digital

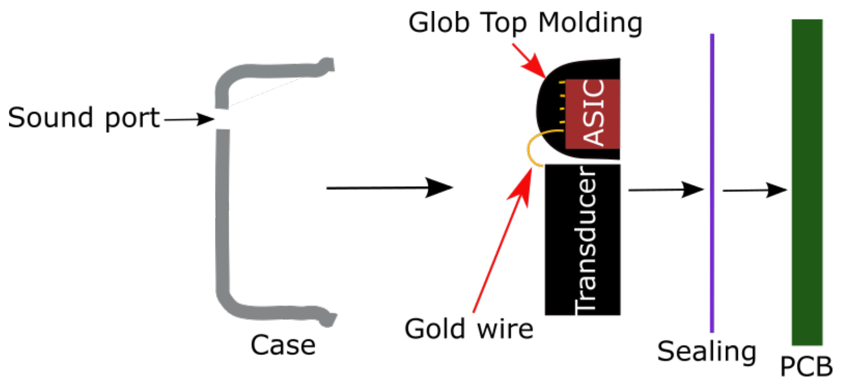


Figure 2.4: Operation principle of MEMS microphone

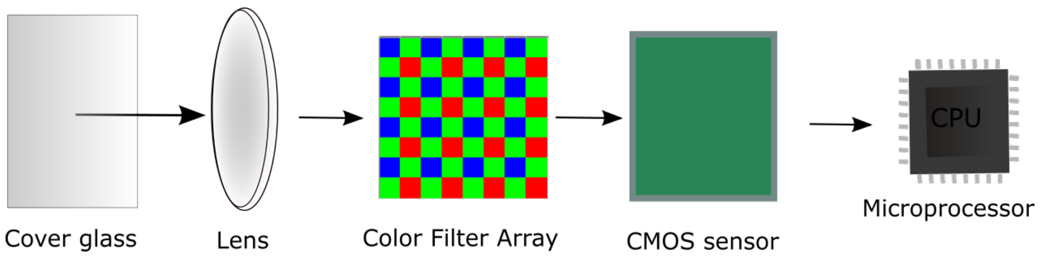


Figure 2.5: Operation principle of camera sensor

Converter) and a transducer, which converts the acoustic signal into an electrical signal. A golden wire connects the transducer to the ASIC. A unique sealing substance is utilized to hermetically isolate the microphone to enhance the quality of the sound received. On the backside of the sealing material is an illustration of the PCB (Printed Circuit Board).

Operation principle of smartphone camera sensor: Digital cameras and systems that need to capture high-quality images employ CCD sensors. The two most popular types of camera sensors are CMOS (Complementary Metal-Oxide Semiconductor) and CCD (Charge-Coupled Device). Due to their reduced size and lower power requirements, CMOS sensors are mostly utilized in small-sized devices like smartphones, laptops, IoT devices, etc. [35]. The working principle of a CMOS sensor is shown in Figure 2.5. The lens collects light, which is divided into three colors, i.e., red, green and blue, by a Bayer filter array. Since the green light is more perceptible to the human eye than red or blue, half of the elements of the filter are green and the other half is split between red and blue. Finally, the CMOS sensor creates an electrical signal from the light.

2.1.2 Frequently used features for device fingerprinting

The most popular feature extraction methods for facilitating smartphone identification based on data acquired from the aforementioned transducers are briefly summarized.

1. Time and frequency domain features. Several works extract different time and frequency domain features for signals obtained from smartphone sensors like accelerometers, gyroscopes, loudspeakers or microphones, etc.

   (a) There are many features in *time-domain* that are extracted from sensor data and used to identify smartphones. The most popular ones are the following: mean, standard and average deviation, kurtosis (tailedness), skewness (asymmetry), Root Mean Square (RMS), Zero-Crossing Rate (ZCR), maximum and minimum values, non-negative count, variance, mode and range, etc. A complete list would be out of scope.

   (b) The most used *frequency-domain* features are the spectral centroid, skewness, spread, kurtosis, flatness, entropy, roll-off, brightness, roughness, RMS, irregularity, flux, attack slope, attack time, variance, mean, low energy rate, standard deviation and DC component from DCT (Discrete Cosine Transform). Again, different works may use other statistical characteristics. Distinct time and frequency domain characteristics are used in [36, 37, 38, 39, 40, 41, 42, 43], while in [44] only time domain features are used.

   (c) Correlation, i.e., $corr(x, y)$, give a statistical connection between two variables x and y. It is calculated as:

   $$corr(x, y) = \frac{cov(x, y)}{\sigma_x \times \sigma_y}$$

   where $cov(x, y)$ is the covariance of x and y, σ_x is the standard deviation of x and σ_y is the standard deviation of y. There are many works which used correlation for smartphone identification, i.e., [45, 46, 47, 48, 49, 50, 51, 52, 53, 54, 55, 56, 57, 58, 59].

   (d) The Euclidean distance is used for loudspeaker identification in [60]. The Euclidean distance is computed as the square root of the difference between two samples:

   $$dist(a, b) = \sqrt{\sum_{i=1}^{n} (a_i - b_i)^2}$$

   where a and b are signals from two devices, a_i is the i-th sample from signal a and b_i is the i-th sample from signal b.

   (e) The Hamming distance gives the number of indexes at which the respective symbols are distinct. It is computed as:

   $$d(s, t) = \sum_{i=1}^{n} |s_i - t_i|$$

where s and t are signals from two devices, s_i is the i-th sample from signal a and t_i is the i-th sample from signal t.

2. Features extracted from camera-collected images:

   (a) Fixed-Pattern Noise (FPN) is the sensor's noise that leads some pixels to be brighter than average. There are two different forms of FPN based on image types: Dark Signal Inhomogeneity (DSNU) [47, 49, 50, 51, 52, 53, 57, 61, 62, 63, 64, 65, 66, 67, 68, 69, 70] which appears in lighting circumstances. The most used method for identifying camera sources is PRNU [71].

   (b) Discrete Cosine Transform (DCT) is a typical method for transforming a picture from the spatial domain to the frequency domain. While the Inverse Discrete Cosine Transform (IDCT) is utilized for decompression, DCT is applied to 8x8 image blocks in JPEG compression [72]. This transformation can also be applied to DSNU and PRNU [73, 74].

   (c) Local Phase Quantization (LPQ) and Local Binary Pattern (LBP) are two additional characteristics that are frequently utilized for image processing in the context of camera identification [75]. LBP is a description of local texture patterns in images. The image is divided into 3x3 blocks, with the central pixel serving as the threshold for the neighbor pixels [76, 77]. The descriptor, LPQ is based on the blur invariance from pictures derived from the extracted Fourier phase spectrum.

3. Features extracted from audio signals:

   (a) The Power Spectrum (which is a normalization of the FFT amplitudes) is the most fundamental technique for extracting frequencies from the spectral estimates of the audio signal. Another way is to use directly the frequency-amplitude pairs generated by applying the FFT transform. In the context of loudspeaker and microphone identification, such features are frequently used for audio signals [21].

   (b) Mel-Frequency Cepstral Coefficients (MFCCs) are the typically exploited feature for audio signals. Since the coefficients are frequently utilized in speech recognition tasks, this technique is specifically used in various research papers to extract features from the human voice in the context of microphone identification [78, 79, 80, 81, 82, 83, 84] and loudspeaker identification [85, 86, 87]. The audio signals are divided into windows for extraction of the MFCC coefficients and the FFT (Fast Fourier Transform) is calculated for each window. The result is fitered using a Mel filter and the logarithm of each Mel frequency is obtained. The DCT is then applied to the result, resulting in the MFCC coefficients.

(c) Linear Frequency Cepstral Coefficients (LFCCs) is a method similar to MFCC. However, instead of employing the Mel filter, a linear filter is employed. Other methods for analyzing human speech include Perceptual Linear Prediction Coefficients (PLPC) and Linear Predictive Codes Coefficients (LPCC).

2.1.3 Frequently used separators and classifiers for device identification

The following paragraphs briefly summarize the most popular classification approaches for fingerprinting each smartphone component mentioned above. Several strategies have been taken into consideration, ranging from some fundamental indicators to deep learning:

1. Thresholding is a well-known technique in image processing for image segmentation or turning a grayscale image into a binary one. It is also used to categorize different sensor data. Thresholding is typically used when fingerprinting smartphone sensors in the context of camera identification [54, 88, 61, 69, 89, 90, 91]. Thresholding is also frequently used as a stand-alone method for classification. This method is also employed for classification when additional signals are present, such as accelerometers [44] or different device configurations [92].

2. The intra and inter-distances help distinguish between devices based on identified distance metrics, such as the Euclidean or Hamming distance.

   (a) The intra-chip distance, computes as the arithmetic average between the fingerprints extracted at various times from the same chip. Although this metric may be calculated for any fingerprint, it is most frequently used to assess PUFs, such as those built using CMOS sensors [90], [91], where the distance is calculated as the Hamming distance between two pictures. The average amount of flipped bits among the PUFs from various images is shown by the intra-chip Hamming distance. The intra-chip Hamming distances can also be used to calculate the BER (Bit Error Rate). Based on intra-chip Hamming distances, the reliability can be determined. As proposed in [93], the intra-chip Hamming distance is computed as:

$$dist_{INTRA} = \frac{1}{m}\sum_{j=1}^{m}\frac{dist(R_i, R_{i,j})}{n} \times 100\%$$

$$BER = dist_{INTRA}$$

$$Reliability = 100\% - dist_{INTRA}$$

where R_i is the correct PUF determined from the average of all PUFs of the examined chip and $R_{i,j}$ is the PUF of the jth image, n is the number of bits and m is the number of images.

(b) The inter-chip distance is computed based on the Hamming distance between the PUFs of two different chips. The inter-chip distance describes the uniqueness of a PUF. Based on [93], the inter-chip distance can be computed as:

$$dist_{INTER} = \frac{2}{m(m-1)} \sum_{u=1}^{m-1} \sum_{v=u+1}^{m} \frac{dist(R_u, R_v)}{n} \times 100\%$$

$$Uniqueness = dist_{INTER}$$

where R_u is the PUF of the u-th chip, R_v is the PUF of the v-th chip, n is the number of bits and m is the number of images.

There are several works which used intra and inter-chip distance for camera identification [88, 94, 89, 95].

3. Classical machine learning approaches:

(a) Support Vector Machine (SVM) is a supervised machine learning approach which can be used to train multi-class or binary models. According to the literature, SVM, a popular classification technique, seems to be more frequently utilized for camera sensor identification [63, 66, 76, 77, 96, 97, 98, 99, 100, 101, 102] and microphone identification [79, 80, 83, 103, 104, 105, 106, 107, 108, 109, 110, 111, 112, 113]. Occasionally, it was also used for other transducers, e.g., loudspeakers [87], accelerometers [37, 38], etc.

(b) K-nearest Neighbor (KNN) is another popular supervised classification technique used in the literature for smartphone identification based on multiple components, such as microphones, [103, 104, 108, 109, 112], loudspeakers [86, 85], accelerometers [37, 38], etc. The KNN usually uses the Euclidean distance between the training and test samples.

(c) Gaussian Mixture Model (GMM) is a probabilistic function defined as the sum of Gaussian component densities. It is advised to employ GMM for speech recognition tasks. In the literature, in the context of sensor identification, the CMM is used for microphone fingerprinting, in particular when human voice is involved [78, 80, 81, 84] and loudspeaker-based identification [85, 86].

(d) Gaussian Supervector (GSV) is a GMM-based technique that creates a supervector by concatenating all of the features from each Gaussian component [114]. As anticipated, GSV was used to identify microphones based on human speech [107, 110].

(e) Random Forest (RF) in an ensemble technique like as Subspace Discriminant, Bagged Trees, Subspace KNN, RUSBoost Trees and GentleBoost. RF was utilized to identify accelerometers [37, 38], camera sensors [73, 101, 115], loudspeakers [87], as well as for smartphone recognition based on multiple sensors [41, 42], etc.

(f) Decision Tree is a different supervised machine learning approach and the data is organized as a tree with internal nodes that hold the features from the dataset. The outputs are represented by leaf nodes, which are the end nodes, while branches hold the decision rules. This method was utilized for magnetometer-based smartphone identification [43], gyroscope [116], multiple sensors [39, 40, 117], etc.

(g) Linear Discriminant Analysis (LDA) and Quadratic Discriminant Analysis (QDA) are methods for supervised machine learning that utilize the Gaussian distribution. LDA employs linear Gaussian distributions, which results in linear class borders, whereas QDA uses quadratic Gaussian distributions, which results in non-linear class boundaries. In the literature, the LDA was used for smartphone identification based on microphones [22], smartphone identification based on wireless charging [39] and for smartphone identification based on magnetic induction emitted by the CPU [118]. QDA was used for smartphone recognition based on accelerometer and gyroscope data [39, 40].

4. Deep learning approaches:

(a) Convolutional Neural Networks (CNN) are deep learning methods that were applied for audio data and numerous other time-domain series in addition to their usual use for extracting patterns from images. CNN are typically utilized for camera sensors for identifying devices based on their sensors [66, 67, 70, 75, 102, 119, 120, 121, 122, 123, 124, 125, 67, 126, 127], microphones [108, 109, 112, 128, 129, 130], loudspeakers [21, 87], but also for signals from other sources such as peripheral input timestamps [131]. AlexNet is a convolutional neural network proposed in 2012 for image classification [132]. A convolutional neural network called AlexNet was proposed in 2012 for image classification. With five convolutional layers, max-pooling layers, three fully connected layers and a softmax layer, AlexNet can be utilized as a pre-trained neural network. It was employed in [119] to identify the camera sensor. Another convolutional neural network with 22 layers was GoogleNet, which was suggested in 2015 [133]. For camera sensor identification, GoogleNet was used in [70, 119]. ResNet, or residual neural network, was first published in [134] in 2016. There are various types of ResNet depending on the amount of layers, such as ResNet18, which has 18 layers, ResNet50, which has 50 layers and ResNet101, which has 101 layers. Although RetNet is a pre-trained neural network, it is adaptable. It is used to identify camera sensors as both a pre-trained network and an adaptive network in [68, 124, 135, 136].

(b) Long Short-Term Memory (LSTM) and Bidirectional Long Short-Term Memory (BiLSTM) are two types of recurrent neural network layers used for time series and sequence data. They were utilized for microphone [130] and loudspeaker identification [21, 87]. According to the results of [21], their performance for loudspeaker detection rate is comparable with CNN.

2.1.4 Commonly used performance metrics for classifiers

The most used performance metrics in the literature are summarized next.

1. The following parameters serve as the basis for all of the following metrics: the true positives TP, true negatives TN, false negatives FN and the false positives FP.

2. Accuracy is the ratio of correctly recognized items to all items:

$$Accuracy = \frac{TP + TN}{TP + FP + TN + FN}$$

   The validation accuracy can be also calculated as:

$$Accuracy = 1 - kfoldLoss$$

   where $kfoldLoss$ is the identification error using k-fold cross-validation.

3. Precision is the ratio of identified items that are correctly predicted to belong to the target class:

$$Precision = \frac{TP}{TP + FP}$$

4. The True Positive Rate (TPR) or recall, is the ratio of relevant outcomes that are correctly recognized: :

$$Recall = TPR = \frac{TP}{TP + FN}$$

5. True Negative Rate (TNR) is the ratio of the wrong samples which are recognized as targets:

$$TNR = \frac{TN}{FP + TN}$$

6. F-measure, or F1-score, is the harmonic average between the recall and precision:

$$F1 - score = \frac{2 \times precision \times recall}{precision + recall}$$

7. False Acceptance Rate (FAR) is the proportion of items which are incorrectly recognized as targets:

$$FAR = \frac{FP}{TN + FP}$$

8. False Rejection Rate (FRR) is the proportion of items which are incorrectly rejected:

$$FRR = \frac{FN}{TP + FN}$$

2.1.5 Application scenarios

Smartphone fingerprinting methods are useful in many areas, including device authentication, numerous applications and forensics investigations. Next, they are discussed.

Authentication

In order to reduce user interaction and the vulnerability brought on by weak security tokens like passwords, device authentication and multi-factor authentication based on device fingerprints are effective. In other words, one factor authentication (user or device) is a specific device fingerprint. This is crucial for IoT applications when quick and secure authentication techniques are required, yet devices lack a user interface or cannot be conveniently accessible (e.g., situated in an awkward location). The works employing device fingerprints for authentication are numerous, some are listed here.

Generic device authentication. Some works have combined data from multiple sensors, such as gyroscope, accelerometer and camera and used it for a robust smartphone authentication [117]. In contrast, others have used specific sensor fingerprints, such as microphone fingerprints [137] and accelerometers [44, 20]. Many papers, including [50], [91] and [95], propose using the PUFs obtained from camera sensors for authentication. Additionally, [41] uses gravity, acceleration, the magnetic field, rotation vector, orientation, gyroscope sensors and linear acceleration to extract smartphone fingerprints for identification. Smartphone recognition, in the scope of authentication, is proposed in [138] using microphones and loudspeakers. A smartphone emits audio signals between 4 and 20 kHz, with a step of 400 Hz and another smartphone records them. Authentication depends on the correlation of the signals.

Specific environments for authentication. Depending on the precise area of application, certain works are more specialized. The automotive environments seem to be a particular scenario that is more interesting. Accelerometer data from diverse transportation situations is explored in [20, 139, 140, 141]. In [21] smartphone fingerprinting is done using data gathered from in-car infotainment systems. The infotainment system records the audio while the smartphone emits a linear sweep from 20Hz to 20kHz. Additionally, [30] suggests an in-vehicle authentication mechanism for use with the infotainment system and the smartphone. Also, smartphones are frequently used for IoT device identification [142], [143].

Specific applications for sensor data

In what follows, various advantageous applications of sensor data are depicted, but it is important to stress that disclosing this information also puts users' privacy at risk. The use of the accelerometer and gyroscope can be utilized to recognize activity [19] and make recommendations for the user's health. Another application that could be useful to insurance companies and car rental businesses is driving style recognition [144, 145, 146, 147, 148]. The accelerometer data has been used for gait identification [149], real-time pothole detection [150] and monitoring of road conditions [151]. The use of motion sensor data for theft detection has also been suggested in [152]. Other use cases covered by other survey works include quality control of smartphone sensors [5] and attack, malfunction, or fault diagnosis based on device fingerprints [9].

Privacy concerns. A significant privacy concern is that smartphone fingerprints can be used to track users. Accelerometers have been utilized for user tracking [36], metro rider tracking [153] and metro station activity detection [15]. Other publications examine safeguarding against privacy threats for certain types of data, such as cameras [56] or loudspeakers [85], [60]. More limits are being implemented to the availability of such (meta) data as smartphone operating system platforms grow more concerned with exploiting sensor data by apps for device fingerprinting and user tracking reasons.

Forensics investigations

The topic of forensics investigations is complimentary. The fingerprinting of smartphones using various sensors, such as the microphone ([83, 111, 130], camera [66, 102, 123, 154], etc. can be used to identify (suspected) criminals by comparing the sounds or images that were captured in connection with the relevant crime [155, 156].

The combat against the potentially hazardous effects of AI is another recently-emerging concern. Deepfake audio and video records are already made using machine learning algorithms to produce incredibly realistic recordings [157]. This technology can be used by organized crime to manipulate public opinion by spreading false information or defaming public figures, or endangering national security [158]. Deepfake detection techniques are (currently) not particularly effective at preventing the harmful impacts of deepfake applications. However, source camera identification can be utilized to enhance the results [159, 160]. Deep fakes may be prevented for image, video and audio documentation by building an end-to-end trust chain from the raw sensor data (with unique device fingerprints) to the final report when sufficiently unique fingerprints are obtained from camera or microphone sensors are combined with strong cryptographic binding to the respective device.

2.1.6 Countermeasures

Malicious apps might utilize these fingerprints to violate users' privacy. Several countermeasures can be used to defend against these assaults, including lowering the sampling rate, introducing noise into the sampled data and limiting access to sensors and data. These strategies can also be combined. The paragraphs that follow describe these.

Adding noise. Adding noise is a fast and easy way to alter smartphone fingerprints. According to [5], this method has no functional impact on smartphones, uses low power and is not computationally expensive. The inclusion of noise has also been explored in the context of microphone identification in [22]. This investigation takes into account a variety of noises, such as traffic, trains, barriers, etc., and finds that when the SNR falls below a particular threshold, such as -40 dB for car horns or -20 dB for car tiers, microphone identification ceases to function and accuracy falls below 50%. Additionally, the accuracy falls below 50% at an SNR of 0-5db, according to the author's analysis of the impact of AWGN (Additive White Gaussian Noise) at various SNR levels in [108]. The research in [21] also demonstrates how the volume might affect the fingerprints while identifying loudspeakers.

Restricted access to device peripherals and data. Another countermeasure suggested in [4] and also covered in [5] is the implementation of policies that control the access rights of other applications to sensor data. Remembering that rogue apps with access to the microphone can intercept the phone's PIN code [161] is perhaps also important. This demonstrates how significant the consequences of granting access to such peripherals are. The authors of [162] propose a cloud-based framework to address the issue of mobile data privacy, but they also claim that the issue is still not entirely resolved and is unlikely to be resolved under the current circumstances. A privacy risk assessment for mobile applications was recently presented in [163]. This evaluation considers the information flow leakage and the permissions granted to the applications.

Lowering sampling fidelity. Lowering the sample rate can be used as a protection mechanism and extends battery life (in the case of data collected from motion sensors). Data filtering and lowering the sample rate can hide some features, making fingerprinting impossible. In [164], is analyzed the effect of changing the sampling rate for accelerometer data from 100Hz to 25Hz.

2.2 Literature review

Many papers discuss smartphone fingerprinting based on their sensors. This section discusses the research papers focussed on smartphone fingerprinting based on their accelerometers, loudspeakers, microphones and cameras.

2.2.1 Smartphone identification based on accelerometers

The author begins a survey of several publications that address device identification based on accelerometer sensors. It is required to note that while several studies describe device pairing using accelerometer data, very few papers specifically address smartphone fingerprinting using accelerometer data.

The authors of [165] discuss smartphone identification based on accelerometer sensors and smartphone identification based on their microphones. In the case of accelerometers, the measurements are taken with the smartphone at a constant speed. The first sample from each measurement is used as the smartphone fingerprint. Only 15.1% of devices could be successfully detected using this method.

The authors in [36] and [37] both employ several time and frequency domain features, however in [44] solely time domain features are used. The authors in [36] use frequency domain features such as spectral standard deviation, spectral centroid, spectral skewness, spectral kurtosis, spectral crest, irregularity-k and J, smoothness, flatness and roll off in addition to time domain features such as mean, standard deviation, average deviation, skewness, kurtosis, RMS, the lowest and highest value. The mean precision and recall are above 99% for 107 accelerometers (25 smartphones, two tablets and 80 freestanding accelerometers). Ten time domain features and ten frequency domain features, including mean, min, max, var, std, mode, range, skewness, kurtosis, RMS, DC, spectral mean, spectral variance, spectral standard deviation, spectral spread, spectral centroid, spectral entropy, spectral skewness, spectral kurtosis and spectral flatness, are used in [37]. SVM, KNN, LR, RF, Extra Tree and eXtreme Gradient Boosting (XGBoost) are employed as six classifiers for classification. The authors find that for seven devices, they can achieve precision between 54.5% and 100% and recall between 88.9% and 94.3%.

There were just 8 time domain features explored in [44], i.e., min, max, kurtosis, RMS amplitude, mean deviation, skewness, standard deviation and mean. The thresholding strategy was employed for classification, which achieved a 0.7444 TPR and 0.0978 FPR for 15 devices.

2.2.2 Smartphone identification based on loudspeakers

The author now reviews various works that address loudspeaker-based smartphone identification.

Loudspeaker fingerprinting employs the use of two different sorts of sounds: (i) synthetic sounds like cosine waves [60] and linear sweeps [21] and (ii) natural sounds like instruments, music [85], [86] and human speech [85], [86], [87].

Fifty identical smartphones are fingerprinted by the authors in [60] using cosine waves between 14kHz and 21kHz, with an increment step of 100Hz, released by each loudspeaker. Euclidean distance is used to identify the smartphones and an error rate of about $1.55 * 10^{-4}\%$ is realized. Using a linear sweep signal between 20Hz and 20kHz transmitted by an in-vehicle head unit, the authors in [21] fingerprint 28 smartphones loudspeakers, of which 16 are identical loudspeakers put in the same smartphone cover. The power spectrum's roll-off characteristics are utilized in this work. A linear approximation is utilized for classification along with machine learning algorithms like KNN, RF and SVM and deep learning algorithms like CNN and BiLSTM (the latter two deep-neural networks are the main subject of the investigation). The accuracy of identical smartphone speakers is between 95% and 100%. The authors also examined how speaker orientation and volume level affect the fingerprinting process. The trials are also conducted for four different smartphones at volume levels of 50%, 75% and 100%. The authors note that while the fingerprints for each smartphone are grouped around the volume, the smartphone can still be recognized unambiguously. The same result was seen in experiments with loudspeakers oriented at 0°, 90°and 180°. The authors performed 3000 measurements for this work.

There are a total of 15 time and frequency domain features used in [85] and [86] including RMS, ZCR, low energy rate, spectral centroid, spectral entropy, spectral irregularity, spectral spread, spectral skewness, spectral roll-off, spectral brightness, spectral flatness, MFCCs, chronogram and total centroid. Three different forms of sounds, i.e., instrumental, natural sound and human speech were used to extract the fingerprints. The authors employ KNN and Gaussian mixture model (GMM) classifiers for classification. Both different and identical smartphones are used in the studies. Using the MFCC coefficients generated from human speech, the authors of [85] achieve 98.8% accuracy for 19 smartphones (identical and distinct), compared to 93% accuracy for 15 identical smartphones. In [86], the authors utilized the MFCC coefficients from each signal (musical instruments, song and human speech) with the KNN classifier to get a 100% F1 score for 52 smartphones, out of which no more than 15 are similar. The F1 score on MFCC is 100% when GMM is utilized for instrumental sounds, however it is just 99.6% when it comes to human voice and music. MFCC produced the best results in these two articles, where 15 time and frequency domain features were used. Twenty-four smartphones are clustered using deep learning techniques, i.e., CNN, BiLSTM and machine learning methods, i.e., SVM and RF. In [87], MFCC and SSF (Sketches of Spectral Features) generated from human voice are employed. The authors reached the highest accuracy of 99.29%.

2.2.3 Smartphone identification based on microphones

Mobile device identification based on their microphones is discussed in the following works.

Identification of microphones using synthetic sounds

In [103], several musical styles, such as metal, pop, techno and instrumental, are employed, for microphone identification, along with sine waves and white noise. From the recorded sounds, Fourier coefficients are retrieved and various classifiers, including NB, multiclass SVM, decision trees and KNN, are then used. With seven microphones, this method was tried and the best accuracy obtained was 93.5%. For microphone identification in [58], ambient noise produced by a fan cooler is employed. The authors cluster eight commercial microphones based on inter-class cross-correlation using 24 recordings and they achieve 100% accuracy in their categorization. For microphone identification, the authors in [104] uses sounds from indoor spaces, outdoor parks and busy streets. Five microphones are identified using one class of classification algorithms: the Gaussian Model (GM), Gaussian Mixture Model (GMM), KNN, Principal Component Analysis (PCA) and Incremental Support Vector Machine (ISVM). In terms of results, the recall ranges are from 0.774 to 0.859 for inside spaces, from 0.7354 to 0.885 for park noise and from 0.206 to 0.784 for street noise. Using the Representative Instance Classification Framework (RICF), the authors enhanced to a recall between 0.741 and 0.874. A technique based on FFT characteristics retrieved from background noise was detailed in [106]. The SVM classifier reached a maximum accuracy of 96.72% for 21 devices.

Smartphones microphone identification using sine waves at 1 kHz and 2 kHz, is discussed in [108]. SVM, KNN and CNN are the classification methods used for classification of 32 smartphones. The authors tested the suggested method at various SNR (Signal-to-Noise Ratio) levels. The accuracy declines to 67.27% for 1kHz and 82.75% for 2kHz for 10dB SNR, whereas accuracy for 20dB SNR is 96.8% at 1kHz and 96.8% at 2kHz. Additionally, sine waves at 1 kHz and the SVM, KNN and CNN classifiers are used in [109]. At 10dB SNR, the accuracy for 34 smartphones is 80% for CNN, 40% for SVM and 10% for KNN. In [112], a pneumatic hammer and gunshot sounds are added to the 1KHz sine wave. In [166], the authors created 80 sine waves between 100Hz and 8kHz before using a single-layer artificial neural network to detect six commercial microphones with 100% accuracy.

The authors in [137] use three binary classifiers in cascade and extract 15 features from the time and frequency domains, such as RMS, ZCR, low energy rate, spectral centroid, etc. Ambient noises from several locations are used, including buses, food court, kids playing, metro, restaurants, etc. The TPR for this method reaches 81% for one model and 98% for the other, when 12 smartphones of two different models were tested.

Identification of microphones using human speech

The MFCC coefficients obtained from the human speech of 12 males and 12 females collected with 21 smartphones are used in [105] to identify smartphone microphones using three classifiers, i.e., Radial Basis Functions Neural Network (RBF-NN), Multi-Layer Perceptron (MLP) and SVM. With RBF-NN, the maximum accuracy was 97.6%. The GMM (Gaussian Mixture Model) is used in [78] and the maximum accuracy attained is 99.58%. Four speakers were recorded using 16 microphones and the features used by the authors include the LPCC, PLPC and MFCC coefficients. Additionally, the SVM classifier and MFCC coefficients collected from human voices are applied in [79] to cluster 26 smartphones. 90% accuracy was attained. The SVM classifier was improved by Utilizing Sequential Minimal Optimization (SMO). Fourteen smartphones are clustered using MFCC, LFCC, GMM and SVM in [80]. The accuracy achieved is 98.39%. Using the GMM and the MFCC coefficients generated from human speech, the greatest reported accuracy is 99.27% in the case of 16 devices [81]. GSV and MFCC are employed in [107] to identify features from human speech. The SVM classifier is used for clustering and an error rate between 2.08% and 7.08% is obtained for 14 devices.

The authors in [59] use the features of audio signals, such as mean, standard deviation, crest factor, dynamic range and auto-correlation, to identify two similar microphones. A neural network and Gaussian SVM were employed by the authors of [82] to identify 21 smartphones based on their microphones. The classifiers were fed with the features given by MFCC from human speech signal. The claimed accuracy is 88.1%. In [167], a band energy descriptor is suggested as a classifier. The accuracy of this method is 96% for 170 devices that capture human voices. Forty smartphones are recognized in [128] with the greatest achieved accuracy of 99% based on human speech. In [110], the voices of 25 speakers are used. For four microphones, GSV and the Sparse Representation-based Classifier (SRC) achieve an accuracy ranging from 78.17% to 85.58%. Human speech is again used in [83],[84], [111], [129], [130] and [168].

In [113], a unique strategy based on Electrical Network Frequency (ENF) is suggested. The true positive rate is greater than 60% for seven devices.

2.2.4 Mobile device identification based on camera sensors

Finally, the author reviews research on camera sensor-based device identification.

PUF-based approaches

The authors in [45]l propose a method to create a PUF from camera sensors based on Photo-Response Non-Uniformity (PRNU) noise. The authors used 320 images from 9 cameras to validate their suggested strategy and the correlation function is used as the classifier. According to their findings, the False Rejection Rate (FRR) ranges from 1.36×10^{-1} to 4.41×10^{-14} depending on the correction factor used and the JPEG compression.

Additionally, PRNU is utilized in [50] to identify cameras. After using a High-Pass filter to reduce the noise from the photos, the high frequencies are used to extract the camera fingerprints. In order to validate the approach, the authors used 14 cameras, consisting of one DSLR and 13 smartphones. The resulting correlation for full photos is between 0.0022 and 0.02, while for flat images, it is between 0.0021 and 0.059.

An alternative strategy based on dust spots from photographs taken by Digital Single Lens Reflex (DSLR) cameras is suggested in [169]. The shape characteristics and a Gaussian identity loss model is used to identify dust patches. The authors employ four cameras for the tests and to cluster them, they compute a confidence value based on each dust spot's occurrence, smoothness and shift validity parameters. The accuracy of the identification reach 99.1%.

In the literature are suggested particular PUFs for various CMOS sensor technologies. In [90], a PUF for 65 nm CMOS sensors using hardware improvements is presented. Results are produced at temperature changes between 0° C and 100° C reaching a uniqueness of 50.12% and reliability of 100% using a thresholding methodology to validate the procedure. In [91], another PUF based on FPN is proposed. Five 180 nm camera sensor chips are utilized to validate the findings and the thresholding approach is employed to cluster the data. The uniqueness is 49.37% and reliability is 99.80% for temperature fluctuations between 15° C and 115° C. For an event-driven PUF for 1.8 V 180 nm the authors in [170] propose CMOS sensors based on Dynamic Vision Sensor (DVS). The uniqueness is 49.96% and the reliability is between 96.3% and 99.2% for temperature changes between -35° C and 115° C. In [95], another PUF for 180 nm CMOS sensors based on DVS is discussed. At temperature ranges between -45° C and 95° C, reliability larger than 98% is obtained. In [89] an optical PUF for 65 nm CMOS sensors are presented. 14 CMOS sensors is employed in the tests and 1-D autocorrelation and thresholding are used to validate the methodology. The authors achieve an intra-chip HD of 0.251% and an inter-chip HD of 49.81%.

In [88], a PUF based on DSNU is suggested for smartphone CMOS sensors. After de-noising the image, the DCT is applied, high-frequency components are removed and the IDCT is used. The thresholding technique is then used to eliminate bright pixels. The method is tested on five identical sensors from 2 different smartphones and the results show that the inter-chip HD ranges from 46% to 54%, while the intra-chip HD is less than 10%.

In [94], a PUF based on a camera sensor SRAM is presented. For 20 devices, the average intra-chip HD is 0.51% and the average inter-chip HD is 49.95%.

Machine learning approaches

Machine learning methods are widely used in research articles to identify cameras. A sizable portion of them makes use of the SVM classifier. In [96] the SVM classifier uses the lens radial distortions as features. The SVM classifier achieves an accuracy of 91% for three cameras. Additionally, the authors in [76] use a multi-class SVM, but the features are extracted based on Local Binary Patterns (LBP). For 18 cameras, the average accuracy is 98%. In [63] is used the SVM classifier, which has as input the PRNU and wavelet transform as features. The average accuracy attained from 14 camera models from 5 manufacturers is 87.214%. The authors in [97] additionally use Local Binary Pattern (LBP) and Local Phase Quantization (LPQ) as inputs for the SVM classifier. For 14 different camera models, the accuracy ranges from 98.13% to 100%. The authors in [99] use SVM with Radial Basis Kernel. They use a Local Binary Pattern to extract an I-Vector for the green and red channels of the images (LBP). Three different cameras are employed in the studies and the overall prediction accuracy is greater than 99%. Additionally, using an SVM classifier, the authors in [99] achieved an accuracy of 99.01% for eight camera models. In [100], the SVM classifier also uses feature representation as an input. The identification accuracy for 27 cameras is 87.6%. In [77], Weber's and LBP's (WLBP) characteristics are discussed. Features are converted into a vector and then used as input by the SVM classifier. For nine cameras, this technique achieves 99.52% accuracy.

Deep learning algorithms are also utilized in a significant number of research papers. For camera identification, the authors in [119] use CNN, AlexNet and GoogleNet. A high-pass filter is used to initially filter the photos and then deep learning techniques are used. The accuracy for 33 cameras from two datasets is 83.5% for GoogleNet, 94.5% for AlexNet and 91.9% for CNN. In [75], a CNN approach based on features extracted using Local Binary Pattern (LBP) and Local Phase Quantization (LPQ) is suggested. For ten different camera models, the accuracy ranges from 84.1% to 99.5%. The photos are divided into k patches using sliding windows in [120] and the extracted features are then utilized as input for a CNN. Additionally, CNN was used to identify the source camera in [121]. The authors achieve an average accuracy of nearly 100% for 74 cameras. A Content-Adaptive Convolutional Neural Network (CA-CNN) is constructed by the authors in [122]. For 74 cameras, the achieved detection accuracy ranges from 89.56% to 97.37%. In [136], a method for identifying source cameras using photos from Facebook is suggested. Based on an existing ResNet50 network, the authors suggest a deep learning neural network. The network's maximum categorization accuracy was 96% when photographs from 5 cameras were posted to Facebook and then downloaded again.

To remove the noise from the photos, the authors in [123] employ a CNN. The average precision for 125 cameras is between 0.144 and 0.399 and the F1-score is from 0.205 to 0.444. In [101], transfer learning and CNN are employed for feature extraction, while SVM, logic regression and Random Forest (RF) are the machine learning methods used for camera identification. In terms of results, five cameras are identified with 98.82% RANK-1 accuracy using SVM as the final layer. RANK-1 accuracy was achieved with RF at 97.16% and logic regression at 98.57%. The RANK-5 accuracy for each of the classifiers included was 100%. A multi-scale High Pass Filter (HPF) was employed by the authors in [124] to eliminate noise from the images. For camera clustering, the authors employ a multi-task learning strategy based on CNN and ResNet.

For 125 devices, this method achieves an accuracy of 84.3%. In [102], multiple classifiers, including Weibull-calibrated SVM (WSVM), Decision Boundary Carving (DBC), Specialized SVM (SSVM), SVM with Probability of Inclusion (PISVM) and Open-Set Nearest Neighbors, are used to classify features extracted using statistical descriptors, CFA (Color Filter Array) and CNN-derived features (OSNN). The upper-left corner of the images serves as the input for a CNN in [125]. The accuracy ranges from 0.98 to 0.994 for the same brand and smartphone model for 74 devices. However, when a pool of 74 devices is utilized, the accuracy decreases to 0.475.

The use of CNN for PRNU characteristics and classification is explored in [67]. In [66], a CNN's noise-print extraction method combined with PRNU is utilized as a feature and the output of three classifiers, i.e., SVM, Likelihood-Ratio Test (LRT) and Fishers Linear Discriminant Analysis (FLD) are employed for classification. SVM achieves the highest accuracy of 0.952. A neural network based on ResNet101 and SVM is utilized in [68] to process PRNU derived from photos. This method achieves a 99.58% accuracy for 28 devices. EfficientNet, a CNN-based neural network, is discussed in [171]. This neural network achieves a 99.1% accuracy for 23,000 photos taken by 27 smartphone cameras. The authors in [126] uses CNN and RemNet. This method achieves an accuracy of 97.59% for 18 different cameras. In [172], the Ensemble classifier based on the demosaicing residual features extracted from the CFA filter is used for camera identification. For the identification of 68 cameras, the authors achieve an average accuracy of 98.14%. Additionally, the demosaicing approach for feature extraction is covered in [127]. A CNN is utilized for clustering, reaching an accuracy of more than 91% on 35 devices for WhatsApp photos and 95% for YouTube scenes. In [70], various pre-trained CNNs, including GoogleNet, SqueezeNet, Densenet201 and Mobilenetv2, are discussed. Eighteen smartphones were used to acquire 4,500 photos and the authors achieved an F1-score of more than 91%. In [135], the features retrieved using patchwise mean, variance scoring and K-means clustering are addressed. A Res2Net is employed for classification, reaching an accuracy of 92.62% for 74 cameras. The authors in [173] discuss a Multiscale Content-Independent Feature Fusion Network (MCIFFN).

In [73], the ensemble classifier is utilized after the features from the images are retrieved using DCT. The authors additionally employ Principal Component Analysis (PCA) to enhance the results. They achieve a 99.1% accuracy for 10,507 photos taken by ten cameras.

In [115], the Discrete Wavelet Transform (DWT) features are combined with nine classifiers Bayes Net (BN), Logistic (L), Logistic Model Tree (LMT), Multi-Layer Perceptron (MLP), Naive Bayes (NB), Naive Bayes Multinomial (NBM), Random Forest (RF), Simple Logistic (SL) and SVM. For the identification of 4 cameras, the average accuracy is 99.25%.

Other approaches

The authors in [54] use adaptive thresholding for camera identification. They obtain an intra-correlation between 0.46 and 0.7 and an inter-correlation between 0.1 and 0.45 for 74 cameras. The authors of [53] discuss correlation-based camera identification using PRNU. Eight hundred photos from the Dresden database, which contains images from 25 cameras, are used in the experiments.

For camera identification, SPN and correction are used in [48]. The authors applied spectral clustering and the Alternating Direction Method of Multipliers (ADMM) for clustering. They achieve an F1 score of between 0.90 and 0.97 for 31 cameras. In [61], the Locally Adaptive Discrete Cosine Transform (LADCT) and PRNU are used to identify cameras. The authors employ two datasets: their dataset with 13 cameras, for which they get FNRs ranging from 5.46% to 21.27% and FPRs ranging from 0.48% to 1.77%; and the Dresden dataset with ten cameras, for which they get FNRs ranging from 0.93% to 14.11% and FPRs ranging from 0.10% to 1.74%. In [174], SPN derived from the green channel using a high pass filter is explored. An FNR of 53% and FPR of 10.75% was obtained for five cameras. Additionally, SPN (Sensor Pattern Noise) and PRNU are applied to cluster 34 camera models in [69]. In [62] the characteristics derived from PRNU are used as input for a hierarchical search utilizing MapReduce. Mean accuracy of 91% was found for 1174 cameras. Large-scale sparse subspace clustering is utilized in [175] to identify cameras using the features collected utilizing the linear dependencies among SPN. Precision is 0.92, recall is 0.88, F1-score is 0.92 and ARI is 0.88 for 107 cameras. The works in [49], [51], [52], [57], [64] and [65] use of PRNU.

For feature extraction, the authors of [154] employed SPN approximation and for classification they used Markov clustering and a recently devised hybrid clustering technique. The precision is 0.997, recall is 0.765, F1-score is 0.866, ARI is 0.863 and purity is 0.997 for a dataset of 35 smartphones. in [176], the authors proposed a method where the cameras are grouped using a ranking index for each fingerprint. Precision is close to 1, recall is between 0.65 and 0.85 and the F1 score is between 0.7 and 0.9 for 10,960 images taken by 53 cameras.

In [74], the peaks are suppressed and the low-frequency SPN frequencies are removed from the images using DCT and the Spectrum Equalization Algorithm High-Frequency (SEA-HF). The TPR is 88.54% for 14,594 photos from 57 cameras. In [55], spatial domain averaged (SDA) frames are utilized. The authors in [46] use the Peak Signal to Noise Ratio (PSNR) for camera identification. In [47], PRNU computed using the Maximum Likelihood estimator is employed for feature extraction from pictures. With a FAR fixed at 10^{-5}, the FRR is between $9.6 * 10^{-2}$ and $8.4 * 10^{-15}$. A technique based on Gaussian blurring and eliminating the Least Significant Bit (LSB) from images is suggested in [56]. For 11,787 photos taken by 48 cameras, the authors obtain a correlation that is less than 0.075. The authors in [177] and [178] review various research works focused on camera source identification.

Chapter 3

Accelerometer-based device pairing and fingerprinting

In this chapter, the results published in [20] regarding mobile device pairing based on accelerometer data are presented. More recent results on fingerprinting devices from accelerometer characteristics, which are currently accepted for publication [24], are also included.

3.1 Accelerometer-based pairings, brief motivation

Smartphones and other small, mobile, or embedded IoT devices are already widely used. Thus it is urgent and crucial to encourage spontaneous connections between devices located in the same area. It makes sense to gather environmental data to obtain shared secure session keys between devices because it saves time and prevents the security disappointments of using weak passwords. While numerous research works concentrate on gathering environmental data to establish such connections, there are yet to be widespread practical implementations, indicating that more research in this area is encouraged.

This chapter examines a variety of transportation scenarios, including those involving people, bicycles, and motorized vehicles, among others. In order to verify that a secure session key can be recovered, it is important to measure the amount of entropy obtained from the accelerometer data in each setting. In particular, data is retrieved from the following typical categories of transportation environments: (i) cars, a mode of transportation that offers a high level of autonomy inside or outside cities; (ii) trams, the lighter form of rail transportation services, primarily used inside cities; (iii) trains, the more robust kind of rail passenger transit, (iv) bikes, an increasingly popular form of transportation inside cities that primarily relies on human power; and (v) walking, as it represents one of the most realistic scenarios. Since many works address device pairing based on shaking patterns. Device shaking as a baseline scenario is also included. Com-

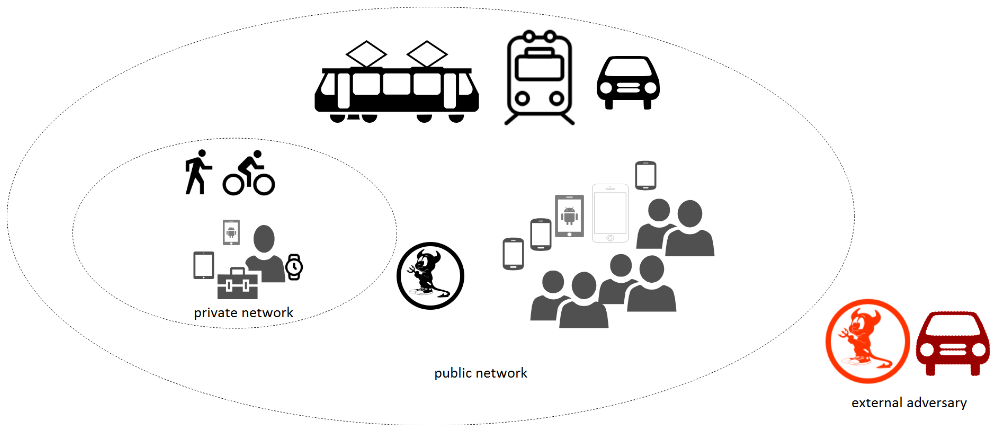


Figure 3.1: Overview of the proposed scenario with adversaries outside and inside the environment.

paring such static and transport scenarios highlights the impact on entropy extraction. Furthermore, users may unintentionally shake their smartphones while traveling in any of the previously mentioned conditions.

Figure 3.1 shows an abstract representation of the addressed scenario. A user caries mobile devices, i.e., smartphone, smartwatch, tablet, etc. which are in their *private network*. The user's private devices may connect based on accelerometer data collected from activities that are more personally meaningful to the user (like biking or walking), and they may share a rich history of accelerometer data. Using cryptographic ratcheting or other key continuity techniques might increase the security of bootstrapping a session key by taking advantage of the widely used history. On the other hand, the user may commute via train, tram, or vehicle and meet other users with whom they may build a *public network* for exchanging different types of data like contact details, pictures, and other media. An adversary may be present as a user within the same environment trying to compromise with the personal network or outside of the environment trying to compromise with the public network. Also, an adversary can use acceleration patterns from these settings to generate session keys for the public network. Both of these scenarios are shown in Figure 3.1, and more information on the adversary capabilities is provided in a subsequent section. In addition to a theoretical approximation of the adversary success rate, experiments are included in which mobile phones are placed in various locations inside a car or train. A car-following scenario, in which data is retrieved from a car that closely follows another one, is included in order to respond to this adversarial model. As people typically use multiple transportation modes to their destinations today, such an inquiry into multi-modal transportation is vital. Even so, individuals could desire to couple their mobile device with another one from the same transportation environment.

In Figure 3.2, two of the mobile phones which were used, i.e., an LG Optimus L7

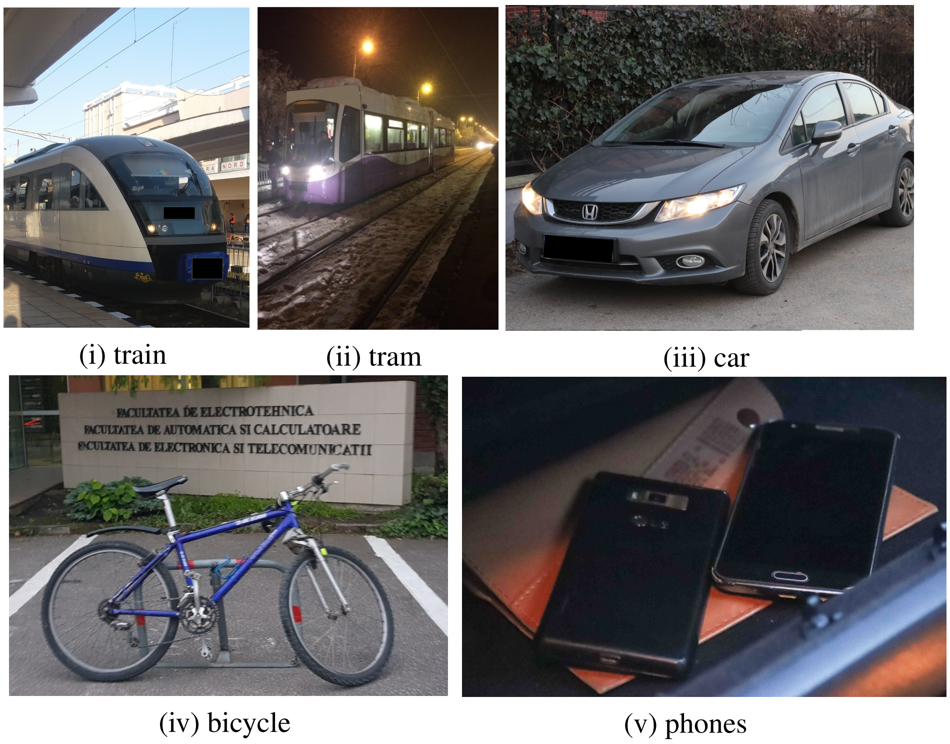

[redacted]
[redacted]
[redacted]
(i) train
(ii) tram
(iii) car
FACULTATEA DE ELECTROTEHNICA
FACULTATEA DE AUTOMATICA SI CALCULATOARE
FACULTATEA DE ELECTRONICA SI TELECOMUNICATII
(iv) bicycle
(v) phones

Figure 3.2: Several transportation modes used in the experiments (i-iv) and two of the phones (LG Optimus and Samsung J5) in the car inside the glove compartment (v)

P700 and a Samsung J5, are shown in a car's glove compartment, along with some of the transportation modes from which were gathered data for the tests. In several cases, a Samsung A3 was employed as a third phone. The characteristics of the three smartphones are compared in Table 3.1, showing differences in computational power and accelerometer features.

The technical difficulties in pairing devices based on accelerometer data are numerous, and multi-modal transportation makes the issue much more difficult. Now let's quickly go through a few issues. First, as might be predicted, different transportation settings result in diverse acceleration patterns. Data recorded inside a tram with data recorded when a person shakes the device are compared in Figure 3.3. Since accelerations inside the tram often occur over a brief period, the question of whether there is sufficient entropy to derive a secure session key arises immediately. Although device shaking is expected to provide a rich source of entropy, other transportation situations may offer considerably lower forms of entropy. Device shaking has already been proposed to extract or verify session keys (e.g., [179, 180, 181, 182]). Secondly, it is commonly known that data gathered from various accelerometers are not identical because of physical flaws

Table 3.1: Smartphones used in the experiments with the specifications for accelerometer sensors

Device	Android	CPU	Memory	Acceleration Sensor
LG Optimus P700	4.0.3	1.0 GHz Cortex-A5	4 GB, 512 MB RAM	BMA250 (Bosch Sensortec), Maximum Range - 39.22, Resolution - 0.038300782, Minimum Delay - 10 μs, Power - 0.03
Samsung J5	5.1.1	Quad-core 1.2 GHz Cortex-A53	8 GB, 1.5 GB RAM	K2HH (STM), Maximum Range - 39.2266, Resolution - 0.038307227, Minimum Delay - 10 μs, Power - 0.13
Samsung Galaxy A3	5.0.2	Quad-core 1.2 GHz Cortex-A53	16 GB, 1.5 GB RAM	BMC150 (Bosch Sensortec), Maximum Range - 19.6133, Resolution - 0.009576807, Minimum Delay - 10 μs, Power - 0.13

in the sensors. In fact, according to earlier studies, accelerometer sensors are so distinctive that it is possible to identify gadgets by their fingerprints using information gleaned from them [36]. The histogram of data obtained from an LG Optimus and a Samsung A3 in the same environment is displayed on the left side of Figure 3.4, and the histogram of data obtained from an LG Optimus and a Samsung J5 in the same environment in the right side of Figure 3.4. The Samsung A3 and LG Optimus include BOSCH BMC150 and BMA250 sensors, whereas the J5 has an STM K2HH sensor. The distributions for the identical manufacturer's sensors (on the A3 and Optimus) slightly differ. Although the J5's sensor is less sensitive and its histogram is much narrower than the Optimus and A3 (left side), the recorded values span a similar range. The inability of some devices to handle high collection rates is the third factor. The LG Optimus experienced significant drifts in the timestamps of the gathered data when the sampling rate was increased to 50Hz, which the sensor is supposed to allow. Drifts at 50Hz (20 ms per sample) are compared to drifts at 5Hz (200ms per sample) in Figure 3.5 . Since the phones do not sample data simultaneously, drifts of 50% to 100% are common and have a significant influence in the first scenario (i.e., at 50Hz for each drift of 20ms, one smartphone will remain one sample behind). The sample time is relatively consistent with a sampling rate of 5Hz (right side of the figure). Fortunately, most transportation modes generated slower variations in the shaking patterns than the shaking scenario, and this research demonstrated that even a modest 5Hz sampling rate is enough for most cases.

Various strategies are investigated for obtaining a common session key in response to this heterogeneity generated by environments, sensors, oscillators, or other device characteristics. High-pass and low-pass (smoothness) filters were employed and then sigma-delta modulation was used to extract feature vectors from the data. Finally, an encrypted key was extracted with the EKE protocol that provides provable security, [183], [184], and one of its variations, [185]. This secure key-exchange mechanism is guessing

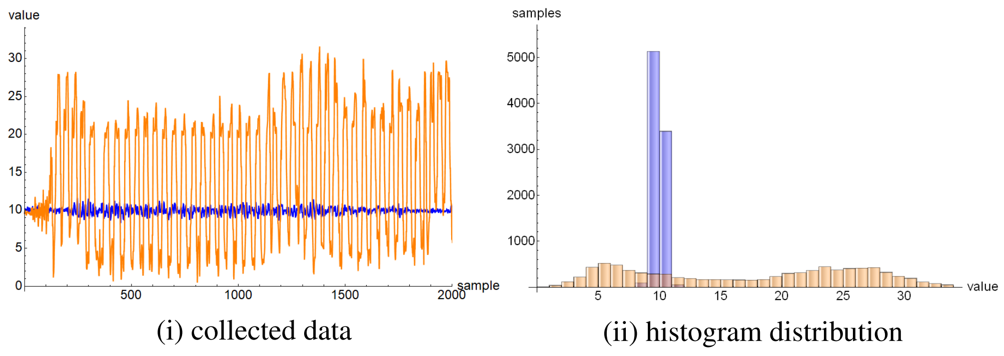


Figure 3.3: Data gathered inside a tram (blue) vs. data gathered during device shaking (orange) and histogram distribution (for LG Optimus smartphone)

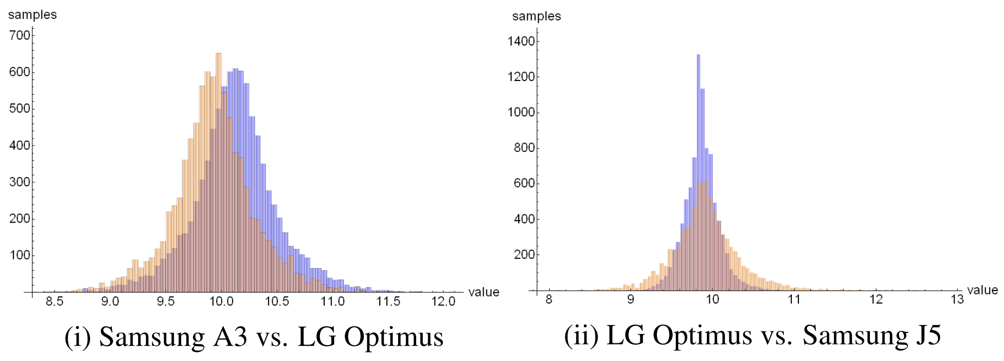


Figure 3.4: Histogram distribution of data gathered from Samsung A3, LG Optimus and Samsung J5 in tram

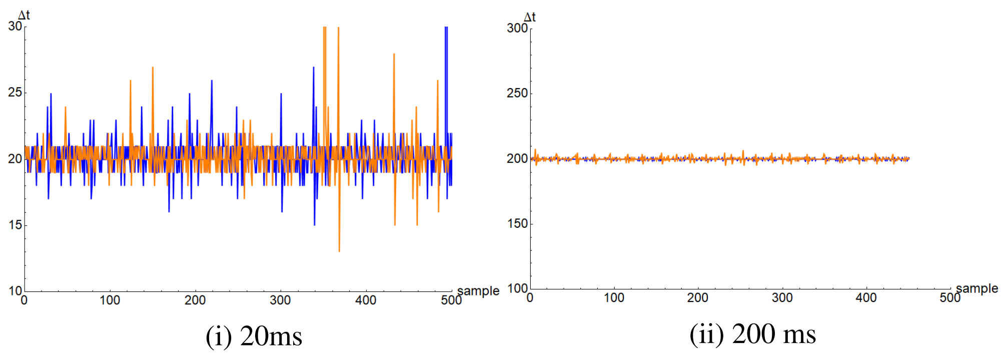


Figure 3.5: Drifts in the sampling rate on LG Optimus at 20ms and 200ms in tram

resilient.

A brief literature review focused on device-to-device (D2D) authentication, which is the primary area of interest in this chapter, is discussed next. For user-to-device (U2D) authentication methods, the most recent U2D authentication surveys [186, 187] are mentioned. The last decade has seen a general interest in device-to-device authentication. Various sources of common sensor data, such as light, sound, and others, are included in the surveyed pairing techniques [188, 189, 190, 191].

The Martini Synch protocol from [192] uses the accelerometer sensor for pairing two devices over Bluetooth and extracts $9 - 20$ bits of entropy per second. Also, in [193] pairing over Bluetooth is addressed for numerous wearable devices based on accelerometer data. However, this is more expensive and complex to build because it uses fuzzy cryptography. In a more recent proposal, in [194], fuzzy cryptography is used to generate common keys from accelerometer sensors. In [179], two accelerometer-based pairing techniques are covered. The Diffie-Hellman-based protocol is one and a symmetric function protocol is the other (the later version does not achieve guessing resilience). Key generation in [182] is carried out using the nearest-neighbor technique. In [195] and [110] methods of pairing for wearable devices based on accelerometer data are discussed. In addition to the acceleration sensor data, the work in [196] also adds audio data (by using microphones and speakers).

In [197], acceleration-based pairing is also discussed. The pairing action is mediated by a remote server (communication is done over TLS). The primary drawback of requiring internet connectivity is that the server first verifies that the datasets are the same before allowing the devices to interact with one another. Machine learning is utilized in [198] to create shared secret keys amongst devices. In a prior protocol proposal, [199], the vulnerability of low-entropy vectors in man-in-the-middle attacks was addressed. The work in [180] proposes a new methodology utilizing heuristic trees and hash functions.

This chapter accounts for several transit scenarios and includes explicit estimations of the extracted entropy. The concrete cryptographic pairing process used to create a pairwise (or group) key is partially orthogonal to this. As a result, a basic EKE-DH protocol is employed, which may be swapped out for variants that are either computationally easier (for low-end IoT devices) or more difficult (for dealing with alternative threat models).

In safe situations, accelerometer features can be used to identify the owner of two devices [200], identify driving patterns [147], identify anomalous driving behavior [148], [201], identify human activities [202]. It has also become more popular to distinguish between various forms of transportation, such as a car, bike, bus, etc., based on accelerometer data, [141], [139], [140], [203], [204]. The authors in [205] recently released a sizable dataset for online transportation mode recognition. However, these studies generally don't take into account pertinent opponent models. Robustness against malicious activities in the shaking scenario, which is utilized to analyze more advanced adversaries, has already been proven, [179, 180, 181]. Most of these earlier analyses lacked the analysis of the entropy of the shaking sequences, so the picture of the security properties of shake-based pairing is still incomplete. Sophisticated attacks to break the system, for example, exploiting technical support by automatically extracting acceleration from video or analyzing the entropy of the shaking sequences, are missing.

Accelerometer data is used in other fields of work, not always in the context of D2D authentication, to determine the location of the passenger [206] or to track the state of the roads [150], [151]. In [149], people are authenticated to the smartphone based on their walking manner using Gaussian Mixture Models.

3.2 Experimental analysis of accelerometer data

In this section, the tools for collecting and processing the data are presented. Nevertheless, the quality of the retrieved data is evaluated using entropy and Hamming distances.

Table 3.2: Summary of relevant notations

$\bar{\mathrm{v}}$	array with the data collected from the accelerometer
$\bar{\mathrm{v}}^0$	array with features of 0-Hamming distance from the accelerometer data
b	one byte of the data collected
t	the accelerometer array's time-stamp
Ham	Hamming distance
H	entropy
$\mathsf{H}_{\mathsf{min}}$	min entropy
p	probability of matching successfully one feature array
$Adv\mathcal{D}$	adversary is aware of the dispersion of feature arrays
$Adv\mathcal{H}$	adversary is aware of the dispersion of 0-Hamming arrays
Usr	honest user
$\epsilon_{\blacklozenge}^{k,n}$	advantage for at least k out of n matches, where $\blacklozenge \in \{Adv\mathcal{D}, Adv\mathcal{H}, Usr\}$
p	large prime number used for Diffie-Hellman Key exchange
g	a generator of Z_p

3.2.1 Tools for processing accelerometer data

The accelerometer data gathered from the smartphones are passed through the alignment, scaling, and filtering phases to generate feature vectors (for the key-exchange protocol). A high-pass filter or a low-pass filter is used at the filtering stage. A low-pass filter is employed using a moving average filter, commonly referred to as a smoothness filter. Additionally, the accelerometer data is expanded using sigma-delta modulation to enhance the recovered entropy. What follows is a technical discussion of these linked issues. Accelerometer data in Android is presented in a Cartesian format on the three axes. In the following discussion, the acceleration is used by combining the results on the three axes as $a = \sqrt{a_x^2 + a_y^2 + a_z^2}$ to overcome the orientation of the device. The notations are listed in Table 3.2.

Algorithm 1 High-pass filtering

```
1: procedure HIGH-PASS FILTER
2:     $\bar{\mathsf{v}} \leftarrow \{0\}^{length(\bar{\mathsf{v}})}$
3:     $\bar{\mathsf{v}}'[1] \leftarrow \bar{\mathsf{v}}[1]$
4:     for $i = 2, i \leq length(\bar{\mathsf{v}})$ do
5:         $\bar{\mathsf{v}}'[i] \leftarrow \alpha(\bar{\mathsf{v}}'[i] + \bar{\mathsf{v}}[i] - \bar{\mathsf{v}}[i-1])$
6:     end for
7:     return $\vec{\ }'$
8: end procedure
```

Figure 3.6: Algorithm for high-pass filtering

Temporal alignment. In order to prevent deviations from the real-time clock, loose time synchronization between the devices is necessary before moving on to the data gathering stage. The method of loose time synchronization involves just two easy steps between the principles:

$$A \to B : t_{A,0}, \mathrm{rnd}_{128}$$
$$B \to A : t_{B,0}, H(t_{A,0}, \mathrm{rnd}_{128}, t_{B,0})$$

Here, rnd_{128} stands for a 128-bit random number, and $t_{A,0}$ stands for the time on side A when sending the first message, and $t_{B,0}$ for the time on side B when sending the second message. The loose time-synchronization protocol lacks security features out of convenience and because it is outside the scope of the current work. The worst that may happen if an adversary tampers with it is a DoS attack due to the misaligned accelerometer data (this can be done by injecting fake data). However, security can be introduced to the time-synchronization protocol if a threat like this arises. Let $t_{A,1}$ represent the moment when A first received the second communication from B. The maximum synchronization error is then $\xi = t_{A,1} - t_{A,0}$, and for each message received at time $t_{A,k}$, by A, the clock on B's side is $t_{B,k} \in [t_{B,0} + t_{A,k} - t_{A,0}, t_{B,0} + t_{A,k} - t_{A,0} + \xi]$. The synchronization error ξ should typically be in the millisecond range. Even after this synchronization step, the sequences can occasionally have an offset of 2 or even 1 sample. The devices can only share the first series of collected data in plaintext to eliminate this offset, making it easy to identify and correct.

High-pass and smoothness filters. Both high-pass and smoothness filters were put to the test. High-pass filters were chosen because, in theory, they are the best at removing noise from accelerometer data, increasing the likelihood that fluctuations in the accelerometer data will appear in the key-exchange material. The idea behind smoothness filters is the opposite: by using them, the undesirable noise from the data may be reduced, which increases the likelihood of a successful pairing (as expected, however, this has a slight negative impact on security). Figure 3.6 shows the high-pass filtering algorithm. This method is the best for high-pass filtering since it enhances the increase with the gap between the values in the original signal, or $\overline{v}[i] - \overline{v}[i-1]$, at each step and increases the value in the filtered signal proportionally with α.

Algorithm 2 Sigma-delta modulation

```
 1: procedure SIGMA-DELTA MODULATION
 2:     $\sigma \leftarrow 0.04$ , $\delta \leftarrow 10$ , $\epsilon \leftarrow 0.01$
 3:     $\widetilde{b} \leftarrow \{\}$ , $index \leftarrow 1$ , $b \leftarrow 0$ , $t \leftarrow 0$
 4:     while $t < \max(\overline{\theta})$ do
 5:         $i \leftarrow index$
 6:         while $t > \overline{\theta}[i+1] \wedge i < |tst|$ do
 7:             $i \leftarrow i + 1$
 8:         end while
 9:         if $|\overline{v}[i] - \widetilde{v}| < \epsilon$ then $b = 0$
10:         else
11:             if $\overline{v}[i] > \widetilde{v}$ then $b = 1$
12:             else $b = -1$
13:             end if
14:         end if
15:         $\widetilde{b} = \text{Append}(\widetilde{b}, b)$
16:         $\widetilde{v} = \widetilde{v} + b\sigma$
17:         $t = t + \delta$
18:     end while
19: end procedure
```

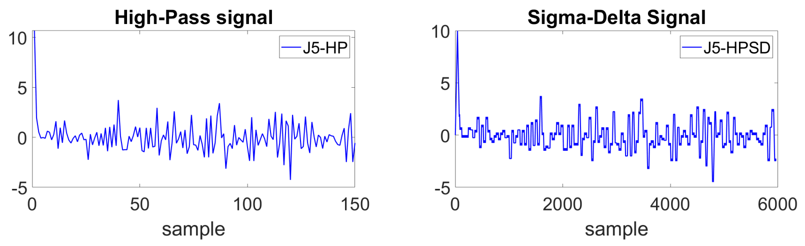
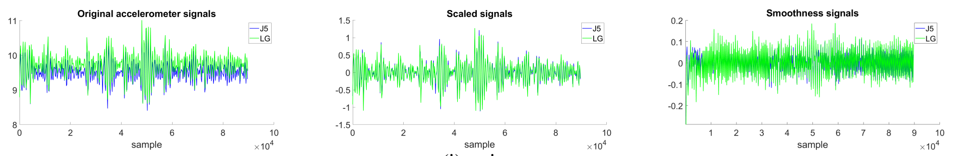
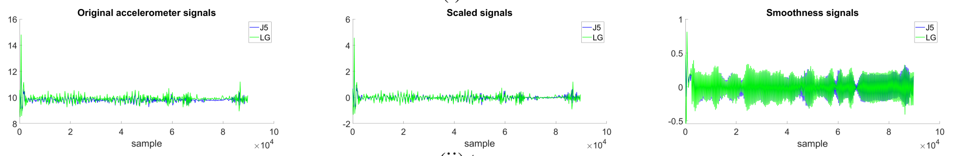


Figure 3.7: Algorithm for sigma-delta modulation (up), signal before and after (down)

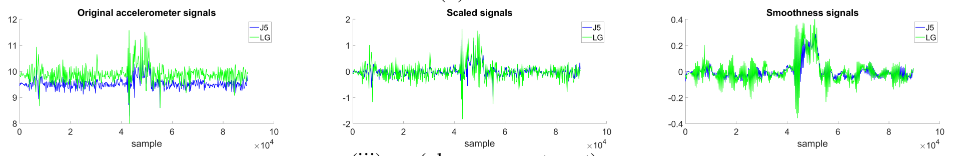
Sigma-delta modulation. The signal must be modulated to extract bits from the samples provided by the accelerometer. A ternary sigma-delta modulation is used, in which the signal is replaced with a 0 at each step if the modulated signal is sufficiently near to the original and a ± 1 in all other cases, i.e., when the modulated signal is smaller or larger. Sigma-delta modulation allows us to obtain 40 additional samples for each original sample by resampling at a rate of $\delta = 5ms$ over the initial acquisition rate of $200ms$. As a result, on the second plot 150 sampling points are multiplied by 40, $40 \times 150 = 6000$. The sigma-delta modulation algorithm is shown in Figure 3.7, and the original signal and the result are shown below. By experimenting, it was determined that $delta = 10$ and $epsilon = 0.01$ are suitable values. The value of σ was set as the signal's standard deviation multiplied by 0.1. When sigma-delta modulation is not employed, the signal is centered around the mean, and a ± 1 is extracted by taking the sign of each sample. Sigma-delta is preferred since this method reduces the number of extracted bits.



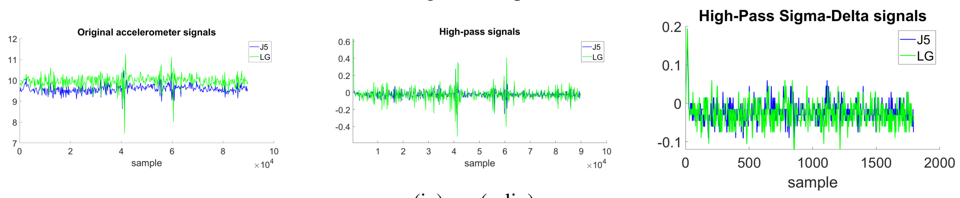
(i) train



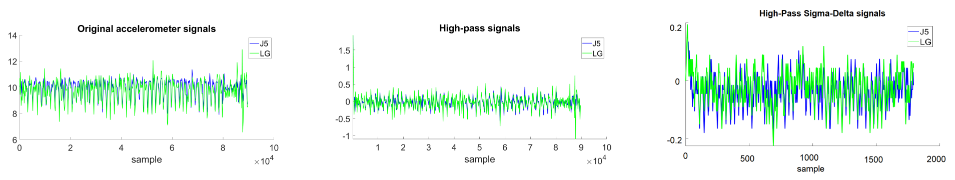
(ii) tram



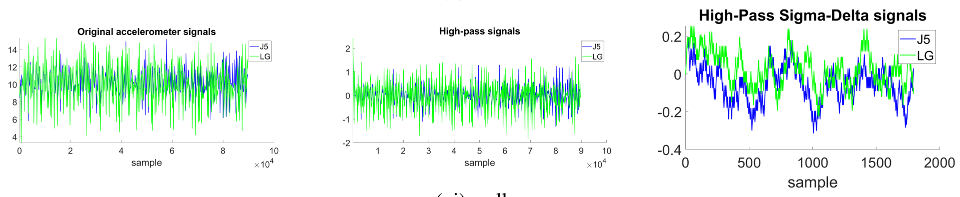
(iii) car (glove compartment)



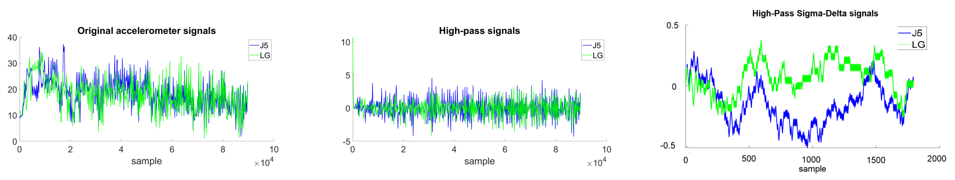
(iv) car (aslip)



(v) bike



(vi) walk



(vii) shake

Figure 3.8: Original accelerometer signals (left), processed signals using different methods (middle and right)

The plots in Figure 3.8 display the original signal (left) and various processing methods (middle and right). All seven experimental scenarios, i.e., train, tram, vehicle, bike, walk, and device shaking, are depicted in the plots. The scaled signal resembles the original one in shape. The signal is more consistent when smoothness filters are used, but high-pass filtering amplifies high frequency noise. The last plots show that the signal takes on a more distinct structure due to the Sigma-Delta modulation.

3.2.2 Theoretical framework for security analysis

The adversary model makes the reasonable assumption that there is a normal Dolev-Yao [207] adversary with complete control over the communication channel. On-device adversaries with access to high-quality accelerometer data streams, e.g., through malware running in the background while a man-in-the-middle attack is being conducted concurrently, are outside the scope of the current work because they can easily extract the key like the legitimate application. As a result, a precise metric for the level of security of the samples is required. However, future studies may examine various on-device adversary types, such as Javascript from malicious websites sampling accelerometers at lower precision or sample rates. This analysis could provide protection even against on-device adversaries with lower fidelity.

Another threat could be a stalker who follows the trustworthy user to get enough accelerometer data to discover the shared secret key across his devices. Devices that display the same accelerations and share the same environment are not immediately seen as competitors but as members of the public peers. As mentioned in the introduction, these should be distinguished from the devices used by the same person based on specific movements or a shared history. A car-following scenario is also included in this studies and demonstrate that the data does not support the natural pairing of devices from different cars, even if they are only a few meters apart.

Devices from the same environment will detect the same acceleration patterns. The user may travel on the tram, train or move his backpack inside the car. However, separation should happen when the user moves his backpack inside the environment, adding particular shaking patterns. The phones from the same context may couple, essentially creating a user's *public* network. On the other hand, the user's devices are connected to their *private* network. A clear distinction between public networks (which may include opponents) and private networks is not necessary for the scope of this chapter.

The current cryptographic models are used as security metrics. In more specific terms, the *guessing probability* of a random variable, X, which stands for one byte of data obtained from accelerometers, i.e., $\gamma(X) = \max\{\Pr[X = b] : b \in [0, 2^8 - 1]\}$ is employed. The *min entropy* of variable X can be calculated directly from this measurement as $\log_2(1/\gamma)$. For more information, see [208]. The security of accelerometer data is examined in the following sections using these measurements.

When the same vector is recorded on both phones, a portion of the key can be recovered thanks to the pairing approach, which depends on the feature vector from accelerometer data of Hamming distance equal to 0. The bits produced by the sigma-delta modulation or the sign of each sample, represented as 0 or 1, respectively, for negative vs. positive, make up the feature vectors. Based on experimental data, the success chance for two genuine parties to extract one such vector can be calculated as follows:

$$p_{Usr} = \frac{|\bar{\mathsf{v}}^0|}{|\bar{\mathsf{v}}|},$$

where $\bar{\mathsf{v}} = \{b_1, b_2, ..., b_\ell\}$ refers to the array of all bytes collected and $\bar{\mathsf{v}}^0 = \{b_1^0, b_2^0, ..., b_{\ell'}^0\}$ the array of all bytes with a Hamming-distance of zero. This leads to $p_{Usr} = \ell'/\ell$.

The probability of predicting a byte b from $\bar{\mathsf{v}}^0$ can alternatively be expressed as follows:

$$\gamma = \max\{\Pr[b = b_i] : i = 1..\ell', b_i \in \bar{\mathsf{v}}^0\}.$$

The minimal entropy of the feature array with a 0-Hamming distance $\bar{\mathsf{v}}^0$ is defined based on this probability as follows:

$$\mathsf{H}_{\mathsf{min}}^{\bar{\mathsf{v}}^0} = \sum_{i=1,\ell'} \log_2(\gamma) = \ell' \log_2(\gamma).$$

The total entropy of the feature vectors, which is also frequently utilized in many related works, will be added to these as well. This is computed as:

$$\mathsf{H}^{\bar{\mathsf{v}}^0} = -\sum_{i=1,\ell'} p_i \log_2(p_i), p_i = \Pr\left[b = b_i, b_i \in \bar{\mathsf{v}}^0\right].$$

Measuring the adversary advantage (success probability) is important to determine the security level of the protocol. A model of the adversary is required to quantify this. The most basic presumption is that the adversary can only infer that each sample has 8 bits (this would trivially suggest that predicting one value has probability 2^{-8}). However, this adversary model is inadequate, particularly when the devices may communicate some samples in cleartext for calibration reasons, giving the adversary access to at least a small dataset. On the other side, the adversary can carry past datasets from the same area and possess similar instruments. One of the most compelling hypotheses about the adversary seems to be that he has access to the set of samples that produce a 0 Hamming distance, but he is unaware of the eventual selection of the samples. Let's call this adversary $Adv\mathcal{H}$.

In more detail, assuming that the feature vectors as arrays $\bar{\mathsf{v}}^0 = \{b_1^0, b_2^0, ..., b_{\ell'}^0\}$ of ℓ' bytes, let $b_{\max}$ be the byte occurring in $\bar{\mathsf{v}}^0$ with the highest probability and $\ell_{\max}$ be the number of times it occurs there (if there are multiple such values, $\ell_{\max}$ is the same for all). If the opponent utilizes this byte to make his best guess, then he has the following advantage:

$$p_{Adv\mathcal{H}} = \frac{\ell_{\max}}{\ell}.$$

This suggests that the attacker is unaware of the current accelerometer value and that his only option is to predict the value using his knowledge of the bytes in $\overline{\mathrm{v}}^0$, which has a greater success rate.

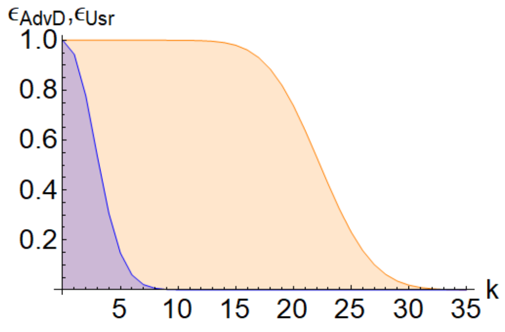
A less strong assumption about the adversary would be that it does not have access to the vectors that produce a Hamming distance of zero, but that it does have access to all of the acceleration vectors $\overline{\mathrm{v}}$ and assumes that the value chosen to have a zero Hamming distance will be the one occurring with the highest probability in $\overline{\mathrm{v}}$. Let's use the symbol $Adv\mathcal{D}$ to identify this. But keep in mind that while there can be several values with the same greater probability in the feature vector $\overline{\mathrm{v}}$, it is possible that these values do not have the same chance of happening in the collection of features with the same Hamming distance. Adversary $Adv\mathcal{D}$ is similar to $Adv\mathcal{H}$ if it chooses the value with the highest probability that also appears in $\overline{\mathrm{v}}^0 0$. It is assumed that $Adv\mathcal{D}$ will choose the value with the highest probability in $\overline{\mathrm{v}}$ array, but the lowest in $\overline{\mathrm{v}}^0$ array to distinguish between the two sorts of adversaries and derive a lower and upper bound for security. If $\overline{\mathrm{v}}_{\max}$ is a vector of byte values with the highest chance of occurring in $\overline{\mathrm{v}}$, then it can be affirmed that $Adv\mathcal{D}$ was successful if and only if:

$$p_{Adv\mathcal{D}} = \min\{\Pr[b = b_{\max}] : b \in \overline{\mathrm{v}}^0, b_{\max} \in \overline{\mathrm{v}}_{\max}\}.$$

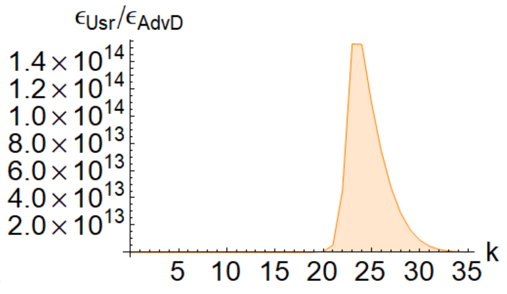
Matching in at least k out of ℓ trials (where ℓ is the total number of bytes collected from the accelerometer) is required to increase the pairing security process. The result is the following two security constraints for genuine users and adversaries:

$$\epsilon_{\blacklozenge}^{k,\ell} = \sum_{i=k,\ell} \binom{\ell}{i} p_{\blacklozenge}^{i}(1-p_{\blacklozenge})^{\ell-i} = 1 - \sum_{i=0,k-1} \binom{\ell}{i} p_{\blacklozenge}^{i}(1-p_{\blacklozenge})^{\ell-i}.$$

Where $\blacklozenge \in \{Adv\mathcal{D}, Adv\mathcal{H}, Usr\}$. The application of these security bounds is explained next, using some real-world situations that ran into both advantageous and adverse circumstances. Experimental data will be covered in more detail in the following section. It is used to show how honest users and adversaries benefit from scaling up. First, it was discovered that honest users had a success rate of $p_{Usr} = 0.39$ and the adversary had a significantly lower success rate of $p_{Adv\mathcal{D}} = 0.05$ when their phones were placed next to each other in a train. It was collecting 56 8-bit vectors per second at a sample rate of 200 ms (i.e., $8 \times 56 \approx 90s$). Setting $k = 9$ results in the honest user success rate being $\epsilon_{Usr}^{9,56} = 99.99\%$ whereas the success rate of an adversary is $\epsilon_{Adv\mathcal{H}}^{9,56} = 0.17\%$. Figure 3.9 depicts the ratio of fair users to the adversary's success rate, as well as the plot for the success rate over a larger range of k. One of the smartphones (the LG Optimus) had significant clock drifts during sampling in an unfavorable circumstance where data was taken at 20ms intervals while walking. This could explain why the success probabilities are only $p_{Usr} = 0.05$ and $p_{Adv\mathcal{H}} = 0.01$. With appropriate parameter tweaking, the prior bounds show that fair users still pair at a higher probability than dishonest adversaries. A total of 541 samples are collected in 90s because sampling is done at 20 ms, or $20 \times 8 \times 541 \approx 90s$. When $k = 14$, was obtained $\epsilon^{14,541}_{Usr} = 99.82\%$ and $\epsilon^{14,541}_{Adv\mathcal{H}} = 0.13\%$. Although it will necessitate trading more feature vectors, this is similar to the previous favorable case. Figure 3.10 shows the ratio between the fair users and the adversary's success rate, as well as the plot for the success rate over a larger range of k. So, in theory, bootstrapping a safe session key is possible as long as the ratio of honest user success rates exceeds the ratio of adversary success rates.

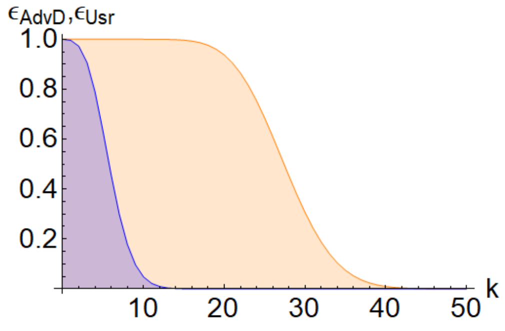


(i) success rates

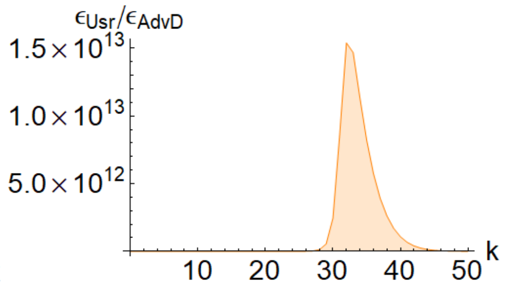


(ii) ratio of success rates

Figure 3.9: Honest users vs. adversaries based on data gathered in train in a favorable scenario $p_{Usr} = 0.39$, $p_{Adv\mathcal{D}} = 0.05$: success probabilities $\epsilon^{56,\ell}_{Usr}$, $\epsilon^{56,\ell}_{Adv\mathcal{D}}$ (left) and ratio of the success probabilities $\epsilon^{56,\ell}_{Usr}/\epsilon^{56,\ell}_{Adv\mathcal{D}}$ (right)

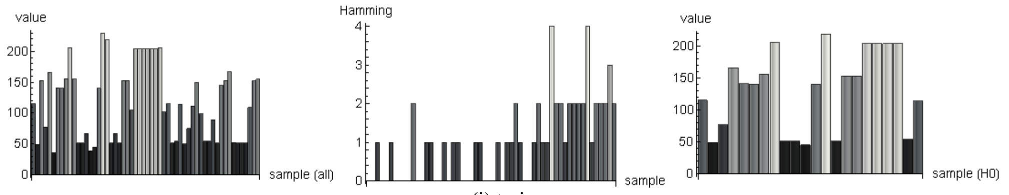


(i) success rates

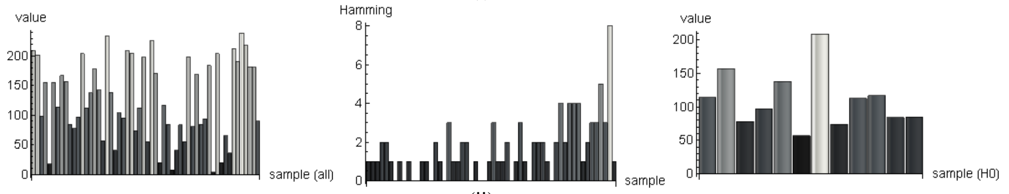


(ii) ratio of success rates

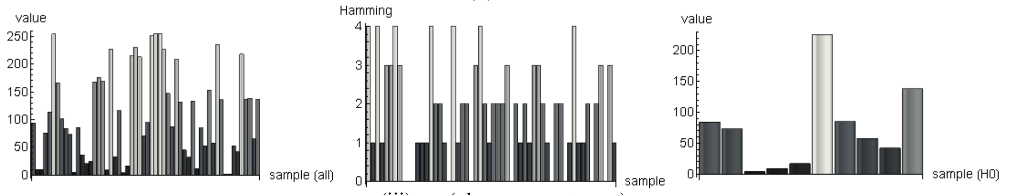
Figure 3.10: Honest users vs. adversaries based on data extracted during walking in a less favorable scenario $p_{Usr} = 0.05$, $p_{Adv\mathcal{D}} = 0.02$: success probabilities $\epsilon^{k,541}_{Usr}$, $\epsilon^{k,541}_{Adv\mathcal{D}}$ (left) and ratio of the success probabilities $\epsilon^{k,541}_{Usr}/\epsilon^{k,541}_{Adv\mathcal{D}}$ (right)



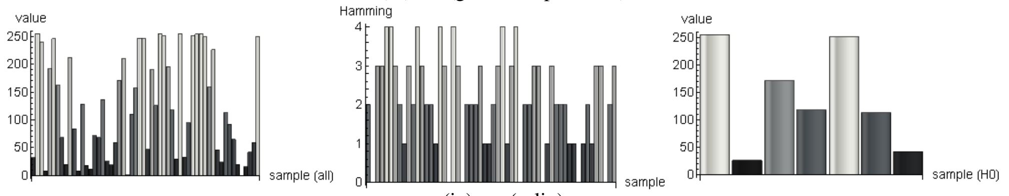
(i) train



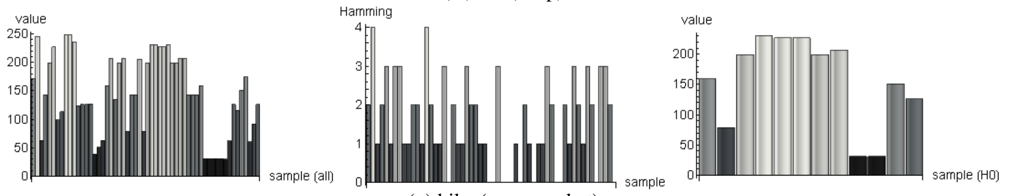
(ii) tram



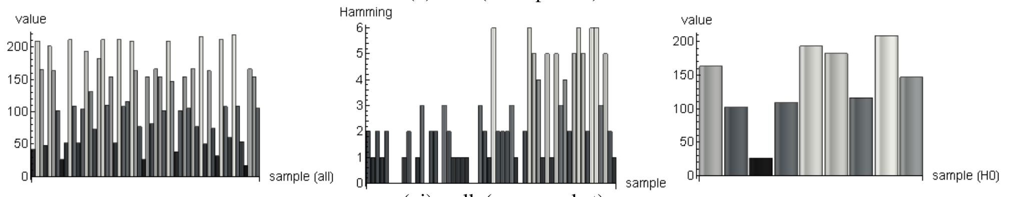
(iii) car (glove compartment)



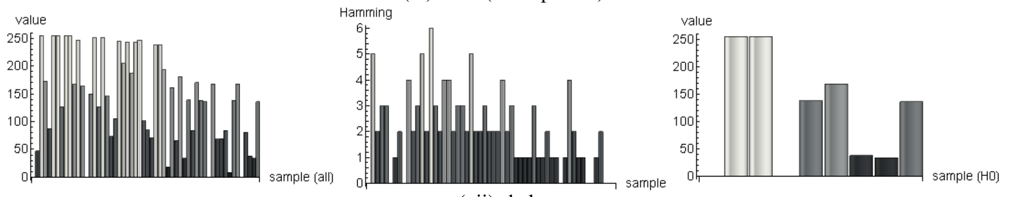
(iv) car (aslip)



(v) bike (same pocket)

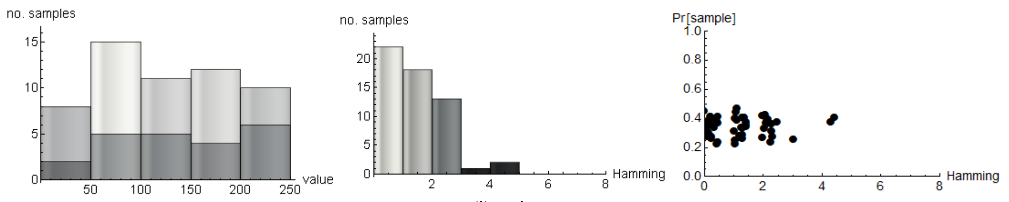


(vi) walk (same pocket)

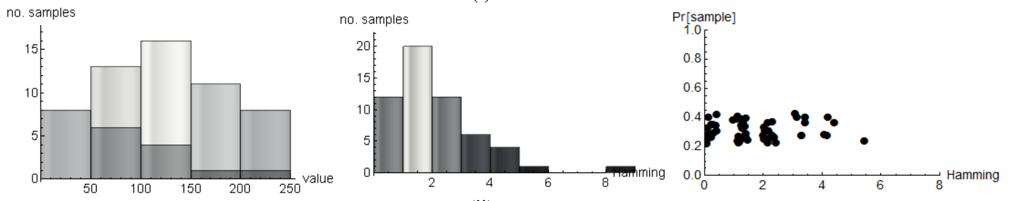


(vii) shake

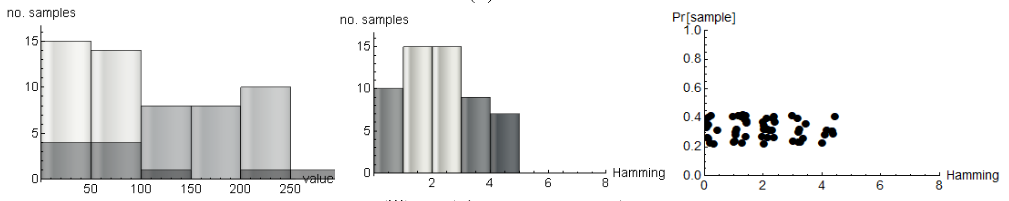
Figure 3.11: Accelerometer sample values as collected on J5 (left), Hamming-distance to LG (middle) and accelerometer values of select samples (matched on J5 and LG) for the five use cases: train (i), tram (ii), car (iii, iv), bike (v), walk (vi) and shake (vii) (right)



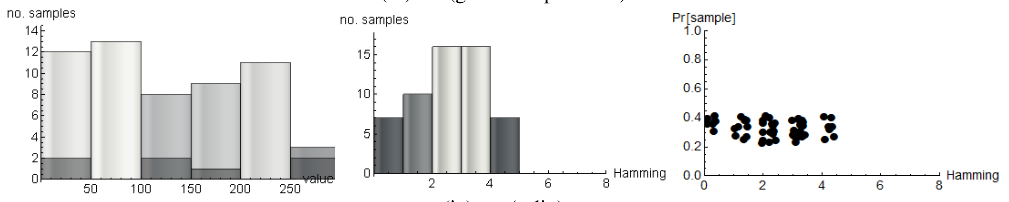
(i) train



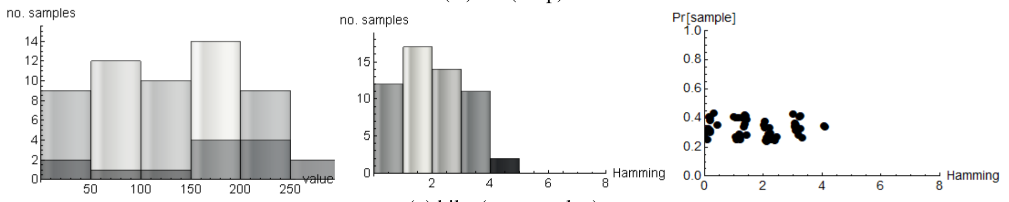
(ii) tram



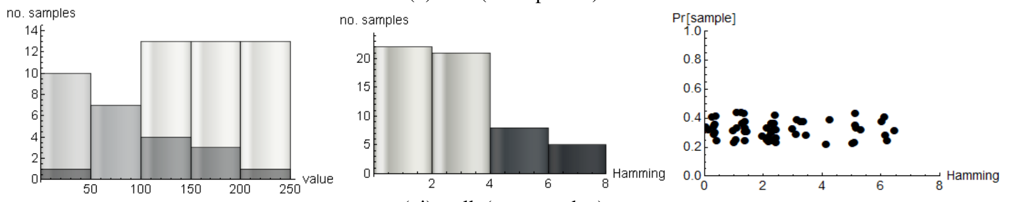
(iii) car (glove compartment)



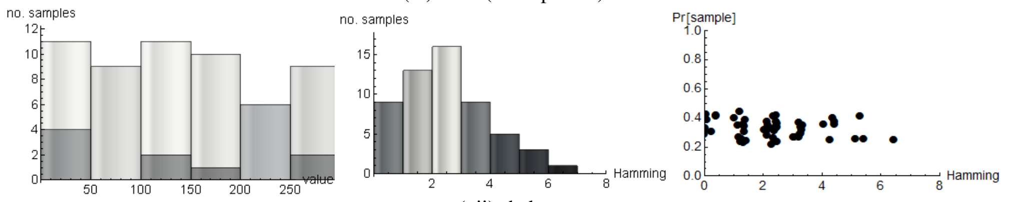
(iv) car (aslip)



(v) bike (same pocket)



(vi) walk (same pocket)



(vii) shake

Figure 3.12: Histogram distribution of acceleration values (left) Hamming distances (middle) and guessing probability (right) as recorded on Samsung J5 for the five use cases: train (i), tram (ii), car (glove compartment) (iii), anti-slip (iv), bike (same pocket) (v), walk (vi) and shake (vii) (simple scaling and sign extraction was applied to the signals)

Table 3.3: Summary of experimental results using simple scaling (SS), high-pass (HP) and smoothness filtering (for 90 seconds of data collected with LG Optimus)

Proc.	Env.	$p_{Adv\mathcal{D}}$	$p_{Adv\mathcal{H}}$	p_{Usr}	$\epsilon^{\ell/16}_{Adv\mathcal{D}}$	$\epsilon^{\ell/16}_{Adv\mathcal{H}}$	$\epsilon^{\ell/16}_{Usr}$	$\epsilon^{\ell/8}_{Adv\mathcal{D}}$	$\epsilon^{\ell/8}_{Adv\mathcal{H}}$	$\epsilon^{\ell/8}_{Usr}$	$\mathsf{H}_{\mathsf{min}}$	H	$\mu(\mathrm{Ham})$	$\lvert\bar{v}_0\rvert$	ℓ
SS	Train	0.07	0.07	0.39	0.77	0.77	1.00	0.10	0.10	1.00	2.46	0.90	0.98	22	56
SS	Tram	0.02	0.02	0.21	0.08	0.08	1.00	$< 10^{-4}$	$< 10^{-4}$	0.97	3.58	0.48	1.63	12	56
SS	CarG	0.02	0.02	0.18	0.08	0.08	1.00	$< 10^{-4}$	$< 10^{-4}$	0.89	3.32	0.37	1.79	10	56
SS	CarA	0.00	0.02	0.13	0.00	0.08	0.98	0.00	$< 10^{-4}$	0.56	2.81	0.22	2.11	7	56
SS	Bike1	0.04	0.04	0.21	0.32	0.32	1.00	0.00	0.00	0.97	2.58	0.41	1.54	12	56
SS	Walk1	0.00	0.02	0.16	0.00	0.08	1.00	0.00	$< 10^{-4}$	0.82	3.17	0.32	2.30	9	56
SS	Shake	0.04	0.04	0.16	0.32	0.32	1.00	0.00	0.00	0.82	2.17	0.27	2.02	9	56
HP	Train	0.02	0.04	0.30	0.08	0.32	1.00	$< 10^{-4}$	0.00	1.00	3.09	0.71	1.05	17	56
HP	Tram	0.09	0.09	0.36	0.89	0.89	1.00	0.23	0.23	1.00	2.00	0.78	1.20	20	56
HP	CarG	0.02	0.02	0.21	0.08	0.08	1.00	$< 10^{-4}$	$< 10^{-4}$	0.97	3.58	0.48	1.36	12	56
HP	CarA	0.00	0.04	0.16	0.00	0.32	1.00	0.00	0.00	0.82	2.17	0.29	1.59	9	56
HP	Bike1	0.02	0.02	0.11	0.08	0.08	0.95	$< 10^{-4}$	$< 10^{-4}$	0.39	2.58	0.17	1.79	6	56
HP	Walk1	0.02	0.02	0.16	0.08	0.08	1.00	$< 10^{-4}$	$< 10^{-4}$	0.82	3.17	0.32	2.50	9	56
HP	Shake	0.09	0.09	0.21	0.89	0.89	1.00	0.23	0.23	0.97	1.26	0.35	2.41	12	56
SM	Train	0.77	0.77	0.84	1.00	1.00	1.00	1.00	1.00	1.00	0.13	0.31	0.18	47	56
SM	Tram	0.95	0.95	0.96	1.00	1.00	1.00	1.00	1.00	1.00	0.03	0.08	0.05	54	56
SM	CarG	0.09	0.11	0.39	0.89	0.95	1.00	0.23	0.39	1.00	1.87	0.65	1.25	22	56
SM	CarA	0.07	0.11	0.32	0.77	0.95	1.00	0.10	0.39	1.00	1.58	0.48	1.63	18	56
SM	Bike1	0.68	0.68	0.68	1.00	1.00	1.00	1.00	1.00	1.00	0.00	0.00	0.45	38	56
SM	Walk1	0.34	0.34	0.39	1.00	1.00	1.00	1.00	1.00	1.00	0.21	0.19	2.05	22	56
SM	Shake	0.18	0.18	0.41	1.00	1.00	1.00	0.89	0.89	1.00	1.20	0.51	1.32	23	56

Table 3.4: Summary of experimental results using sigma-delta modulation over simple scaling (SS), high-pass (HP) and smoothness filtering for 90 seconds of data collected with Samsung J5)

Proc.	Env.	$p_{Adv\mathcal{D}}$	$p_{Adv\mathcal{H}}$	p_{Usr}	$\epsilon^{\ell/16}_{Adv\mathcal{D}}$	$\epsilon^{\ell/16}_{Adv\mathcal{H}}$	$\epsilon^{\ell/16}_{Usr}$	$\epsilon^{\ell/8}_{Adv\mathcal{D}}$	$\epsilon^{\ell/8}_{Adv\mathcal{H}}$	$\epsilon^{\ell/8}_{Usr}$	$\mathsf{H}_{\mathsf{min}}$	H	$\mu(\mathrm{Ham})$	$\lvert\bar{v}_0\rvert$	ℓ
SSSD	Train	0.06	0.09	0.27	0.55	1.00	1.00	$< 10^{-26}$	$< 10^{-7}$	1.00	1.56	14.04	2.85	611	2240
SSSD	Tram	0.10	0.10	0.23	1.00	1.00	1.00	$< 10^{-5}$	$< 10^{-5}$	1.00	1.20	12.42	3.19	508	2240
SSSD	CarG	0.10	0.10	0.21	1.00	1.00	1.00	$< 10^{-6}$	$< 10^{-6}$	1.00	1.11	10.68	3.27	462	2240
SSSD	CarA	0.15	0.15	0.23	1.00	1.00	1.00	1.00	1.00	1.00	0.65	9.28	3.17	521	2240
SSSD	Bike1	0.06	0.06	0.26	0.48	0.65	1.00	$< 10^{-27}$	$< 10^{-25}$	1.00	2.00	15.38	2.42	576	2240
SSSD	Walk1	0.05	0.07	0.26	0.01	0.96	1.00	$< 10^{-40}$	$< 10^{-19}$	1.00	1.87	14.57	2.01	586	2240
SSSD	Shake	0.06	0.06	0.19	0.17	0.17	1.00	$< 10^{-32}$	$< 10^{-32}$	1.00	1.76	10.08	2.05	436	2240
HPSD	Train	0.11	0.11	0.23	1.00	1.00	1.00	0.03	0.03	1.00	1.02	11.39	3.43	507	2240
HPSD	Tram	0.07	0.08	0.23	0.94	1.00	1.00	$< 10^{-19}$	$< 10^{-14}$	1.00	1.55	10.25	3.42	510	2240
HPSD	CarG	0.11	0.11	0.24	1.00	1.00	1.00	0.00	0.00	1.00	1.18	11.30	3.47	536	2240
HPSD	CarA	0.10	0.10	0.24	1.00	1.00	1.00	0.00	0.00	1.00	1.22	11.05	3.46	548	2240
HPSD	Bike1	0.06	0.06	0.19	0.22	0.48	1.00	$< 10^{-31}$	$< 10^{-27}$	1.00	1.63	10.38	3.34	431	2240
HPSD	Walk1	0.05	0.07	0.24	$< 10^{-3}$	0.98	1.00	$< 10^{-48}$	$< 10^{-17}$	1.00	1.69	14.05	2.57	530	2240
HPSD	Shake	0.08	0.08	0.23	1.00	1.00	1.00	$< 10^{-13}$	$< 10^{-11}$	1.00	1.44	11.31	2.17	506	2240
SMSD	Train	0.22	0.22	0.26	1.00	1.00	1.00	1.00	1.00	1.00	0.27	6.68	2.93	581	2240
SMSD	Tram	0.03	0.05	0.16	$< 10^{-11}$	$< 10^{-4}$	1.00	$< 10^{-77}$	$< 10^{-49}$	1.00	1.80	10.01	3.06	358	2240
SMSD	CarG	0.46	0.46	0.48	1.00	1.00	1.00	1.00	1.00	1.00	0.08	4.55	1.85	1083	2240
SMSD	CarA	0.44	0.44	0.47	1.00	1.00	1.00	1.00	1.00	1.00	0.10	5.81	1.82	1060	2240
SMSD	Bike1	0.02	0.03	0.10	$< 10^{-40}$	$< 10^{-12}$	1.00	$< 10^{-151}$	$< 10^{-79}$	$< 10^{-4}$	1.60	6.19	3.15	222	2240
SMSD	Walk1	0.06	0.07	0.25	0.65	0.81	1.00	$< 10^{-25}$	$< 10^{-23}$	1.00	1.88	14.44	2.45	551	2240
SMSD	Shake	0.10	0.10	0.22	1.00	1.00	1.00	0.00	0.00	1.00	1.07	7.82	2.23	490	2240

3.2.3 Analyzing experimental data

A detailed explanation of the experimental results is discussed following. First, the values gathered from the seven usage cases are shown in Figure 3.11: train, tram, vehicle (glove compartment and anti-slip), bike, walk, and shake. Accelerometer sample value measured on the J5 (left), Hamming distance to the LG (center), and selected accelerometer sample values (matched on the J5 and the LG) (right). The bytes appear to be distributed rather uniformly, which is advantageous for security. In Figure 3.12, the distribution of acceleration values in a histogram (left), Hamming distances (middle), and guessing probability (right) are illustrated. The worst results are from the car and the shaking process, while the train and tram have noticeably lesser Hamming distances. The probabilities are quite uniform, indicating that vectors with a Hamming distance of 0 should be somewhat difficult to differentiate from the other vectors (which is desirable for security reasons).

The findings for all experiments and filtering mechanisms are compiled in Tables 3.3 and 3.4. The results only from one smartphone are provided because they are comparable to the other. Sigma-delta modulation increases the number of samples (due to time division by the parameter δ). Thus it can be differentiated from simple filtering procedures.

It can be seen that smoothness reduces the entropy to 2 bits, and occasionally even to 0, using the filtering algorithms in Table 3.3. In some instances, this results in the guessing adversary winning with a chance of 1. Smoothness does not appear to benefit security. The devices will, in fact, always be properly paired when smoothness is used, i.e., $p_{agen} = 1$. High-pass filtering and simple scaling appear to produce comparable results. High-pass filtering causes a slight reduction in pairing probability because it is more sensitive to changes in the signal. High-pass can occasionally cause a slight boost in byte entropy, but it can also cause entropy losses in other circumstances. However, the success probability of honest parties remains high (82% to 100% in the situation where the 7 out of 56 limits are applied, or $\ell/8$), and the minimal entropy $\mathsf{H}_{\mathsf{min}}$ is often 1-2 bits/byte. The two exceptions are the car anti-slip scenario under simple scaling and the bike scenario with high-pass filtering. Here, the likelihood of success for honest parties drops to 0.56 and 0.39, respectively. Switching to the $\ell/16$ limit, which calls for at least 4 matches rather than 8, is the solution. In this scenario, the likelihood of honest users rises to 95% while the likelihood of the adversary remains at 7%. Figures 3.13 and 3.14 provide a graphic description of all the tests. The likelihood of identical vectors is depicted in the first picture, while the mean Hamming distance across all experiments and processing techniques is depicted in the second. Using the smoothness filters increases the likelihood of matching. In this regard, sigma-delta modulation is also beneficial. The habitats of trains and trams are likewise the best for matching feature vectors, but it should be noted that these should also be more vulnerable to attackers who might gather information from the same environment.

Pointing out that these values were taken from accelerometer data collected for 90 seconds; nevertheless, over a longer time (a few minutes), certain parameters (k, n) might be chosen to achieve insignificant success rates for the adversary. Additionally, the entropy gathered every second is shown in the H column. This figure is a little low at fewer than one bit per second. This is expected given that the sign of the samples is simply

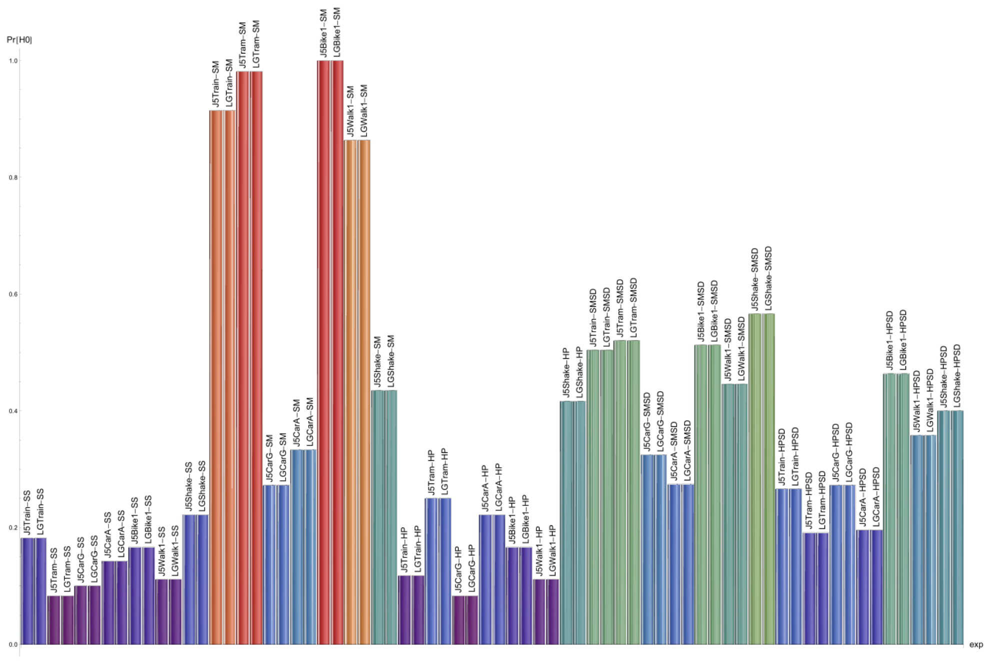


Figure 3.13: Probability of identical vectors (for 8 bit acceleration vectors on LG and Samsung J5)

extracted and will then be enhanced by sigma-delta modulation.

The minimal entropy $\mathsf{H}_{\mathsf{min}}$ for sigma-delta modulation in Table 3.4 is approximately one bit per byte, but as the number of samples increases 40 times, the H extracted each second rises to 10-15 bits/second, which is consistent with the findings of earlier work. Since we choose $\delta = 10ms$ and the sampling rate of the accelerometer is $200ms$, there are have 20 values for each sample, which explains why there are more samples than usual. Additionally, each sigma value has two bits, thanks to the ternary encoding $0, 1, -1$, which results in $2 \times 20 = 40$ values on each sample point. The larger data pool more than offsets the slight loss of entropy for each byte.

Compared to Table 3.3, the findings with sigma-delta modulation show lesser adversary advantages when simple scaling and high-pass are used. High-pass filtering is often performed somewhat worse than simple scaling, but it also avoids the undesirable condition when the adversary advantage is equal to 1. Interestingly, when sigma-delta modulation was not employed, this scaling issue did not arise. Smoothness enhances the likelihood of pairing but also raises the success rate of adversaries, eliminating this mechanism. Oddly, pairing fails at $\ell/8$ in the bike environment with smoothness. Since smoothness performs better than the other filters in terms of success rates for pairing, it may be because the values were not aligned properly in the first place.

Figure 3.13 presents a graphic depiction of the likelihood of similar vectors in all

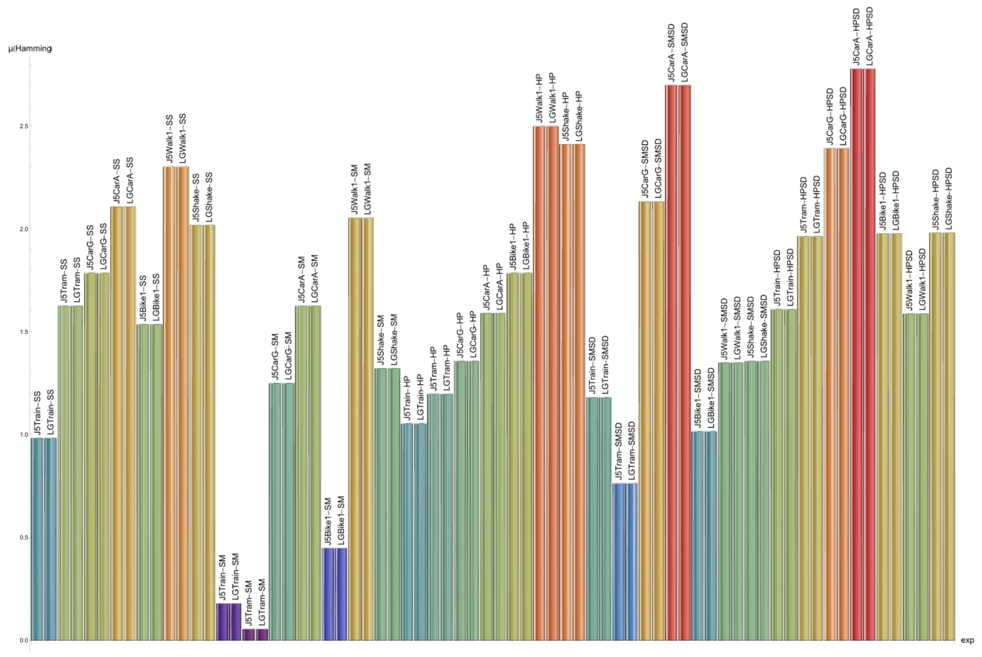


Figure 3.14: Mean Hamming distance (for 8 bit acceleration vectors on LG and Samsung J5)

experiments. Using the smoothness filters increases the likelihood of matching. In this regard, sigma-delta modulation is also beneficial. The habitats of trains and trams are likewise the best for matching feature vectors, but it should be noted that these should also be more vulnerable to attackers who might gather information from the same environment.

First, since it was the default and all phones could easily handle it, a modest 200ms sampling rate was used. A faster sampling rate will enhance the results, but it will also use more battery life and computational resources. Additionally, the precise location of the phones in the car and train scenarios demands more consideration. In what follows, all of these are discussed next.

In order to test different pairing scenarios, the LG is switched for a Samsung A3, as shown in Table 3.5. When the sampling rate was increased, the LG Optimus did not respond well; most samples had large clock drifts from the Samsung J5, which complicated pairing. The results at 20 or 100 ms do not demonstrate significant improvements in most situations. The shaking procedure, where p_{Usr} rises to 0.53 at $20ms$, represents the only notable improvement. This may be expected given that fluctuations happen more quickly in case of shaking than in case of walking or using a train or bus. There is also a loss in the car scenario where p_{Usr}, but this is covered in other experiments for cars and trains in the following sections.

Table 3.5: Results with simple scaling (SS) in distinct environments at 20ms and 100ms (Samsung A3)

Proc.	Env.	$p_{Adv\mathcal{D}}$	$p_{Adv\mathcal{H}}$	p_{Usr}	$\epsilon^{\ell/8}_{Adv\mathcal{D}}$	$\epsilon^{\ell/8}_{Adv\mathcal{H}}$	$\epsilon^{\ell/8}_{Usr}$	$\epsilon^{\ell/4}_{Adv\mathcal{D}}$	$\epsilon^{\ell/4}_{Adv\mathcal{H}}$	$\epsilon^{\ell/4}_{Usr}$	$\mathsf{H}_{\min}$	H	$\mu(\text{Ham})$	$\lvert\bar{v}_0\rvert$	ℓ
20ms	Tram	0.02	0.03	0.27	$< 10^{-31}$	$< 10^{-20}$	1.00	$< 10^{-102}$	$< 10^{-77}$	0.84	2.98	0.05	1.45	150	562
20ms	Bike	0.23	0.23	0.40	1.00	1.00	1.00	0.12	0.12	1.00	0.80	0.02	1.22	223	562
20ms	Walk	0.11	0.11	0.36	0.10	0.13	1.00	$< 10^{-21}$	$< 10^{-20}$	1.00	1.71	0.04	1.09	200	562
20ms	Shake	0.30	0.30	0.53	1.00	1.00	1.00	1.00	1.00	1.00	0.84	0.02	1.04	298	562
100ms	Tram	0.00	0.01	0.13	0.00	$< 10^{-12}$	0.54	0.00	$< 10^{-31}$	$< 10^{-4}$	3.81	0.04	1.87	14	112
100ms	Bike	0.07	0.07	0.45	0.03	0.03	1.00	$< 10^{-8}$	$< 10^{-8}$	1.00	2.64	0.04	0.96	50	112
100ms	Walk	0.05	0.05	0.24	0.00	0.00	1.00	$< 10^{-11}$	$< 10^{-11}$	0.45	2.17	0.04	1.27	27	112
100ms	Shake	0.01	0.04	0.22	$< 10^{-12}$	$< 10^{-3}$	1.00	$< 10^{-31}$	$< 10^{-13}$	0.28	2.32	0.04	1.08	25	112

Table 3.6: Results with simple scaling (SS) on experimental data collected inside the car at 20ms, 100ms and 200ms (Samsung J5)

Proc.	Env.	$p_{Adv\mathcal{D}}$	$p_{Adv\mathcal{H}}$	p_{Usr}	$\epsilon^{\ell/32}_{Adv\mathcal{D}}$	$\epsilon^{\ell/32}_{Adv\mathcal{H}}$	$\epsilon^{\ell/32}_{Usr}$	$\epsilon^{\ell/8}_{Adv\mathcal{D}}$	$\epsilon^{\ell/8}_{Adv\mathcal{H}}$	$\epsilon^{\ell/8}_{Usr}$	$\mathsf{H}_{\min}$	H	$\mu(\text{Ham})$	$\lvert\bar{v}_0\rvert$	ℓ
20ms	CarG	0.00	0.00	0.01	0.00	$< 10^{-10}$	0.00	0.00	$< 10^{-82}$	$< 10^{-42}$	2.00	0.03	3.15	8	562
20ms	CarA	0.01	0.02	0.05	0.00	0.05	0.99	$< 10^{-42}$	$< 10^{-33}$	$< 10^{-11}$	1.40	0.03	2.63	29	562
20ms	CarPA	0.02	0.02	0.09	0.03	0.05	1.00	$< 10^{-36}$	$< 10^{-33}$	0.01	2.27	0.04	2.30	53	562
20ms	CarGA	0.02	0.02	0.07	0.24	0.24	1.00	$< 10^{-27}$	$< 10^{-27}$	$< 10^{-5}$	1.55	0.03	2.72	41	562
100ms	CarG	0.03	0.03	0.07	0.58	0.58	0.99	$< 10^{-6}$	$< 10^{-6}$	0.03	1.42	0.03	2.85	8	112
100ms	CarA	0.03	0.03	0.05	0.58	0.58	0.94	$< 10^{-6}$	$< 10^{-6}$	0.00	1.00	0.02	2.63	6	112
100ms	CarPA	0.00	0.02	0.15	0.00	0.32	1.00	0.00	$< 10^{-8}$	0.82	3.09	0.04	1.69	17	112
100ms	CarGA	0.03	0.03	0.06	0.58	0.58	0.97	$< 10^{-6}$	$< 10^{-6}$	0.01	1.22	0.02	2.51	7	112
200ms	CarG	0.00	0.02	0.16	0.00	0.64	1.00	0.00	$< 10^{-4}$	0.82	3.17	0.04	1.84	9	56
200ms	CarA	0.02	0.02	0.05	0.64	0.64	0.95	$< 10^{-4}$	$< 10^{-4}$	0.03	1.58	0.02	2.88	3	56
200ms	CarPA	0.07	0.07	0.20	0.98	0.98	1.00	0.10	0.10	0.94	1.46	0.03	1.77	11	56
200ms	CarGA	0.07	0.07	0.11	0.98	0.98	1.00	0.10	0.10	0.39	0.58	0.01	2.48	6	56

Table 3.7: Results with sigma-delta modulation over simple-scaling (SSSD) on experimental data collected inside the car at 20ms and 200ms (Samsung A3)

Proc.	Env.	$p_{Adv\mathcal{D}}$	$p_{Adv\mathcal{H}}$	p_{Usr}	$\epsilon^{\ell/16}_{Adv\mathcal{D}}$	$\epsilon^{\ell/16}_{Adv\mathcal{H}}$	$\epsilon^{\ell/16}_{Usr}$	$\epsilon^{\ell/4}_{Adv\mathcal{D}}$	$\epsilon^{\ell/4}_{Adv\mathcal{H}}$	$\epsilon^{\ell/4}_{Usr}$	$\mathsf{H}_{\min}$	H	$\mu(\text{Ham})$	$\lvert\bar{v}_0\rvert$	ℓ
20ms	CarG	0.00	0.04	0.10	$< 10^{-147}$	$< 10^{-7}$	1.00	$< 10^{-945}$	$< 10^{-273}$	$< 10^{-96}$	1.32	0.03	2.10	219	2249
20ms	CarA	0.04	0.04	0.10	$< 10^{-5}$	$< 10^{-5}$	1.00	$< 10^{-259}$	$< 10^{-259}$	$< 10^{-94}$	1.23	0.02	2.17	221	2249
20ms	CarPA	0.05	0.05	0.14	0.00	0.00	1.00	$< 10^{-228}$	$< 10^{-228}$	$< 10^{-43}$	1.53	0.03	2.09	315	2249
20ms	CarGA	0.05	0.05	0.10	0.00	0.00	1.00	$< 10^{-224}$	$< 10^{-224}$	$< 10^{-87}$	1.06	0.02	2.30	231	2249
200ms	CarG	0.12	0.12	0.33	1.00	1.00	1.00	$< 10^{-68}$	$< 10^{-68}$	1.00	1.49	0.03	2.80	731	2245
200ms	CarA	0.10	0.10	0.23	1.00	1.00	1.00	$< 10^{-95}$	$< 10^{-95}$	0.03	1.26	0.03	3.15	523	2245
200ms	CarPA	0.15	0.15	0.36	1.00	1.00	1.00	$< 10^{-36}$	$< 10^{-36}$	1.00	1.27	0.03	2.58	800	2245
200ms	CarGA	0.14	0.14	0.29	1.00	1.00	1.00	$< 10^{-47}$	$< 10^{-47}$	1.00	1.12	0.03	3.01	660	2245

Although the J5 and A3 can handle a greater sampling rate, the experimental data collected inside the vehicle produced fewer matching vectors than in the other cases, i.e., $p_{Usr} \in [0.1, 0.2]$. As a result, several locations for the phones within the vehicle were tested. Two phones in the anti-slip, two in the glove compartment, one in the anti-slip and the other in the glove compartment, and one in the passenger's pocket and the other in the glove compartment were tested.

The results are presented in Table 3.6 at three different sampling rates: $20ms$, $100ms$,

Table 3.8: Results with simple scaling (SS) in the car following scenario golf vs. honda at 20ms, 100ms and 200ms (Samsung J5)

Proc.	Env.	$p_{Adv\mathcal{D}}$	$p_{Adv\mathcal{H}}$	p_{Usr}	$\epsilon^{\ell/16}_{Adv\mathcal{D}}$	$\epsilon^{\ell/16}_{Adv\mathcal{H}}$	$\epsilon^{\ell/16}_{Usr}$	$\epsilon^{\ell/4}_{Adv\mathcal{D}}$	$\epsilon^{\ell/4}_{Adv\mathcal{H}}$	$\epsilon^{\ell/4}_{Usr}$	$\mathsf{H}_{\mathsf{min}}$	H	$\mu(\mathrm{Ham})$	$\lvert\bar{v}_0\rvert$	ℓ
20ms	CarF	0.00	0.00	0.01	$< 10^{-15}$	$< 10^{-15}$	$< 10^{-6}$	$< 10^{-102}$	$< 10^{-102}$	$< 10^{-61}$	2.00	0.02	3.98	4	562
100ms	CarF	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	4.08	0	112
200ms	CarF	0.02	0.02	0.02	0.64	0.64	0.64	$< 10^{-4}$	$< 10^{-4}$	$< 10^{-4}$	0.00	0.00	3.96	1	56

Table 3.9: Results with simple scaling (SS) for various placements of the LG Optimus in train

Proc.	Env.	$p_{Adv\mathcal{D}}$	$p_{Adv\mathcal{H}}$	p_{Usr}	$\epsilon^{\ell/8}_{Adv\mathcal{D}}$	$\epsilon^{\ell/8}_{Adv\mathcal{H}}$	$\epsilon^{\ell/8}_{Usr}$	$\epsilon^{\ell/4}_{Adv\mathcal{D}}$	$\epsilon^{\ell/4}_{Adv\mathcal{H}}$	$\epsilon^{\ell/4}_{Usr}$	$\mathsf{H}_{\mathsf{min}}$	H	$\mu(\mathrm{Ham})$	$\lvert\bar{v}_0\rvert$	ℓ
200ms	Table-Arm	0.11	0.11	0.38	0.39	0.39	1.00	$< 10^{-3}$	$< 10^{-3}$	0.98	1.81	0.03	1.41	21	56
200ms	Table-Seat	0.00	0.04	0.13	0.00	$< 10^{-3}$	0.56	0.00	$< 10^{-9}$	$< 10^{-3}$	1.81	0.03	2.66	7	56

and $200ms$. With simple scaling, the results of p_{Usr} are between 0.05 and 0.20, and it were hard to distinguish clearly across placements, which shows that speed and other road impediments (such as road bumpers and turning places) were more important than phone placement. The number of matching vectors increased once more with the sigma-delta modulation. For simplicity, sigma-delta modulation findings were provided for acquisition delays of 20 ms and 200 ms in Table 3.7. Even if the data in the car appears to have less entropy, it is still sufficient to distinguish hostile behavior from other behaviors. Tables 3.6 and 3.7 show a reduction in the number of successful trials to $\ell/16$ – $\ell/32$ (as opposed to $\ell/4$ – $\ell/8$ as suggested in Table 3.5 for the other environments).

A car-following scenario is also included to further explain how these results relate to information gleaned from several vehicles on the same road. In these tests, a VW Golf was followed by a Honda Civic, and the Samsung J5 was used inside the Honda while the Samsung A3 was used inside the Golf. For both cars, the anti-slip systems were used to place the smartphones. According to Table 3.8 in the car-following scenario, p_{Usr} is between 0 and 0.02, indicating that smartphones from different vehicles do not successfully pair. Road obstructions were not reached simultaneously because one car was a few meters behind. Thus potentially better results could be achieved with correct time alignment. Future work could involve a more thorough study. The sole goal was to demonstrate that information gathered from other vehicles does not necessarily produce the same results. A similar scenario partially addresses the situation of an outside attacker.

Also different phone positions inside the train were tried as an experiment. Two positions were tested: one phone was set down on the table, and the other was on the nearby seat or the armrest. This partially addresses the situation in which one person is carrying both devices, bringing them closer together, or when one device belongs to a different passenger (or an enemy), and the devices are farther apart (i.e., distinct seats). Table 3.9 contains a summary of the findings; for simplicity, the results for simple scaling were included. When the devices were placed closer together, a $p_{Usr} = 0.38$ was reached, and when they were placed farther apart, a $p_{Usr} = 0.13$ was reached. This is consistent with the notion that devices belonging to the same user can be divided to create a private network, although pairing with devices belonging to other travelers is still possible at a lower p_{Usr}.

3.3 A protocol for device pairing and computational results

In this section, a discussion of the suggested protocol and computational results follows.

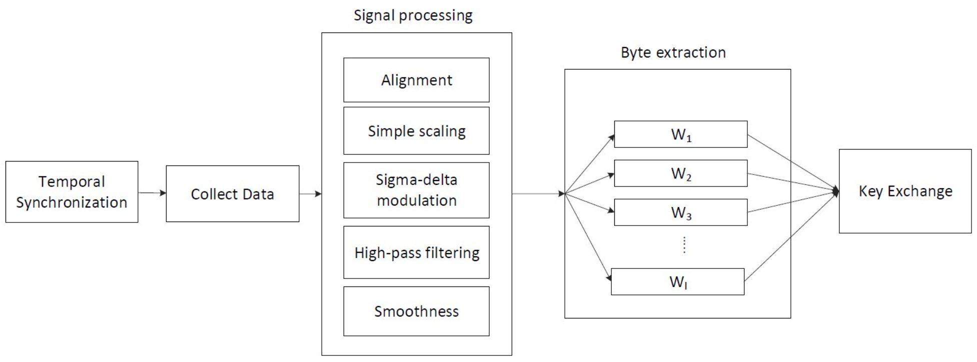


Figure 3.15: Flowchart for signal processing and key-exchange

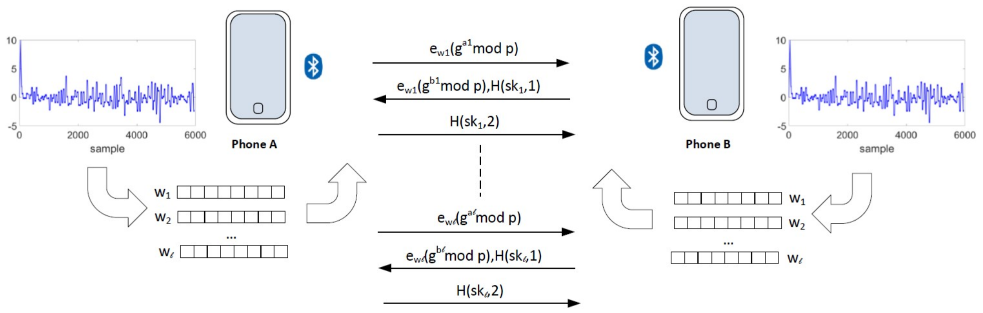


Figure 3.16: Data collection and key-exchange between two smartphones

3.3.1 Exchanging accelerometer data with EKE-DH

Figure 3.15 shows the flowchart for the key exchange step-by-step, from synchronization to data collection, processing and window-splitting for the final key exchange. The

EKE-DH-based technique is described in outline in Figure 3.16. Through Bluetooth communication, multiple windows of accelerometer data are transferred between the phones. A formal description of the protocol is then provided.

EKE-DH based protocol. It is assumed that the generator g of Z_p and a large prime number p are fixed as the public system parameters. Additionally, the security level is fixed at a parameter k. The two phones, A and B, use wireless communication, e.g., Bluetooth, as described below:

1. $\mathsf{Coll}(\Delta)$ used by both phones A and B to store the collected data within predetermined time-windows Δ, applying the filtering methods (time-alignment, scaling, high-pass, and sigma-delta modulation) and then splitting the data into ℓ windows, i.e., $w_1^{\mathsf{id}}, w_2^{\mathsf{id}}, ..., w_\ell^{\mathsf{id}}$ where $\mathsf{id} \in \{A, B\}$;

2. $\mathsf{EKE-DH}(w_i^{\mathsf{id}}, \mathsf{id} \in \{A, B\}, i = 1..\ell)$ where phones A and B exchange data by encrypting the Diffie-Hellman key-shares with the information from each window, i.e.,

$$\begin{aligned} &for \quad i = 1..l \\ &\qquad A \to B : e_{w_1}(g^{a_1}) mod p \\ &\qquad B \to A : e_{w_1}(g^{b_1}) mod p, H(sk_1, 1) \\ &\qquad A \to B : H(sk_1, 2) \\ &\qquad ... \\ &\qquad A \to B : e_{w_\ell}(g^{a_\ell}) mod p \\ &\qquad B \to A : e_{w_\ell}(g^{b_\ell}) mod p, H(sk_\ell, 1) \\ &\qquad A \to B : H(sk_\ell, 2) \end{aligned}$$

   where $sk_i, i = 1..l$ is the Diffie-Hellman key which is secretly shared, i.e., $sk_i = g^{a_i b_i} mod p, i = 1..l$ and $a_i, b_i, i = 1..l$ are chosen randomly and kept secret on each side;

3. $\mathsf{Extract}(\{(sk_1, H(sk_1, 1), H(sk_1, 2)), ..., (sk_\ell, H(sk_\ell, 1), H(sk_\ell, 2))\})$ where each smartphone A, B, verifies the shares $sk_i, i = 1..\ell$ by comparing $H(sk_i, 1)$ and $H(sk_i, 2), i = 1..\ell$ and only keeps the key shares for which it has equivalent hashes. In the case that there are at least k valid key shares, the connection is closed, and the shared session key is retrieved using a key derivation function applied across the valid shares.

It is considered that extraction happens for ℓ samples in the preceding description. When the surrounding transportation environment changes, this can be done again (this can be detected both by changes in accelerometer patterns and by changes in speed). Future sessions can be made more secure by using a ratcheting algorithm that considers prior key sharing between devices. This is easily accomplished by incorporating earlier keys into the key elicitation process for the present session key (cf. [209]).

Table 3.10: Computational time for the pairing operation: extracting one window of shared data, based on EKE-DH or SPEKE

	EKE-DH Z_p-1024		EKE-DH Z_p-2048		SPEKE-192		SPEKE-256	
Phone	Share	Recover	Share	Recover	Share	Recover	Share	Recover
LG Optimus	$42ms$	$64ms$	$230ms$	$411ms$	$94ms$	$39ms$	$145ms$	$70ms$
Samsung J5	$25ms$	$40ms$	$138ms$	$248ms$	$21ms$	$7ms$	$37ms$	$11ms$
Samsung A3	$22ms$	$39ms$	$126ms$	$236ms$	$20ms$	$7ms$	$34ms$	$12ms$

3.3.2 Computational results

On the Android smartphones, a proof-of-concept pairing application was developed. The computational results for the EKE-DH key exchange as it was performed in Java using the Spongy Castle package are stored in Table 3.10. The decryption of the key shares, $e_w(aP)$ and $e_w(bP)$, in the case of the elliptical curve variant of the EKE-DH protocol, may yield points that do not fit the curve. Entropy leakage results from the correct key, possibly leading to only points on the curve (this is a known issue when elliptical curves on EKE-DH are used). To correct it, the equivalent SPEKE technique from Jablon [185] was used, where the point P is produced by utilizing the shared secret as seed (in this case, w) rather than being encrypted with the secret. Recent research in [210] has shown specific weaknesses in the protocol (impersonation under simultaneous sessions and key malleability), but the same study also makes it clear that the protocol is simply fixable.

The time required to compute the key share for each participant, e.g., g^a or aP, and the time to extract the session key, e.g., g^{ab} or abP, are split in Table 3.10. In the instance of Z_p, the computing time is moderate and ranges from 20 to about 200 ms for both the 1024-bit and 2048-bit modules. With elliptical curves, the data representation is more condensed and the computational time at key recovery is marginally slower, falling between 7 and 69ms. When creating the shares, the benefits of elliptical curves do not seem particularly important (in fact, on one of the smartphones, this is even slower). This is because, for each share, a new base point P on the curve must be generated using the data from the accelerometer, w. To do this, XOR the X-coordinate of a regular point on the curve with the accelerometer data w was used to seed a pseudo-random number generator (PRNG). The point is then determined by computing the Y-coordinate based on the X-coordinate. Nevertheless, because $x^3 + ax + b$ must be a square, not all X coordinates will correspond to locations on the curve. Therefore, if that extraction is unsuccessful, a new point is extracted using the PRNG's subsequent random bytes, and so on. Since the extraction of a new P, it takes longer to build the key share or aP. The benefits of employing elliptical curves instead of the standard Z_p are marginal in the Spongy Castle-based implementation that was used, and they only matter when extracting the session key or determining the size of the key shares not during initialization.

The application acquires accelerometer data depending on the triggered event of the sensor, *onSensorChanged*, which occurs at 200 milliseconds. As previously mentioned, the sample takes into consideration the accelerations along the X, Y, and Z axes in the theoretical analysis. In order to prevent battery depletion and meet the computational requirements of the Diffie-Hellman protocol, the default sample rate was not increased. For the 56 8-bit samples obtained in 90 seconds at a $200ms$ sampling rate, it would take approximately 5.6 seconds, or $56 \times 100ms$, to exchange the key using EKE-DH on the lowest performing phone (LG Optimus). The key exchange can also be done instantly when the samples are being gathered. Sigma-delta modulation considerably increases the number of samples, and there may be too many samples to exchange using the Diffie-Hellman protocol. This issue can be resolved while maintaining a similar success rate by choosing fewer samples.

In Tables 3.3 and 3.4 are presented the results that are verified by the Android application. The same matching vectors from Matlab and Mathematica were obtained as in the theoretical study.

3.4 Accelerometers characteristics as smartphone fingerprints

As the author already mentioned in the previous chapter, many research works focussing on fingerprinting mobile phones using their sensors have been put along in the literature, with a focus on microphones [22, 137], loudspeakers [87, 60, 21], cameras [23], magnetometers [43], gyroscopes [116], etc. Any sensor type could have unstable features due to environmental factors, making identification more difficult. The topic of smartphone identification based on accelerometer flaws is covered in this section.

3.4.1 Data collection for fingerprinting analysis

In a vibrating environment, data is simultaneously acquired from all smartphones. The author created an Android application for data collecting that uses the smartphone's accelerometer to capture data at a sample rate of 10ms. The collected data was then placed in a text file for signal analysis. Each file comprises data gathered over 40 minutes or roughly 240.000 samples. In order to perform the measurements independent of the orientation of the phone, first in computed the square for each axis in the three-axis accelerometer data, added the results, and then extracted the square root as follows:

$$a = \sqrt{a_X^2 + a_Y^2 + a_Z^2}.$$

The data gathered from each phone is divided into 200 signals containing 1.000 samples, or 10 seconds, each, in order to do the classification. Then, with a 10% increase, the signals are randomly divided into training and test data, starting with 20% training and 80% testing and increasing to 80% training and 20% testing. The signals are then

Table 3.11: A brief overview of the smartphones used

No.	Type	Smartphone	Label
1.	identical	Samsung Galaxy J5	A
2.		Samsung Galaxy J5	B
3.		Samsung Galaxy J5	C
4.		Samsung Galaxy J5	D
5.		Samsung Galaxy J5	E
1.	different	LG Optimus P700	F
2.		Samsung Galaxy S7	G
3.		Samsung Galaxy A21s	H
4.		Samsung Galaxy J5	I
5.		Allview V1 Viper I	J

processed and analyzed using several well-known machine learning methods, including the Wide Neural Network, Ensemble, KNN, SVM, and Decision Tree.

Regarding the devices, 5 identical Samsung Galaxy J5 smartphones and 5 distinct smartphones, i.e., LG Optimus P700, Samsung Galaxy S7, Samsung Galaxy A21s, Samsung Galaxy J5 and Allview V1 Viper I are used. In Table 5.1, an overview of the smartphones that are used and the corresponding labels is provided.

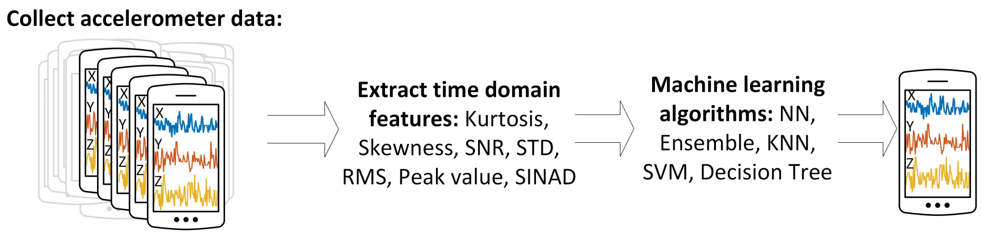
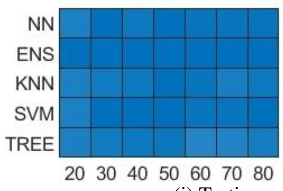

Collect accelerometer data:
X
Y
Z
Extract time domain features: Kurtosis, Skewness, SNR, STD, RMS, Peak value, SINAD
Machine learning algorithms: NN, Ensemble, KNN, SVM, Decision Tree

Figure 3.17: Concept overview

3.4.2 Signal processing and machine learning algorithms

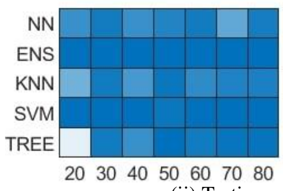
In Figure 3.17 the concept overview is shown starting with data collection, followed by feature extraction and then smartphone classification. From each signal, the following time-domain features were extracted: Kurtosis, Skewness, SNR (Signal-to-Noise Ratio), STD (Standard deviation), RMS (Root-Mean-Square), peak value and SINAD (Signal to Noise and Distortion Ratio).

The extracted time domain features are finally sent as input to several classification algorithms to learn the characteristics of the smartphone based on their accelerometers. The following five machine learning algorithms that are available in Matlab are used: NN (Wide Neural Network), Ensemble, KNN, SVM, and Decision Tree.



	20	30	40	50	60	70	80
NN	0.975	1.0	0.987	1.0	0.991	0.992	0.993
ENS	1.0	1.0	1.0	1.0	1.0	1.0	1.0
KNN	0.975	0.983	0.987	1.0	0.991	0.978	0.98
SVM	0.975	1.0	1.0	1.0	0.991	0.992	1.0
TREE	0.975	0.983	0.987	0.99	0.966	0.971	0.981

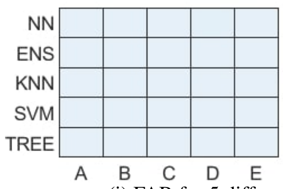
(i) Testing accuracy for 5 different accelerometers



	20	30	40	50	60	70	80
NN	0.950	0.983	0.950	0.970	0.983	0.914	0.981
ENS	1.0	1.0	1.0	1.0	1.0	1.0	1.0
KNN	0.900	0.983	0.937	0.990	0.958	0.971	0.975
SVM	1.0	1.0	1.0	0.990	1.0	1.0	1.0
TREE	0.800	0.983	0.950	1.0	0.983	0.992	0.993

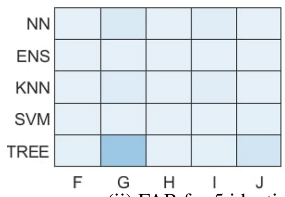
(ii) Testing accuracy for 5 identical accelerometers

Figure 3.18: Testing accuracy as heatmap and numerical values



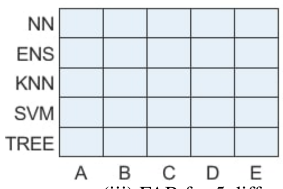
	A	B	C	D	E
NN	0	0	0	0	0
ENS	0	0	0	0	0
KNN	0	0	0	0	0
SVM	0	0	0	0	0
TREE	0	0	0	0	0

(i) FAR for 5 different accelerometers in the case of 20% training



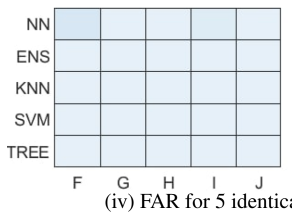
	F	G	H	I	J
NN	0	0.046875	0	0.023	0
ENS	0	0	0	0	0
KNN	0	0.039	0	0.023	0
SVM	0	0	0	0	0.007
TREE	0.007	0.320	0	0.007	0.085

(ii) FAR for 5 identical accelerometers in the case of 20% training



	A	B	C	D	E
NN	0	0	0	0.007	0
ENS	0	0	0	0	0
KMN	0	0	0	0	0
SVM	0	0	0	0	0
TREE	0	0	0	0	0

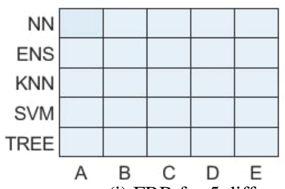
(iii) FAR for 5 different accelerometers in the case of 80% training



	F	G	H	I	J
NN	0.062	0	0	0.031	0
ENS	0	0	0	0	0
KNN	0	0	0	0	0
SVM	0	0	0	0	0
TREE	0	0	0	0	0

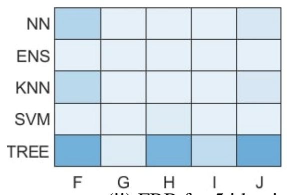
(iv) FAR for 5 identical accelerometers in the case of 80% training

Figure 3.19: FARs as heatmap and numerical values



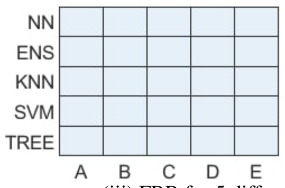
	A	B	C	D	E
NN	0.031	0	0	0	0
ENS	0	0	0	0	0
KNN	0	0	0	0	0
SVM	0	0	0	0	0
TREE	0	0	0	0	0

(i) FRR for 5 different accelerometers in the case of 20% training



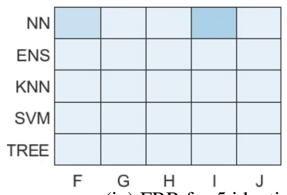
	F	G	H	I	J
NN	0.218	0	0	0	0.062
ENS	0	0	0	0	0
KNN	0.187	0	0	0	0.062
SVM	0	0	0.031	0	0
TREE	0.500	0.031	0.468	0.156	0.531

(ii) FRR for 5 identical accelerometers in the case of 20% training



	A	B	C	D	E
NN	0	0	0	0	0
ENS	0	0	0	0	0
KNN	0	0	0	0	0
SVM	0	0	0	0	0
TREE	0	0	0	0	0

(iii) FRR for 5 different accelerometers in the case of 80% training



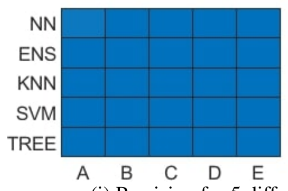
	F	G	H	I	J
NN	0.125	0	0	0.250	0
ENS	0	0	0	0	0
KNN	0	0	0	0	0
SVM	0	0	0	0	0
TREE	0	0	0	0	0

(iv) FRR for 5 identical accelerometers in the case of 80% training

Figure 3.20: FRRs as heatmap and numerical values

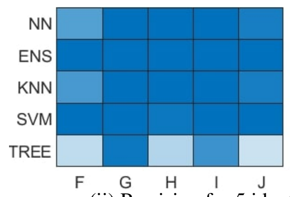
3.4.3 Results

The average testing accuracy by each classifier and the amount of training data utilized are displayed in Figure 3.18. The accuracy for different smartphones is displayed in Figure 3.18 (i), and for identical devices in Figure 3.18 (ii). Overall, each training percentage has satisfactory results. As expected, some classifiers perform better for different phones than in the case of identical ones in terms of accuracy. However, the Ensemble classifier was able to reach 100% accuracy for all training percentages in both scenarios, i.e., with identical and different smartphones. The SVM classifier also yields good accuracy in both cases ranging from 99% to 100%. The only exception is the situation of different smartphones, where 20% of the data was used for training. Although accuracy in this instance is only 97.5%, this is to be expected given the small training set. With regard to the accuracy, KNN achieves a range of 97.5% to 100% for different smartphones and 90% to 99% for identical smartphones. Similarly, NN produces an accuracy of between



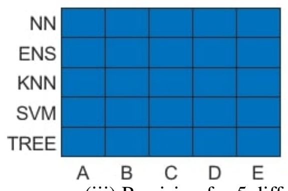
	A	B	C	D	E
NN	0.968	1.0	1.0	1.0	1.0
ENS	1.0	1.0	1.0	1.0	1.0
KNN	1.0	1.0	1.0	1.0	1.0
SVM	1.0	1.0	1.0	1.0	1.0
TREE	1.0	1.0	1.0	1.0	1.0

(i) Precision for 5 different accelerometers in the case of 20% training



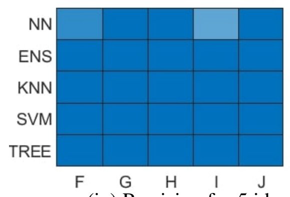
	F	G	H	I	J
NN	0.781	1.0	1.0	1.0	0.937
ENS	1.0	1.0	1.0	1.0	1.0
KNN	0.812	1.0	1.0	1.0	0.937
SVM	1.0	1.0	0.968	1.0	1.0
TREE	0.500	0.968	0.531	0.843	0.468

(ii) Precision for 5 identical accelerometers in the case of 20% training



	A	B	C	D	E
NN	1.0	1.0	1.0	1.0	1.0
ENS	1.0	1.0	1.0	1.0	1.0
KNN	1.0	1.0	1.0	1.0	1.0
SVM	1.0	1.0	1.0	1.0	1.0
TREE	1.0	1.0	1.0	1.0	1.0

(iii) Precision for 5 different accelerometers in the case of 80% training



	F	G	H	I	J
NN	0.875	1.0	1.0	0.75	1.0
EMS	1.0	1.0	1.0	1.0	1.0
KNN	1.0	1.0	1.0	1.0	1.0
SVM	1.0	1.0	1.0	1.0	1.0
TREE	1.0	1.0	1.0	1.0	1.0

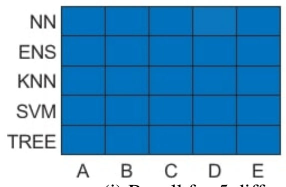
(iv) Precision for 5 identical accelerometers in the case of 80% training

Figure 3.21: Precision as heatmap and numerical values

91% and 98.3% for identical smartphones and between 97% and 100% for different devices. The accuracy of TREE varies between 97.5% and 99% for various devices and between 80% and 100% for identical smartphones. The testing accuracy is higher than 97.5%, barring a few outliers.

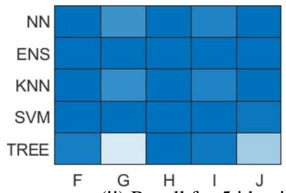
The FAR, FRR, accuracy, precision, and recall are shown only at 20% and 80% training (there is no significant difference for other training percentages in this range). Figures 3.19, 3.20, 3.21, 3.22 show the false acceptance rate (FAR), false rejection rate (FRR), precision, and recall for every classifier and every smartphone. The figures display the metrics for different smartphones (i) and distinct smartphones (ii) trained at 20%. The results for different smartphones (iii) and identical smartphones (iv) at 80% training are then displayed.

In case of 20% training for distinct phones, the FAR is zero for all classifiers and all mobile devices. Except for phone A, which reached an FRR of 3.1% when NN is utilized, the FRR also is zero. Except when NN is applied, the results are 100% in terms



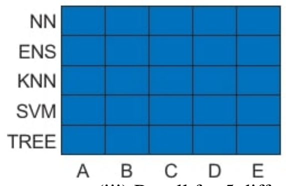
	A	B	C	D	E
NN	1.0	1.0	1.0	0.969	1.0
ENS	1.0	1.0	1.0	1.0	1.0
KNN	1.0	1.0	1.0	1.0	1.0
SVM	1.0	1.0	1.0	1.0	1.0
TREE	1.0	1.0	1.0	1.0	1.0

(i) Recall for 5 different accelerometers in the case of 20% training



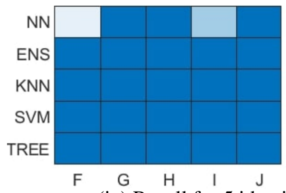
	F	G	H	I	J
NN	1.0	0.842	1.0	0.914	1.0
ENS	1.0	1.0	1.0	1.0	1.0
KNN	1.0	0.864	1.0	0.914	1.0
SVM	1.0	1.0	1.0	1.0	0.969
TREE	0.941	0.430	1.0	0.964	0.576

(ii) Recall for 5 identical accelerometers in the case of 20% training



	A	B	C	D	E
NN	1.0	1.0	1.0	1.0	1.0
ENS	1.0	1.0	1.0	1.0	1.0
KNN	1.0	1.0	1.0	1.0	1.0
SVM	1.0	1.0	1.0	1.0	1.0
TREE	1.0	1.0	1.0	1.0	1.0

(iii) Recall for 5 different accelerometers in the case of 80% training



	F	G	H	I	J
NN	0.777	1.0	1.0	0.857	1.0
ENS	1.0	1.0	1.0	1.0	1.0
KNN	1.0	1.0	1.0	1.0	1.0
SVM	1.0	1.0	1.0	1.0	1.0
TREE	1.0	1.0	1.0	1.0	1.0

(iv) Recall for 5 identical accelerometers in the case of 80% training

Figure 3.22: Recall as heatmap and numerical values

of precision and recall. Phone A reached a precision of 96.8% in this instance, while Phone D reached a recall of 96.9%.

The results for FAR, FRR, accuracy, and recall for identical smartphones are slightly poorer after 20% training. The only exception is Ensemble, which provides 100% precision and recall with zero FAR and FRR. With the exception of smartphone G, the FAR for the remaining classifiers is between 0% and 8.5% when TREE is used. The FRR in this instance is 32%. The precision is between 53.1% and 100%, the recall is between 43% and 100%, and the FAR is between 0% and 53.1%.

As anticipated, 80% training produces more significant results except when NN was used, when the FAR is below 6.2% and the FRR is below 25% for one phone in the case of a different device and two smartphones in the case of identical smartphones. For the remaining classifiers, which include different and identical phones, the FAR and FRR are both 0 in both scenarios. Precision and recall for all classifiers using various phones are 100% when 80% was used. Devices F and I exhibit the worst precision and recall in the scenario of identical smartphones, just above 75% with NN. For the remaining classifiers, accuracy, and recall are 100% for all in the scenario of 80% training.

Overall, across all training percentages, the Ensemble achieves the best results, but the other classifiers still deliver excellent outcomes.

3.5 Concluding remarks

Accelerometer patterns differ considerably depending on the type of motion. The results in this chapter demonstrated that accelerometer data contain enough entropy in any transportation mode to enable the generation of a secure session key. The length of the extracted sequence is a unique parameter that needs to be tuned for each case. Guessing-resilient protocols make it possible to exchange low-entropy values securely by preventing them from being subjected to the brute-force of a hostile adversary. The results show that using different filtering strategies are comparable, although even little variations could favor one technique over another. Simple scaling of the accelerometer data seems to be the best solution due to its simplicity. Sigma-delta modulation, however, is advantageous for increasing feature vectors and might even increase entropy due to improved resolution, but one should take into account that it also results in more features being exchanged and, consequently, more calculations. A proper trade-off between the security level and matching probability requires setting particular parameters according to the environment because transportation environments differ.

Using a variety of machine learning classifiers, including NN, Ensemble, SVM, KNN,and Decision Trees with 7 time domain features, including Kurtosis, Skewness, peak value, STD, SNR, RMS, and SINAD, smartphone fingerprinting based on accelerometer sensors was also presented in this chapter. Several fundamental machine learning techniques, including Decision Trees, NN, Ensemble, KNN, and SVM were employed. The dataset included samples from five different and identical smartphones, and the Ensemble classifier provided a maximum identification accuracy of 100%.

Chapter 4

Fingerprinting smartphones based on loudspeakers characteristics

This chapter is based on the results of the author from [21], a work which addresses smartphones fingerprinting based on loudspeaker characteristics. Experiments with 16 identical and 12 different smartphones were done, trying to identify them based on loudspeaker characteristics extracted from the emitted sound.

4.1 Loudspeaker-based fingerprinting, concept summary

As was already discussed in the literature review section, many related works have been published which use features specific to audio signals, such as mel-frequency cepstral coefficients, spectral kurtosis, spectral centroid, etc. This chapter investigates the application of device loudspeaker roll-off characteristics produced from a linear sweep signal. Calculating the roll-off slope in the scope of the fingerprint takes a straightforward linear approximation. This basic characteristic is enough to distinguish between various devices. However, for more accuracy, specifically in distinguishing speakers from the same smartphone model, deep learning algorithms have a higher success rate for identification.

The significant component which is used to identify the phones in the target scenario is an in-vehicle head unit. A smartphone is fingerprinted based on the recordings made by this device. In this approach, users can authenticate without physical keys based on the device features and in-vehicle head unit systems may use the device fingerprint to unlock specific functionalities. The setup is illustrated graphically in Figure 5.6. Using a similar head unit, 3.000 recordings using the 28 devices are made. Although other scenarios can be envisioned, the car environment is chosen for the experiments that follow. The particular interest in this scenario comes from recent industry and research initiatives to use smartphones as smart car keys, such as those in [211], [212], [213] or [29], a task in which smartphone identification by car head-units may find a practical

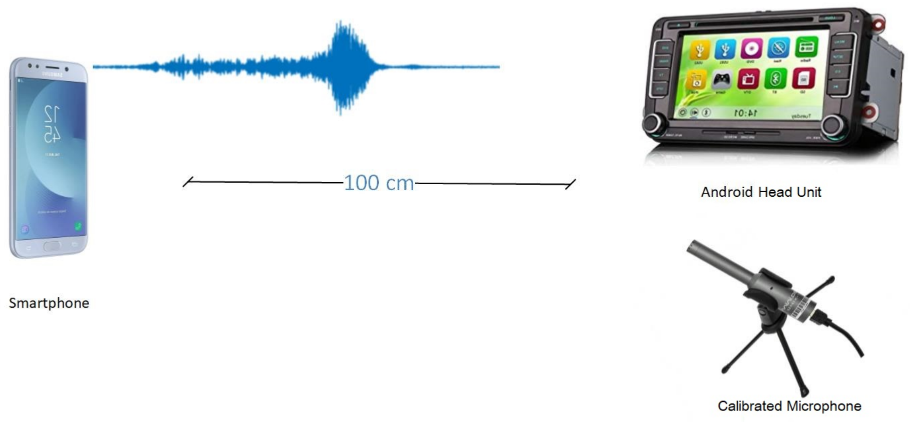


Figure 4.1: Suggestive depiction of the setup: an Android infotainment unit or microphone records sound emitted by a smartphone

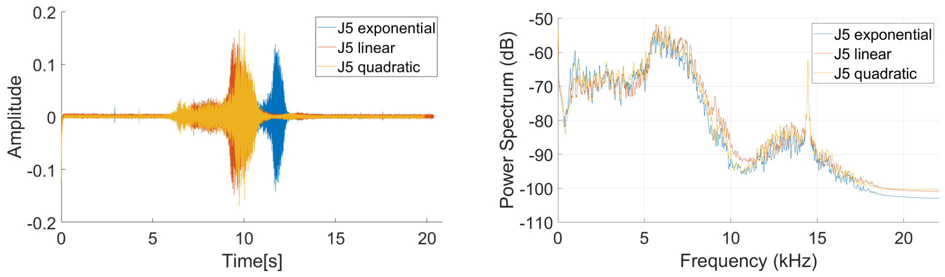


Figure 4.2: Signals in the time (left) and frequency domain (right) corresponding to the three types of chirps emitted by a Samsung J5 (linear, quadratic and exponential)

benefit. In specific automotive scenarios, such as the generation of secure keys and component identification, the use of physical features has been proposed as an authentication approach. For automotive contexts, several sources have been suggested for producing physical unclonable functions (PUFs), including SRAM [214], optical channels [215], and look-up tables (LUTs) [216]. Other sources, like sound and vibrations, can be used for smartphones when interacting with head units. Recent findings in [217] also point to the in-vehicle scenario as an example of a non-interactive device pairing scenario.

HiFi amateurs and experts frequently test loudspeaker responses with the *linear sweep* function. The term *sweep signal*, sometimes known as *chirp*, refers to a signal whose

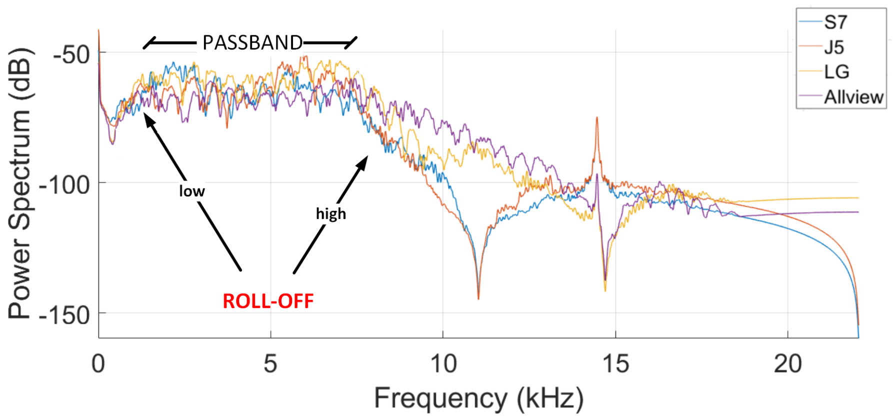


Figure 4.3: Frequency sectors after filtering with a smoothness filter

frequency rises over time. The ideal response for loudspeakers is linear, mainly in the 20Hz-20kHz range (or perhaps even further). However, due to intrinsic technological constraints, the response is not linear and spikes or drops occur at the low and high ends. There are three types of chirps that are frequently employed and are easily accessible in the Matlab mathematical environment: i) linear, i.e., $f(t) = f_0 + \frac{(f_1 - f_0)}{t_1} t$, ii) exponential, i.e., $f(t) = f_0 \left(f_1/f_0\right)^{t/t_1}$ and ii) quadratic $f(t) = f_0 + \frac{(f_1 - f_0)}{t_1^2} t^2$. Here, time is denoted by t, the initial frequency is f_0, and the instantaneous frequency is f_1 at time $t1$. In the implementation $f_0 = 20$Hz, $f_1 = 20$kHz and $t_1 = 10$s is used. Figure 4.2 shows the results from the three chirp functions in the frequency and time domains. One of the smartphones used in the tests is a Samsung J5, which played the signals and the head unit recorded it. The frequencies shift noticeably with time. Therefore, variations are more evident in the time domain. Since the same frequency range (20–20kHz) is investigated in all three forms of chirps – linear, quadratic, and exponential – the frequency response, or power spectrum, is similar for the same smartphone, as would be expected.

However, distinct loudspeakers have relatively varied frequency responses because they have difficulties to reproduce the chirp edges: the bass response is constrained and the high frequencies may begin to beam, all of which result in distinct roll-offs. The power spectrum obtained from the linear sweep signal, between 20Hz and 20kHz, captured by four smartphones i.e., the Samsung S7 (blue), Samsung J5 (red), LG Optimus P700 (orange), and Allview V1 Viper I (magenta) is shown in Figure 4.3. The plot was created in Matlab based on the collected data after applying a smoothness filter, and the same Android infotainment unit made the recordings. Specifically, the range is divided into three sectors after recording the linear sweep function between 20Hz and 20kHz: the first sector was between 700Hz and 3kHz, the second was between 3kHz and 7kHz, and the third was between 7kHz and 11kHz. The investigation is focused in the 700Hz

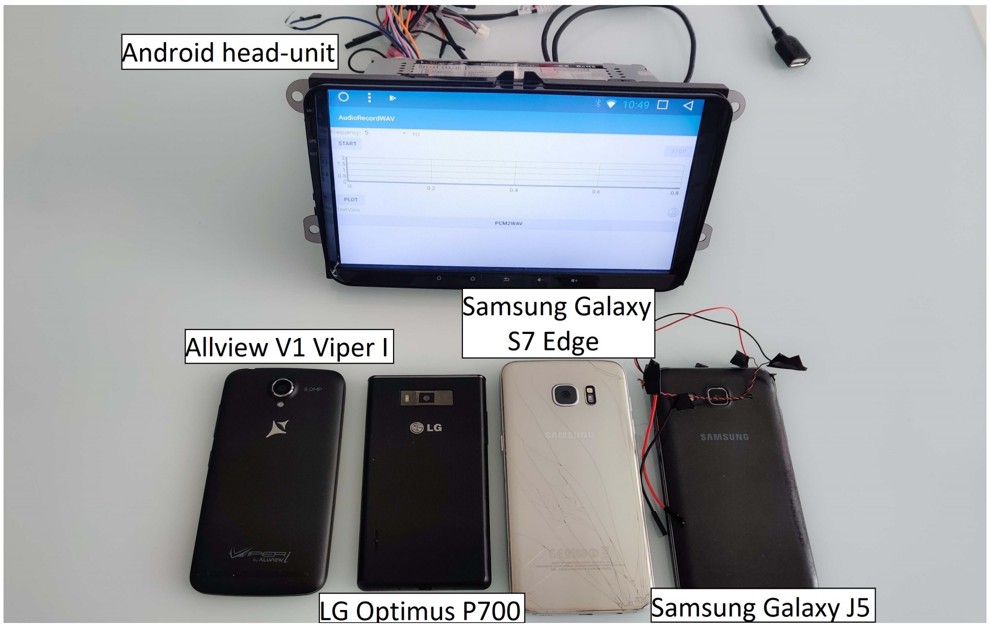


Figure 4.4: The head-unit and four smartphones used in the experiments: Allview V1 Viper I, LG Optimus P700, Samsung Galaxy S7 and Samsung Galaxy J5

- 11kHz range because the power spectrum, computed in Matlab, revealed that loudspeaker response was poor below 700Hz and above 11kHz. A passband region in the center that corresponds to the loudspeaker's midrange frequencies is clearly seen. The left and right stop-bands, which stand for a frequency range that the smartphone loudspeaker has problems reproducing, generate low and high roll-offs. This splits this signal into three sectors, which correspond to the low, middle, and high frequencies, which is natural. For instance, the majority of high-end HiFi systems use a 3-way design and separate loudspeakers for the reproduction of the bass, midrange, and treble. Based on the experimental observation of the rising and falling edges of the signals, the cut-off frequencies of 700Hz, 3kHz, 7kHz, and 11kHz were chosen because they were well suited for the devices in this analysis. These ranges ought to work for most smartphones, according to the variety of the chosen devices. Figure 4.3 illustrates that the smartphone loudspeakers are scarcely able to cover 20Hz – 20kHz range.

4.2 Setup and methodology

This section provides a summary of the experiments and methods. The experimental setup, the software tools and the equipment utilized to get the data are discussed. Also, an early analysis of the experimental results is provided.

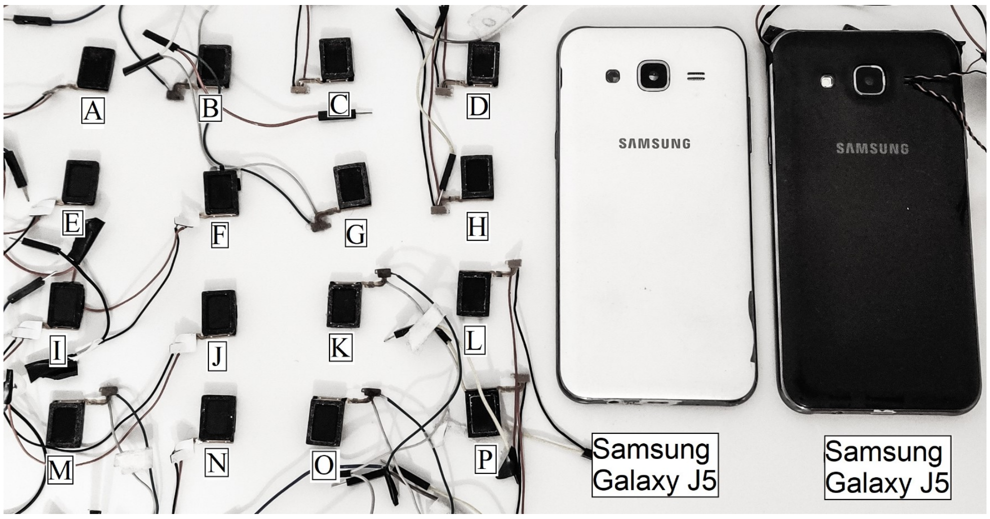


Figure 4.5: Samsung J5, the 16 dismantled loudspeakers and the case

Several different smartphones were used for the initial analysis: an Allview V1 Viper I, a Samsung Galaxy S7 Edge, a Samsung Galaxy J5, a LG Optimus P700, and a Samsung Galaxy J5. Since they were simple to distinguish, the set was expanded with five and later 16 identical Samsung J5 loudspeakers, which made the separation more difficult. In addition to these, a car head unit made by Erisin was used. This device accepts external loudspeakers and has a microphone. Figure 4.4 shows the head-unit and four smartphones from the experiments. The same smartphone used for the fingerprinting scenario is connected to 16 identical J5 loudspeakers from old smartphones that have been disassembled for use in the analysis. Figure 4.5 displays the 16 loudspeakers from J5 smartphones that have been disassembled, together with a J5 case and the J5 used for fingerprinting.

The devices and related characteristics are listed in Table 5.1. 28 different devices have been fingerprinted, 16 of which are identical smartphone loudspeakers housed in the same smartphone cover. A total of 3000 sweep signals have been measured from which 2800 samples target the identical loudspeakers. The 16 identical loudspeakers in the first row make up the bulk of this analysis because they are the most challenging to distinguish, in other words, the worst-case situation. 100 measurements were collected for 13 loudspeakers, which were adequate for using machine learning methods to create fingerprints. In order to examine how separation works in a much bigger dataset, 3 loudspeakers were chosen out of these with highly similar fingerprints and 500 measurements were done. Rows 2-5 depict 4 devices that will be used in subsequent studies, 30 measurements were collected to test them under various volume and angle adjustments. Last but not least, rows 6 – 12 depict eight devices that were collected from casual visitors. Just 10 measurements were done for these devices, but they are enough to ascertain that

Table 4.1: Summary of devices used and the associated measurements

	Phones	Label	No.	Meas.	Total
1.	Samsung Galaxy J5	A, C and F to P	13	100	1300
	(distinct conclspeakers in phone)	B, D, E	3	500	1500
2.	Samsung Galaxy S7 Edge	S7	1	30	30
3.	LG Optimus P700	LG	1	30	30
4.	Allview V1 Viper I	AV	1	30	30
5.	Samsung Galaxy J5 (other)	J5′	1	30	30
6.	Samsung Galaxy J5 (other)	J5″/J5‴	2	10	20
7.	Samsung Galaxy Note 8	N8	1	10	10
8.	Samsung Galaxy A21s	A21	1	10	10
9.	Samsung Galaxy Tab S7	T7	1	10	10
10.	Xiaomi Mi A3	X3	1	10	10
11.	Xiaomi Redmi 7A	X7	1	10	10
12.	Leagoo Z10	LE	1	10	10
	Total		28		3000

they separate sufficiently even with linear estimates of the roll-off frequency. The displacement for all the smartphones used in the studies is shown in Figure 4.6, where the devices are obviously separated. In this picture and several of the figures that follow, the smartphones (or loudspeakers) are positioned in a Cartesian coordinate system in which the ordinate (y-axis) represents the slope of the *low roll-off* and the abscissa (x-axis) represents the slope of the *high roll-off*. Although there is a clear distinction between the devices, two of the J5s, namely J5″ and J5‴, do overlap.

The fingerprint also depends on the recorder because microphone hardware flaws will cause little measurement differences. The recording device is fixed in the case of device-to-device authentication (assuming that only one device emits the sweep signal at once). If numerous devices emit the signal, there will likely be overlaps that prevent identification. Similar effects from background noise will be covered later.

In order to conduct numerous audio measurements, Room EQ Wizard (REW), a free piece of room acoustic software, was utilized. It was used to create a Linear Sweep signal with a frequency ranging from 20Hz to 20kHz. The same signal was afterward created in Matlab, which provides a more advanced math processing environment and a variety of other processing possibilities. A .wav file which has a sampling rate of 48kHz and a resolution of 16 bits is used to store the signal. Each smartphone plays the sample and the infotainment unit records the sweep. More information is provided in the following sections. The Matlab environment was used (`https://www.mathworks.com/products/matlab.html`), which provides a complex tool-set for signal processing to analyze the recorded data offline. For clarification, Figure 4.7 describes the fingerprinting process. In the first step, a sweep signal with a 10-second duration is emitted by the smartphone, which is placed 1 meter from the head unit. The power spectrum is extracted from the signal recorded by the head unit and it is then used to either deter-

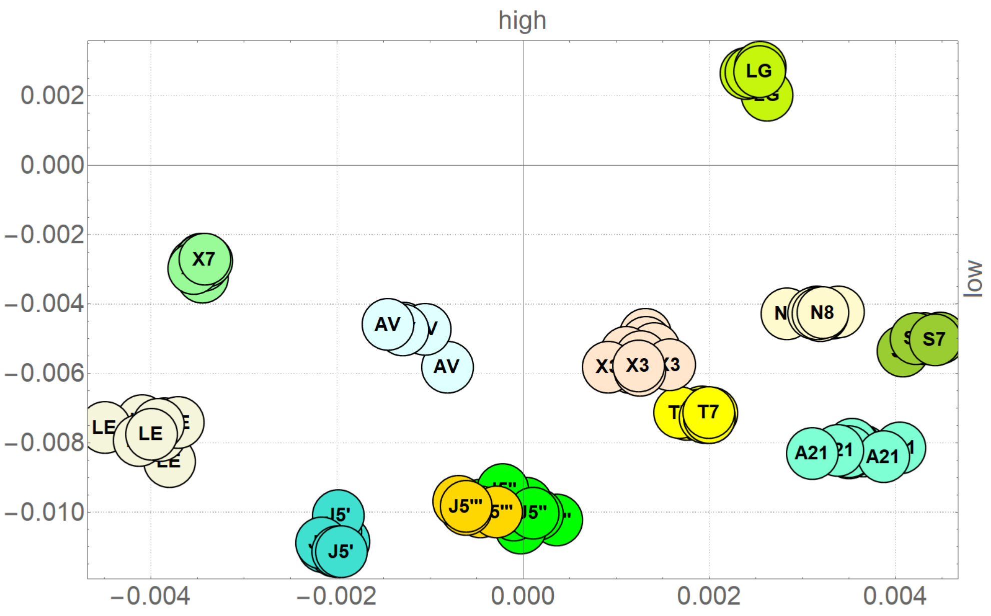


Figure 4.6: Overview of the displacement of 12 smartphones in the experiments according to the roll-off slopes

mine the slopes of the low and high roll-offs or to do more complicated machine learning techniques.

Each smartphone emits the Linear Sweep signal produced by REW from the .wav file. An Android app was used to record the sound from the smartphones on the Erisin head unit. For further investigation, the recorded signal is stored as a .wav file. The experiments are conducted in a $3 \times 3.7 \times 2.5\text{m}$ room (room acoustics may also influence the results). The equipment was set up on a desk and a fixed distance of 1 meter was established between the smartphone playing the sound and the Erisin head unit. Four smartphones were used to test three different volume scenarios: 100%, 75%, and 50% volume. For each J5 loudspeaker, a variety of experiments were conducted. Figure 4.8 provides a graphic summary of the results. First, the loudspeakers in three different locations were tested: the original casing (OR), an acrylic board (AC), and a dampening material (DP). Other records were performed inside a car (ST) or created by adding additive white Gaussian noise (AWGN) to further suppress the original signal. Figure 4.8 shows how this change in the setup affects the recorded roll-offs, but it is also evident that results from the same environment tend to cluster together.

The early stages of analysis attempted to employ a more traditional approach with easier classifiers based on MFCCs, as indicated in other works. However, regular classifiers like KNN cannot adapt to volume variations and thus the volume level may be problematic for such techniques. That is, even though technically they are all set to the same volume level, a classifier may correctly identify smartphones based on the output

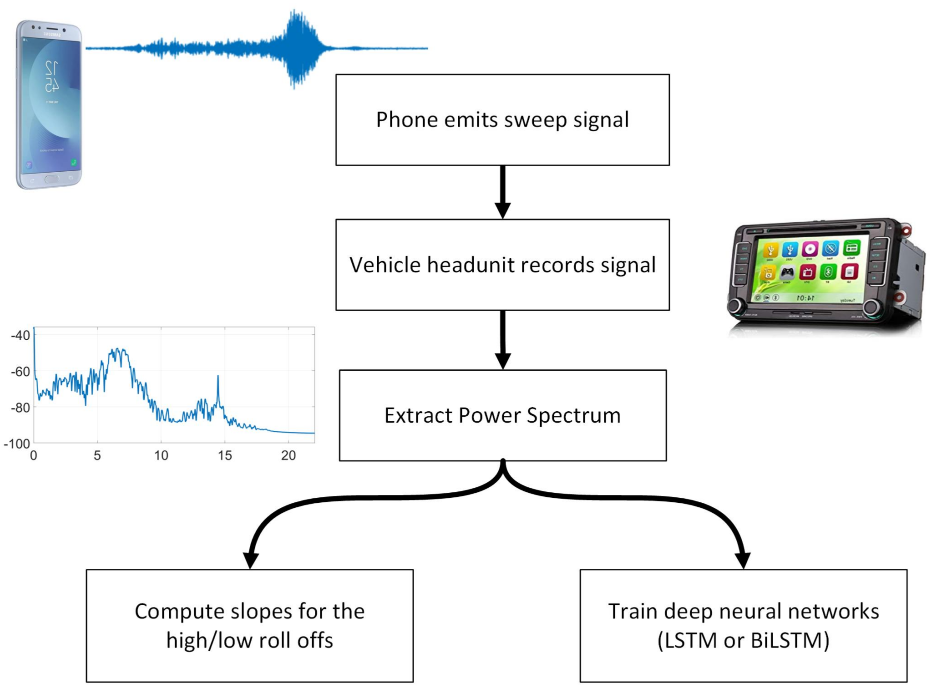

Phone emits sweep signal
Vehicle headunit records signal
Extract Power Spectrum
Compute slopes for the high/low roll offs
Train deep neural networks (LSTM or BiLSTM)
-40
-60
-80
-100
0
5
10
15
20

Figure 4.7: Overview of the fingerprinting procedure

volume, which is distinct on different smartphones, but then it may mismatch devices at a distinct volume (when changed by a user). In this way, the loudness level rather than different patterns in the audio signal determines whether an object is correctly identified. Thus, the classifier may fail to identify the smartphone when adjusting the volume accurately. The experimental results from Figure 4.13, which are examined later, demonstrate that roll-off characteristics are more resistant to changes in volume and direction.

In this analysis the focus is on sweep signals and then recurrent neural networks are used. The potential shortcomings that were identified using traditional machine learning methods applied to periodic data is briefly discussed in the following sections. In particular, when MFCCs is used (the discriminant advised by [86]) directly with the recorded audio signal, the identification works best when the audio output is held at the same level on different smartphones, which in reality produces separate amplitudes for the output. The identification results are no longer accurate after the volume changes to the same actual level.

The results of employing KNN on the features recovered from a linear sweep with MFCCs at different volume levels may be inconsistent, as seen in Table 4.2 (although the roll-offs seems to be less affected, as shown in Figure 4.13). In another scenario, the features recovered from the audio signal from one experiment for the four smartphones were used as training data and the features extracted from the other four smartphones

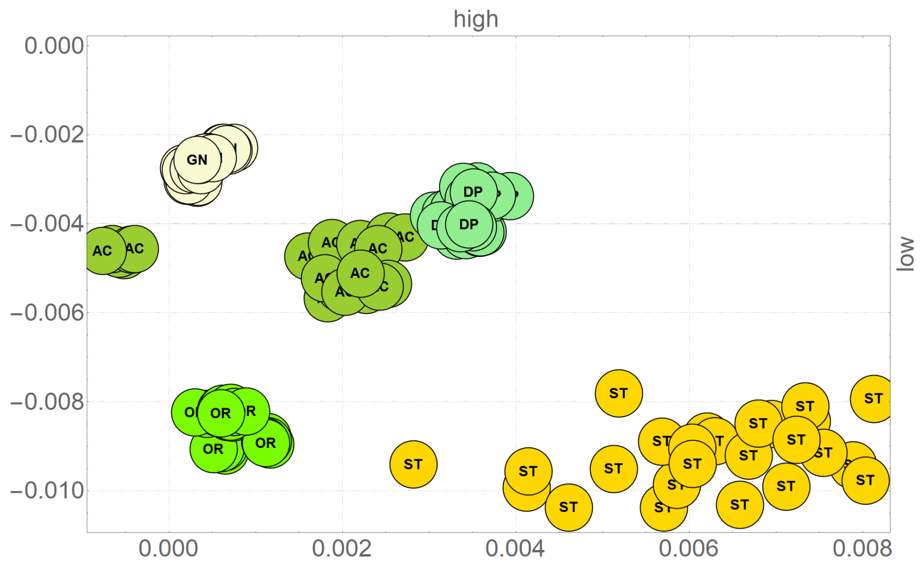


Figure 4.8: Overview of the experimental results for distinct Samsung J5 speakers: speakers placed in original case (OR), on acrylic board (AC), on damping material (DP), suppression with AWGN (GN), and recording in car with street traffic (ST)

were used as test data. In the initial tests, all smartphones are displayed at full volume. In this instance, every smartphone is appropriately recognized. Then, the same number of measurements were keep, but the volume was dropped on all smartphones to 75%. However, 5 experiments with the Samsung S7 with the volume at 100% were used as test data. In this instance, only the LG is accurately identified, and the S7's identification with the J5 and the Allview overlap. The scenario is the same when the volume is reduced to 50%. Testing data for the S7 has been added again at 100%. The volume also varies and depends on the frequency, such variations may be noticeable.

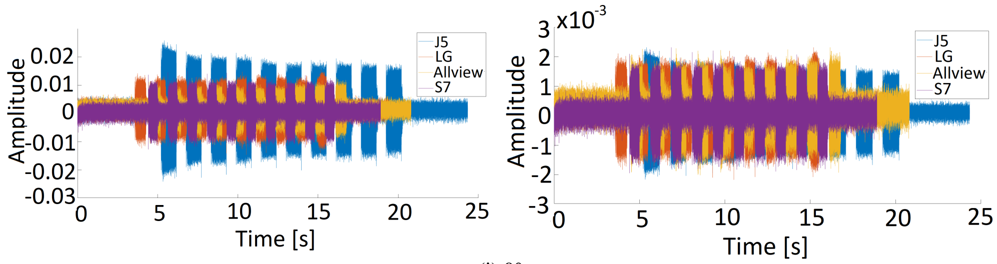
A sine wave with the formula $s(t) = a \sin(2\pi f t / f_s)$, where a is the amplitude, f is the frequency, f_s is the sampling frequency, and t is the time, was also used to test the smartphones. A 1 was represented as a tone, and a 0 was represented as a pause. The loudness levels of the smartphones varied similarly. The amplitude of the noise changes when the volume is scaled to the same value. For instance, the J5 is louder than the others at 1 kHz when all smartphones are held at 100% volume level, whereas the Allview is limited at about 68% of the J5 volume, LG is at 67%, and the S7 is only at 39% of the J5 volume (this is also obvious in Figure 4.10 which is examined later). This difference is sufficient to identify a fingerprint, but the volume level preferred by the user cannot be known. For further clarification, the audio data from four smartphones recorded by the in-vehicle infotainment system is shown in Figure 4.10. The original data are displayed on the left side of the figure and it has been scaled to account for volume level changes that are displayed on the right side. At the same volume level, the classification works,

Table 4.2: Misinterpretations with MFCCs and KNN classification at various volume levels: 100%, 75% and 50%

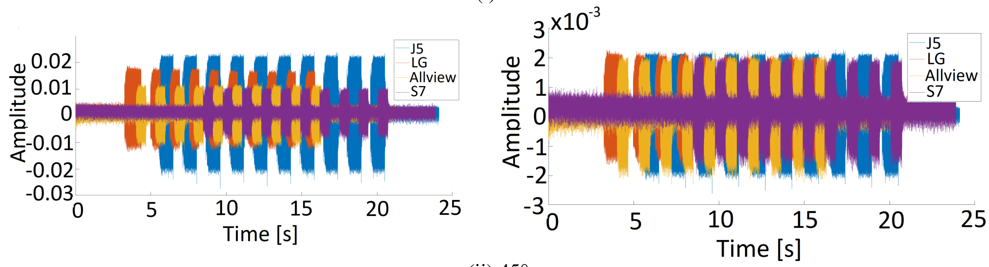
Volume	Phone	J5	S7	LG	AV
100%	J5	56.36%	7.18%	2.46%	34.01%
	S7	1.06%	96.57%	1.19%	1.19%
	LG	12.75%	14.70%	61.43%	11.12%
	AV	28.77%	0.57%	2.83%	67.89%
75%	J5	5.61%	80.42%	1.53%	12.45%
	S7	88.23%	6.95%	2.60%	2.22%
	LG	18.83%	33.12%	45.84%	2.21%
	AV	0.82%	75.71%	0.64%	22.84%
50%	J5	6.70%	78.60%	2.12%	12.58%
	S7	90.61%	1.82%	3.85%	3.73%
	LG	19.61%	28.98%	46.64%	2.78%
	AV	1.02%	65.47%	0.99%	32.51%

but it appears that the classifier is again dependent on the volume level. The noise level appeared to be the main differentiator when the data was adjusted to eliminate variations in volume levels. When the loudspeaker is silent for a while, there is noise. When the data is scaled, the noise also scales up, increasing the discriminant.

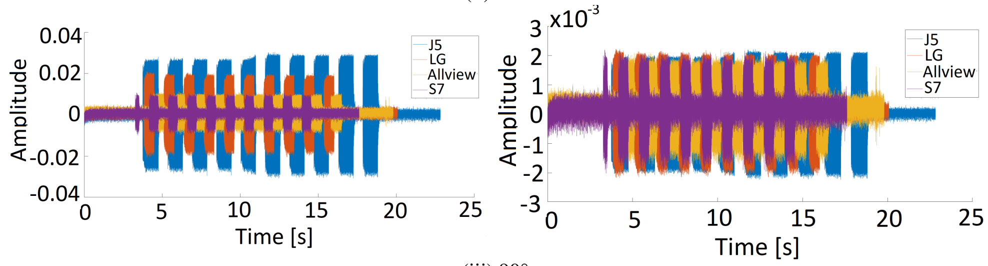
Measurements using a calibrated microphone UMIK-1 from MiniDSP `https://www.minidsp.com/` were also performed in an effort to eliminate measurement issues caused by the lower quality microphone in the in-vehicle Android unit. The signal recorded with the calibrated microphone UMIK-1 from MiniDSP are illustrated in Figure 4.10. The classification results were comparable to the case of the infotainment unit, even though the noise level is obviously considerably lower and the signal level is also higher due to the higher sensitivity of the microphone. The periodic signals are not used in the remaining experiments because they have a much cleaner frequency domain representation. As opposed to Figure 4.10 which displays the far more complex power spectrum of sweep signals, Figure 4.11 illustrates the impact of loudspeaker placement and background noise on a periodic tone. The figure displays the recorded signal from the five speakers for a tone at 1kHz with a periodicity of 500ms. When the loudspeakers are positioned on the acrylic board (left) or the dampening material, the signal is very weakly defined (middle). When the smartphone case is used for the loudspeakers, the signal becomes clearer (right). The power spectrum can be seen at the bottom of the image and features a noticeable spike at 1kHz.



(i) 0°



(ii) 45°



(iii) 90°

Figure 4.9: Recorded signal (left) and scaled (right) on four smartphones for a periodic tone of 1kHz with 500ms periodicity (recordings by infotainment unit)

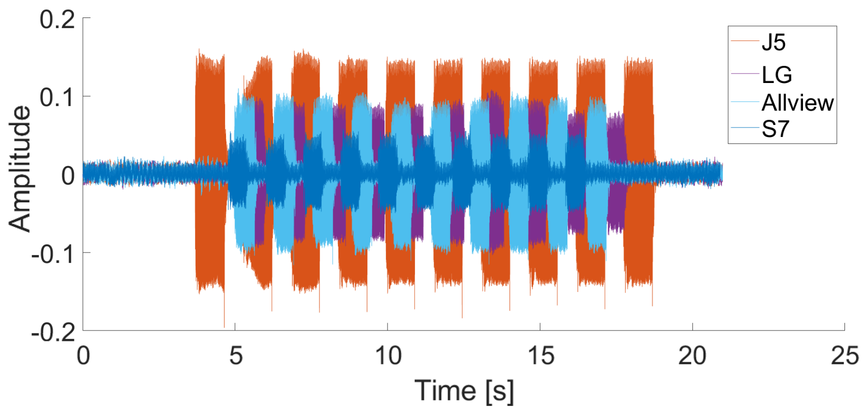
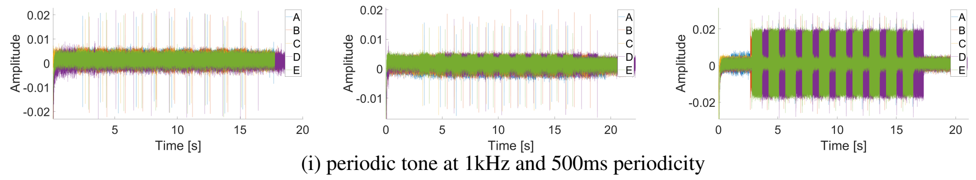


Figure 4.10: Recorded signal on four smartphones for a periodic tone of 1kHz and 500ms periodicity with microphone UMIK-1


(i) periodic tone at 1kHz and 500ms periodicity

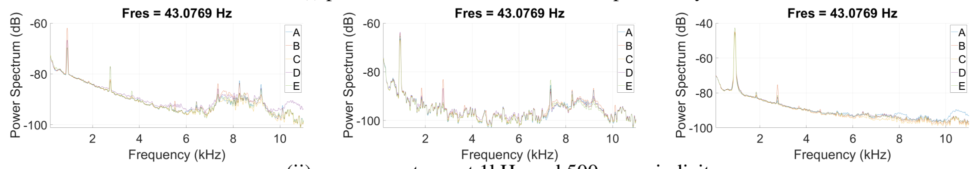

(ii) power spectrum at 1kHz and 500ms periodicity

Figure 4.11: Recorded signal on the five loudspeakers on acrylic board (left), damping material (middle) and inside the smartphone case (right) and power spectrum for a periodic tone at 1kHz and 500ms periodicity (recordings by infotainment unit)

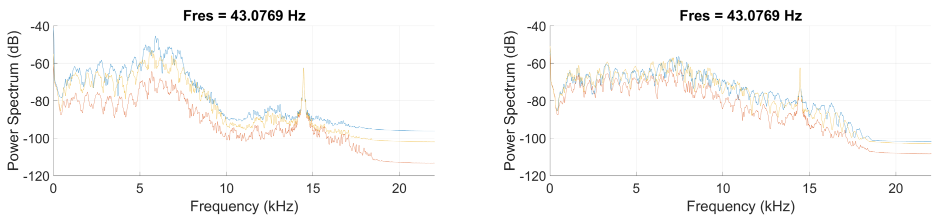


Figure 4.12: Power spectrum of the audio signal at several volume levels 100% (blue), 75% (orange) and 50% volume (red) for Samsung J5 (left) and Allview (right)

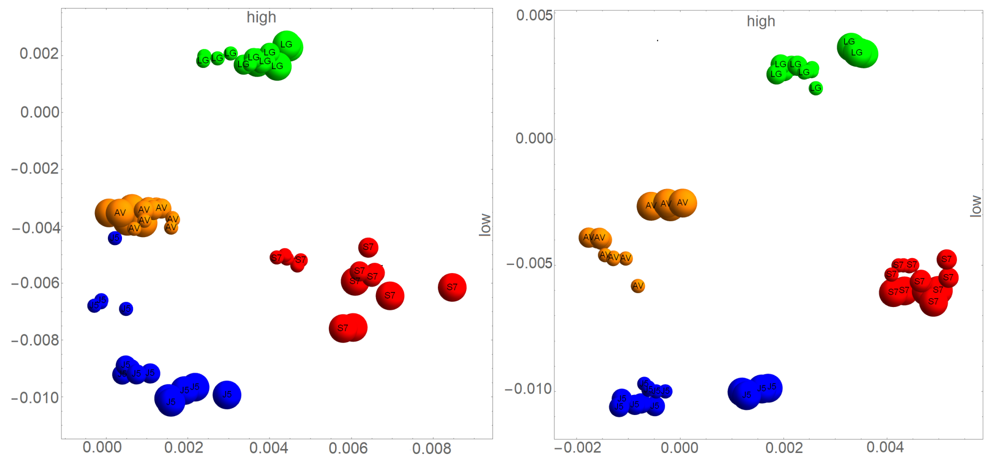
4.3 Fingerprinting speakers based on roll-off slopes

Loudspeakers analysis based on their roll-off characteristics is discussed next. Subsequently, for comparison, an analysis based on periodic signals is performed that exhibit rising and falling edges at a faster rate, for which more demanding machine learning algorithms are used. Finally, the impact of noise on fingerprinting speakers is analyzed in the following subsections.

4.3.1 Roll-off characteristics on distinct smartphones

The Matlab environment is used to conduct a more thorough analysis of the signals captured by the Erisin head-unit. The Signal Analyser App from Matlab's Signal Processing Toolbox is also used to obtain a clear view of the collected data. The base of the analysis is the power spectrum of the signals, i.e., the frequencies of the spectral estimates derived from the power spectrum.

Plots of the power spectrum at three volume levels, i.e., 100% (blue), 75% (orange), and 50% volume (red) are shown in Figure 4.12. When the user (or an adver-



(i) recordings at distinct volume (ii) recording at various angles

Figure 4.13: Clustering based on low and high roll-offs: at disctinct volume levels 50%, 75%, 100% volume (i) and various angles 0°, 45°, 90° (ii) - for the four smartphones S7 (red), J5 (blue), LG (green) and Allview (orange)

sary) changes the volume level of the smartphone, the signal shape remains identical but moves vertically as expected. This can cause misclassification. Unwanted noise may alter the results of the computation of the slope of the signal using linear approximations. Due to this, a smoothness filter is used to reduce the noise. The moving mean filter, which is implemented in the Matlab toolset by the function `smoothdata(sampled data, 'movmean')`, takes as parameters the sampled data and the moving average method, i.e., `movmean` was used to achieve this. Other solutions, such as `movmedian` or `gaussian` were also tried, but they did not produce better results. The noise in the signal did not impact the accuracy of the result.

The frequency from 700Hz to 11kHz is analyzed based on results from Figure 4.12. As shown in Figure 4.3 for each smartphone Samsung S7 (blue), Samsung J5 (red), LG (orange) and Allview (magenta), this frequency range is divided into three sectors that are important for the roll-off characteristics: the first sector is between 700Hz and 3kHz, the second sector is between 3kHz and 7kHz, and the last sector is between 7kHz and 11kHz. A linear approximation is used for each of the three sectors to distinguish between signals. The `polyfit(frequencies, power spectrum, degree)` function in Matlab is used to approximate the linear function. Its parameters are the frequencies of the spectral estimates from the power spectrum, the power spectrum in decibels, and the degree of the approximation polynomial (1 in this case). The formula returns the coefficients of an approximation polynomial of degree 1. The `polyval(polynomial coefficients, points)` function, which takes the coefficients of the polynomial to be queried at evaluation points as parameters, is another option. The function returns the polynomial values for each point.

Table 4.3: Classification: 0°

Phones	TREES	KNN	NB-MVNM	RF	NB-KB	AdaBstM2
J5/J5	**99.33%**	**99.96%**	**99.82%**	**99.33%**	**63.25%**	**63.67%**
J5/S7	0.16%	0.02%	0.16%	0.16%	10.25%	15.50%
J5/LG	0.51%	0.02%	0.03%	0.51%	11.17%	6.28%
J5/AV	0%	0%	0%	0%	14.74%	14.56%
S7/J5	0.02%	0.47%	0.03%	0.02%	44.35%	45.55%
S7/S7	**99.94%**	**99.49%**	**99.97%**	**99.97%**	**14.86%**	**21.25%**
S7/LG	0.02%	0.01%	0%	0%	22.52%	15.07%
S7/AV	0.02%	0.03%	0%	0.01%	18.27%	18.13%
LG/J5	0.2%	0.86%	0.25%	0.18%	36.15%	34.97%
LG/S7	0.28%	0.06%	0.15%	0.18%	7.62%	12.54%
LG/LG	**99.32%**	**98.98%**	**99.6%**	**99.6%**	**29.52%**	**25.91%**
LG/AV	0.2%	0.01%	0%	0.05%	26.71%	26.58%
AV/J5	0.03%	0.27%	0.02%	0.02%	23.16%	23.65%
AV/S7	0.18%	0%	0.17%	0.18%	8.74%	12.20%
AV/LG	0.01%	0.02%	0%	0%	24.21%	15.80%
AV/AV	**99.78%**	**99.7%**	**99.81%**	**99.81%**	**43.89%**	**48.35%**

Figure 4.13 shows how four smartphones cluster at different volume levels (i) and different angles in a Cartesian coordinate system where the x axis represents the slope of the high roll-off and the y - axis is the slope of the low roll-off (ii). The plots consider each of the three different volume levels of the smartphones, which are 50%, 75%, and 100%. The size of the chart element changes depending on the volume level (bigger elements correspond to higher volume). Similarly, the size of the plot element is kept inversely proportional to the angle, i.e., higher elements correspond to 0°, and we investigated three values for the recording angle: 0°, 45°, and 90°. The distinct cluster of four smartphones is easy to discern. There are no overlaps in the data and the angle has a minimal impact. Volume level has some effect on clustering, however, the J5 at 50% volume level, which overlaps with the AV in one of the observations, is the only smartphone that might be misclassified. More measurements are probably required to correctly distinguish between smartphones closer to one another. Because the results were relatively similar with and without the smoothness filter, the graphs for the original signal are used next (without smoothness).

4.3.2 Further analysis of periodic signals

The tests ran using periodic signals and various machine learning techniques are discussed in this section. For classification, the recorded peaks of the audio signals from the four smartphones at 1 kHz with 500 ms of periodicity were used. Each audio signal is split in half, with the first half as training data and the second as test data. The measurements were performed with the loudspeaker of the smartphone positioned at 0°, 45°and 90°from the head unit microphone. In Tables 4.3, 4.4 and 4.5 the results are shown

Table 4.4: Classification: 45°

Phones	TREES	KNN	NB-MVNM	RF	NB-KB	AdaBstM2
J5/J5	**99.80%**	**99.94%**	**99.79%**	**99.81%**	**74.25%**	**76.89%**
J5/S7	0.01%	0%	0.03%	0%	2.61%	2.51%
J5/LG	0.18%	0%	0.18%	0.19%	12.68%	11.85%
J5/AV	0.01%	0.06%	0%	0%	10.46%	8.76%
S7/J5	0.01%	0.22%	0%	0.01%	15.50%	15.34%
S7/S7	**99.8%**	**99.71%**	**99.83%**	**99.81%**	**41.43%**	**43.79%**
S7/LG	0.17%	0.03%	0.17%	0.17%	13.94%	16.19%
S7/AV	0.01%	0.03%	0%	0.01%	29.13%	24.67%
LG/J5	0.03%	0.3%	0 %	0.03%	33.37%	32.68%
LG/S7	0.02%	0.01%	0.03%	0 %	16.58%	16.61%
LG/LG	**99.95%**	**99.66%**	**99.97%**	**99.97%**	**16.32%**	**24.3%**
LG/AV	0%	0.03%	0 %	0 %	33.73%	26.41%
AV/J5	0%	0.14%	0%	0%	41.66%	43.43%
AV/S7	0%	0%	0%	0%	7.49%	7.55%
AV/LG	0.18%	0.05%	0.18%	0.18%	8.05%	9.03%
AV/AV	**99.82%**	**99.81%**	**99.82%**	**99.82%**	**42.8%**	**39.99%**

Table 4.5: Classification: 90°

Phones	TREES	KNN	NB-MVNM	RF	NB-KB	AdaBstM2
J5/J5	**98.35%**	**95.79%**	**98.73%**	**98.87%**	**78.97%**	**79.67%**
J5/S7	0.66%	0.45%	0%	0.15%	1.63%	1.34%
J5/LG	0.05%	0.28%	1.26%	0.05%	3.26%	3.78%
J5/AV	0.94%	3.48%	0.01%	0.93%	16.14%	15.2%
S7/J5	0.25%	0.29%	0.23%	0.24%	11.86%	11.41%
S7/S7	**99.75%**	**99.67%**	**99.76%**	**99.76%**	**22.71%**	**13.48%**
S7/LG	0%	0%	0.01%	0%	47%	55.77%
S7/AV	0%	0.04%	0%	0%	18.43%	19.34%
LG/J5	0.17%	0.23%	0.15 %	0.16%	9.2%	8.75%
LG/S7	0.01%	0.01%	0%	0 %	16.77%	11.06%
LG/LG	**99.82%**	**99.74%**	**99.84%**	**99.83%**	**59.14%**	**64.68%**
LG/AV	0.01%	0.02%	0 %	0.01%	14.89%	15.51%
AV/J5	0.25%	0.31%	0.17%	0.25%	25.66%	25.93%
AV/S7	0%	0%	0%	0%	7.08%	5.09%
AV/LG	0%	0%	0.01%	0%	10.36%	11.81%
AV/AV	**99.75%**	**99.69%**	**99.82%**	**99.75%**	**56.9%**	**57.17%**

Table 4.6: Cross-validated loss

Degrees	TREES	KNN	NB-MVNM	RF	NB-KB	AdaBstM2
0°	0.0031	0.0032	0.0014	0.0029	0.5512	0.5361
45°	0.0012	0.0021	0.0012	0.0012	0.4854	0.4500
90°	0.0023	0.0066	0.0018	0.002	0.4227	0.4265

from Classification Trees (TREES), K-Nearest Neighbors (KNN) using Euclidean nearest neighbors search method and nine neighbors, Naive Bayes using multivariable multinomial distribution (NB - MVNM), Naive Bayes using uniform kernel smoothing density estimate (NB - KB), Random Forest using trees for learners (RF) and Adaptive boosting using trees for learners (AdaBstM2). In Table 4.6 the cross-validation using average loss over 10 folds is shown. As stated, the results correctly identify the devices but as the signal-to-noise level is very distinct between smartphones, it suggests that the separation is also vulnerable to changes in volume level. Next, the more difficult task of classifying loudspeakers from the same smartphone model is discussed.

4.3.3 Identifying speakers from the same smartphone model

The distinguish between different loudspeakers made for the same smartphone model is a trickier issue. First five identical loudspeakers (labels A to E) for the Samsung Galaxy J5 are used for the initial analysis. Another 11 identical loudspeakers are added to this set (labels F to P). For the tests that followed, the loudspeakers were taken apart from used smartphones, soldered to new wires, and attached to the same smartphone. A wide range of variables, such as manufacturing variabilities, material aging, physical stress, the various volume levels at which they were typically played, or other environmental effects during the usage of those second-hand smartphones, may have contributed to the audible differences between the loudspeakers. However, similar effects are present in smartphones used in the real world, making the results accurate predictors of real-life scenarios.

Three experiments are conducted, in which the first five loudspeakers were positioned in front of three different background materials: an acrylic board, a sound-dampening material (felt), and the smartphone cover, which is the primary use case. The impact of the positioning of the loudspeakers and the background material is shown in Figure 4.14. The smartphone case amplifies frequencies below 7 kHz that are difficult to distinguish on dampening boards or Plexiglas. This demonstrates how the loudspeaker case has significant impact on the quality of the sound. Fortunately, the situation will be the same for the same smartphone model. Evaluating variations caused by physical damage to the smartphone case or the use of other outer shells is outside the scope of this analysis.

In light of the three different background materials, the separation based on low and high roll-offs is further examined. The separation is depicted in Figure 4.15 utilizing the first and third separation sectors. According to the slope of their roll-offs, the five loudspeakers appear to divide into groups. When the loudspeakers are positioned in the original case, the separation is more visible, but it is also noticeable when they are positioned on the soundproofing material. When the loudspeakers are mounted on the acrylic board, there is more confusion between them, which may be caused by the stronger sound reflections of the acrylic board.

Since the overlap between loudspeakers B, D, and E is more pronounced when ex-

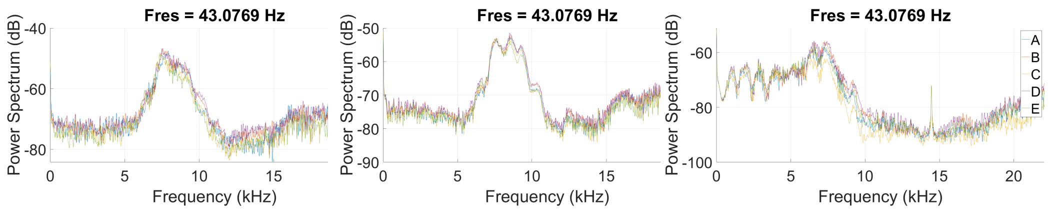
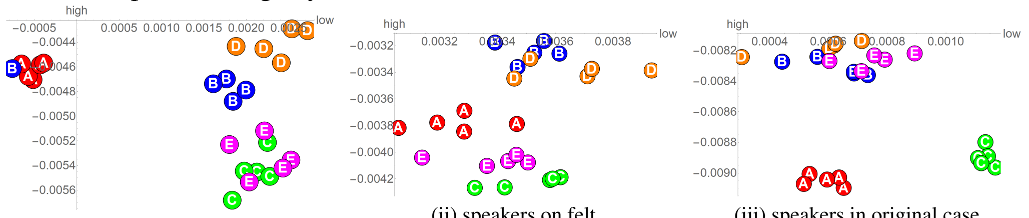
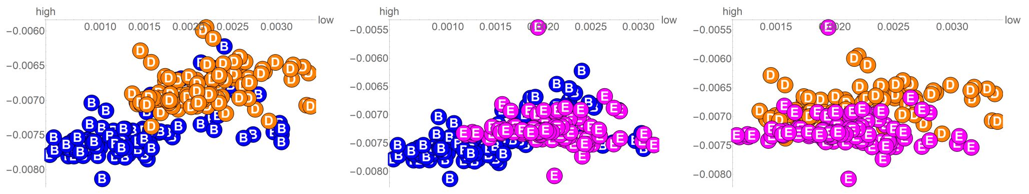


Figure 4.14: Power spectrum for the recorded signal using the five loudspeakers on acrylic board (left), damping material (middle) and inside the smartphone case (right) for linear sweep (recordings by infotainment unit)



(i) speakers on acrylic (ii) speakers on felt (iii) speakers in original case

Figure 4.15: Separation based on low and high roll-offs for first five identical J5 loudspeakers A to E



(i) separation between speakers B and D (ii) separation between speakers B and E (iii) separation between speakers D and E

Figure 4.16: Separation based on low and high roll-offs for the three closer J5 loudspeakers B, D and E using 100 samples

amining the results for the original example, which is also more applicable in practice, the distinction between them needs to be more rigorously handled. 100 measurements with each loudspeaker are required to achieve this. Figure 4.16 shows the clustering in this case. The following separation ratios are obtained by calculating the Euclidean distance of each sample from the mean of the samples from the same loudspeaker, or the intra-distance, and from the mean of the samples from the other loudspeaker, or the inter-distance: 83% between B and D, 77% between B and E, and 73% between D and E. Despite the possibility of outliers, the separation becomes clear with additional measurements.

Now, a quick quantitative analysis of the inter- and intra-distances that can be derived from the roll-off slopes is performed. The inter- and intra-distances between the 12 different smartphones are displayed in Figure 4.17 as numerical values. To make it

	T7	J5′	J5″	LE	N8	A21	X3	X7	S7	J5‴	LG	AV
T7	**0.0001**	0.0053	0.0034	0.0059	0.0031	0.0019	0.0018	0.0069	0.0031	0.0036	0.0097	0.0038
J5′	0.0053	**0.0003**	0.0022	0.0037	0.0083	0.0061	0.0062	0.0081	0.0085	0.0018	0.0141	0.0059
J5″	0.0034	0.0022	**0.0004**	0.0045	0.0065	0.0040	0.0046	0.0078	0.0065	0.0005	0.0127	0.0051
LE	0.0059	0.0037	0.0045	**0.0004**	0.0079	0.0075	0.0056	0.0048	0.0087	0.0041	0.0121	0.0039
N8	0.0031	0.0083	0.0065	0.0079	**0.0001**	0.0039	0.0023	0.0067	0.0013	0.0067	0.0068	0.0044
A21	0.0019	0.0061	0.0040	0.0075	0.0039	**0.0002**	0.0035	0.0088	0.0032	0.0044	0.0108	0.0057
X3	0.0018	0.0062	0.0046	0.0056	0.0023	0.0035	**0.0004**	0.0054	0.0031	0.0047	0.0081	0.0025
X7	0.0069	0.0081	0.0078	0.0048	0.0067	0.0088	0.0054	**0.0001**	0.0080	0.0076	0.0080	0.0030
S7	0.0031	0.0085	0.0065	0.0087	0.0013	0.0032	0.0031	0.0080	**0.0002**	0.0068	0.0078	0.0055
J5‴	0.0036	0.0018	0.0005	0.0041	0.0067	0.0044	0.0047	0.0076	0.0068	**0.0002**	0.0128	0.0050
LG	0.0097	0.0141	0.0127	0.0121	0.0068	0.0108	0.0081	0.0080	0.0078	0.0128	**0.0002**	0.0083
AV	0.0038	0.0059	0.0051	0.0039	0.0044	0.0057	0.0025	0.0030	0.0055	0.0050	0.0083	**0.0005**

Figure 4.17: Inter- and intra-distances of the slope for the 12 distinct smartphones as numerical values

	A	B	C	D	E	F	G	H	I	J	K	L	M	N	O	P
A	**0.0002**	0.0035	0.0006	0.0037	0.0037	0.0021	0.0014	0.0009	0.0019	0.0014	0.0021	0.0011	0.0008	0.0020	0.0016	0.0018
B	0.0035	**0.0009**	0.0031	0.0013	0.0008	0.0046	0.0037	0.0038	0.0044	0.0038	0.0047	0.0032	0.0034	0.0032	0.0031	0.0029
C	0.0006	0.0031	**0.0003**	0.0032	0.0032	0.0025	0.0017	0.0014	0.0023	0.0017	0.0025	0.0013	0.0011	0.0021	0.0016	0.0018
D	0.0037	0.0013	0.0032	**0.0011**	0.0010	0.0050	0.0041	0.0041	0.0048	0.0041	0.0051	0.0035	0.0037	0.0036	0.0035	0.0033
E	0.0037	0.0008	0.0032	0.0010	**0.0005**	0.0048	0.0039	0.0040	0.0046	0.0039	0.0049	0.0033	0.0036	0.0033	0.0032	0.0030
F	0.0021	0.0046	0.0025	0.0050	0.0048	**0.0008**	0.0014	0.0014	0.0009	0.0014	0.0010	0.0019	0.0017	0.0022	0.0020	0.0022
G	0.0014	0.0037	0.0017	0.0041	0.0039	0.0014	**0.0003**	0.0006	0.0008	0.0004	0.0011	0.0008	0.0007	0.0011	0.0008	0.0011
H	0.0009	0.0038	0.0014	0.0041	0.0040	0.0014	0.0006	**0.0003**	0.0010	0.0007	0.0012	0.0008	0.0005	0.0015	0.0011	0.0014
I	0.0019	0.0044	0.0023	0.0048	0.0046	0.0009	0.0008	0.0010	**0.0003**	0.0008	0.0006	0.0014	0.0013	0.0016	0.0015	0.0017
J	0.0014	0.0038	0.0017	0.0041	0.0039	0.0014	0.0004	0.0007	0.0008	**0.0003**	0.0011	0.0008	0.0008	0.0011	0.0008	0.0011
K	0.0021	0.0047	0.0025	0.0051	0.0049	0.0010	0.0011	0.0012	0.0006	0.0011	**0.0005**	0.0016	0.0015	0.0018	0.0017	0.0019
L	0.0011	0.0032	0.0013	0.0035	0.0033	0.0019	0.0008	0.0008	0.0014	0.0008	0.0016	**0.0004**	0.0005	0.0011	0.0006	0.0008
M	0.0008	0.0034	0.0011	0.0037	0.0036	0.0017	0.0007	0.0005	0.0013	0.0008	0.0015	0.0005	**0.0002**	0.0013	0.0008	0.0011
M	0.0020	0.0032	0.0021	0.0036	0.0033	0.0022	0.0011	0.0015	0.0016	0.0011	0.0018	0.0011	0.0013	**0.0005**	0.0007	0.0006
O	0.0016	0.0031	0.0016	0.0035	0.0032	0.0020	0.0008	0.0011	0.0015	0.0008	0.0017	0.0006	0.0008	0.0007	**0.0002**	0.0004
P	0.0018	0.0029	0.0018	0.0033	0.0030	0.0022	0.0011	0.0014	0.0017	0.0011	0.0019	0.0008	0.0011	0.0006	0.0004	**0.0001**

Figure 4.18: Inter- and intra-distances of the slope for the 16 identical J5 loudspeakers as numerical values

simple to identify the diagonal of the matrix, which stands for intra-distance (highlighted with bold). The distances between the planar coordinates created by the slopes of the low and high roll-offs were calculated as the average Euclidean distances. Except for the two identical J5 phones, the inter-distances are bigger and consistently above $10^{˘3}$, the intra-distances are consistently below this amount. The inter- and intra-distances between the 16 identical J5 loudspeakers are displayed in Figure 4.18 as numerical values. While in this situation, the intra-distances almost always fall below the $10^{˘3}$ threshold (loudspeaker D is the lone exception), it is possible for the inter-distances to also fall below this level. As a result, it is more challenging to separate based just on the slope of the roll-offs. The values are also noisier, and while the diagonal, or intra-distances, is still apparent, it is more difficult to detect.

As a partial conclusion, it can be said that the slopes may offer enough hints to distinguish between different smartphones, but they may cause issues when identical loudspeakers or smartphones are utilized.

4.3.4 Noise influence on roll-offs slopes

The impact of noise is also analyzed, keeping in mind that in a real-world scenario, background noises are present and can affect the fingerprinting mechanism of the loud-

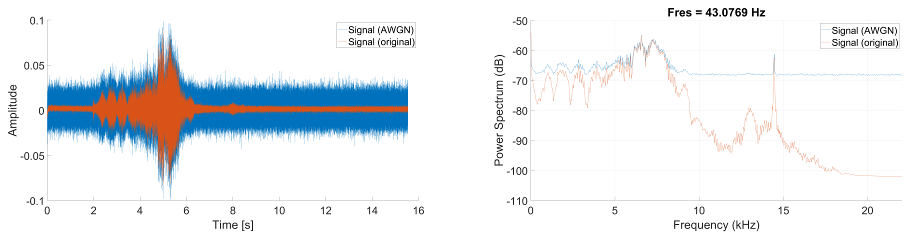


Figure 4.19: The recorded audio signal and the signal with AWGN time domain (left) and the power spectrum (right)

speakers. Two significant types of noise are taken into account: *additive white Gaussian noise* (AWGN), which imitates the impacts of several random processes seen in nature and may also account for noise inside cars, and *street noise*, which is unique to the situation involving cars. In [218], the attenuation of the sound from the loudspeaker to the microphone was also simulated using the additive white Gaussian noise.

By setting the SNR (signal-to-noise ratio) to 0dB, an AWGN noise is applied over the clean recordings with noise levels proportionate to the original signal power. The signal played by the loudspeakers of the Samsung J5 while being recorded by the infotainment unit is contained in the clean recordings (the loudspeakers were positioned within the smartphone casing as previously mentioned). Figure 4.19 shows the recorded audio signal in the time domain (left), the signal with AWGN, and the power spectrum of the signals (right).

The AWGN, that affects the linear sweep sound emitted by the first five loudspeakers of the Samsung J5 are analyzed next. The frequency between 700Hz and 11kHz is divided into three sectors based on the power spectrum of the audio signal with AWGN (the same as was done previously). Plots from two experiments with a linear fit to the power spectrum signals for the first sector (left) and third sector (right) are shown in Figure 4.20. The five loudspeakers of the Samsung J5 are displayed on each plot at 100% volume. The loudspeakers can be distinguished based on the slope of the linear approximation for each of the three regions. The separation based on the low and high roll-offs is depicted on the left side of Figure 4.21. When AWGN is included, some clustering still appears in the data, but overlaps are more pronounced.

The infotainment unit inside the vehicle in a parking lot close to an urban road having four lanes on a two-way street with tram lines is used to examine the impact of the *street noise* on fingerprinting the loudspeakers. The five speakers of the Samsung J5 emitted a linear sweep signal recorded by the infotainment system, which was mounted in the center of the dashboard. The passenger held the Samsung J5 roughly 50 cm from the microphone of the infotainment unit. The front left window was open, letting in as much street noise as possible. In a separate recording, the street noise is captured using the infotainment system in the same parking lot with the front left window of the car opened. The first five loudspeakers of the Samsung J5, which are housed inside the smartphone

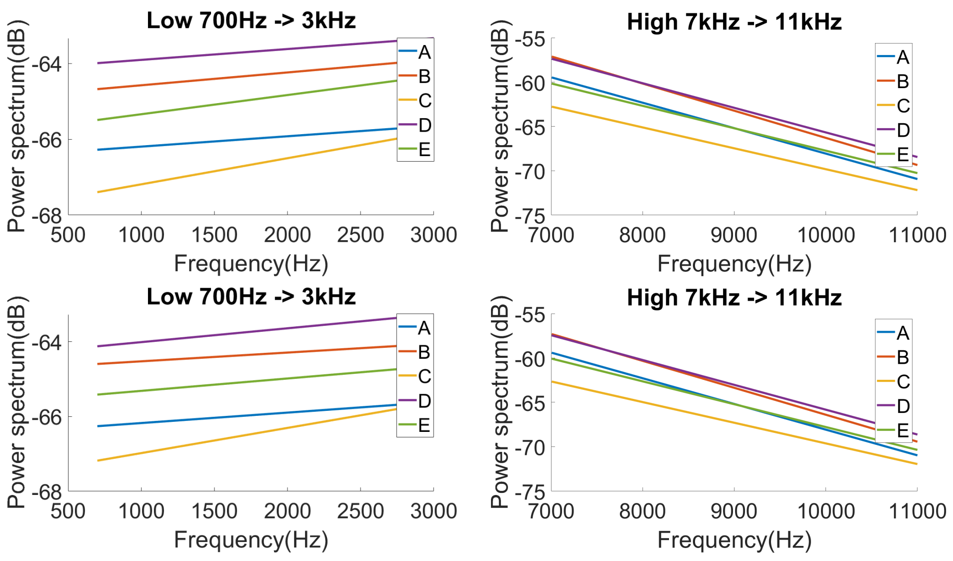


Figure 4.20: Linear fit results over the recorded audio signal from two experiments (up, down) depicting five loudspeakers of the Samsung J5 in the low (left) and high sector (right).

case as explained in Section 4.1 are used to play the recorded audio signal once the recorded street noise has been added (as also done in the related work from [218]). The differentiation between loudspeakers was still visible after identical noise was introduced to the recording. Repeated measurements taken inside the vehicle with the left window opened revealed a less distinctive separation, as seen on the right side of Figure 4.21. This is very likely caused by distinct street noises at each new measurement, such as shifting traffic, horn sound, passing tramways, etc.

This shows that adding artificial noise does not result in a very accurate simulation of a real-world setting, although testing a large number of loudspeakers on the street is by no means simple. For this reason, artificial AWGN noise is used later (as also done by other related works).

4.4 Loudspeaker classification with neural networks

Due to the separation between speakers B, D, and E is not very clear, Deep Learning algorithms are used to obtain more accuracy. To achieve this, 500 measurements were performed with each loudspeaker in the original case. The power spectrum for each linear sweep recorded is calculated, and then the neural networks are trained and tested on the frequencies in the range of 700Hz — 11kHz.

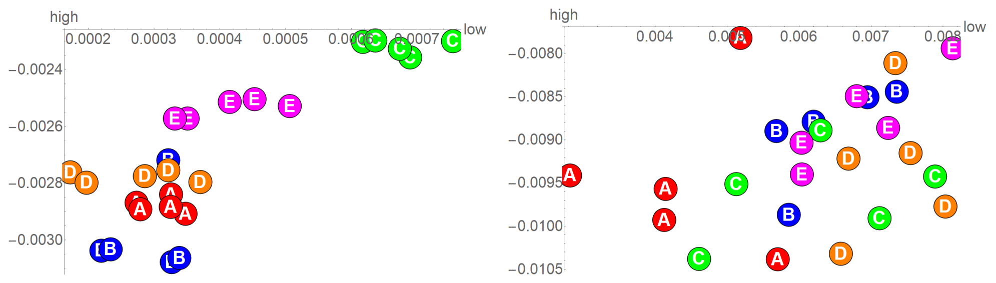


Figure 4.21: Separation based on low and high roll-offs in the case when the original signal is cumulated with AWGN signal (left) and street recording (right)

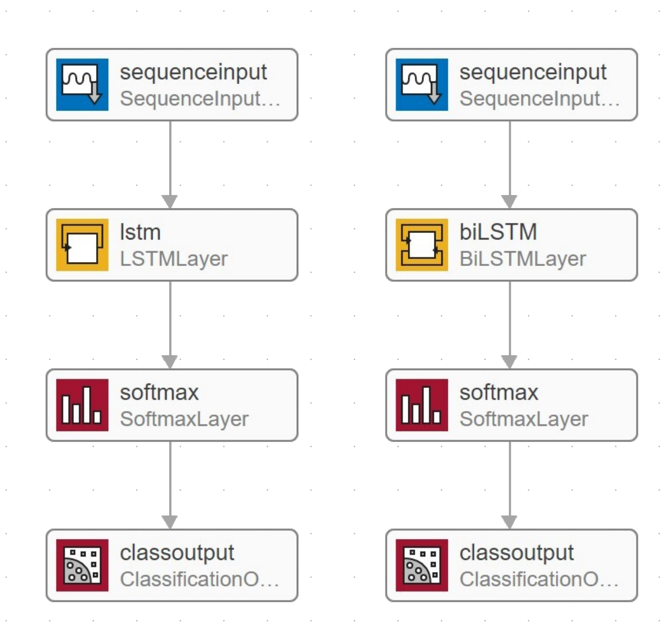


Figure 4.22: Architecture of the LSTM (left) and BiLSTM (right) neural network

4.4.1 Recurrent neural networks

The proposed neural network contains a sequence input layer followed by a Long Short-term memory (LSTM) Network or by a Bidirectional Long Short-Term Memory (BiLSTM) Network with 100 hidden units, three fully connected layers, and two output layers: a softmax layer and a classification layer. In Figure 4.22 the architectures are depicted for the proposed recurrent neural networks. To improve the results for the LSTM Network, the number of hidden units varies between 100 and 250.

Regarding the analysis, 30% of the dataset is used for testing, and 70 percent is used

Table 4.7: Neural Networks experimental results

Network	Epochs	Hidden Units	Speaker	Precision	Recall
			B	100%	94.93%
BiLTSM	100	100	C	95.33%	100%
			E	99.33%	100%
			B	85.33%	82.85%
LTSM	100	100	C	86%	86.57%
			E	96%	98.63%
			B	94.66%	88.19%
LTSM	100	150	C	89.33%	97.1%
			E	97.33%	97.33%
			B	90.66%	90.66%
LTSM	150	100	C	92%	95.17%
			E	98%	94.83%
			B	93.33%	87.5%
LTSM	100	250	C	89.33%	93.05%
			E	96.66%	99.31%
			B	90.66%	91.89%
LTSM	200	250	C	98%	90.18%
			E	92.66%	100%

for training and validation. The proposed optimization algorithm is the stochastic gradient descent with momentum (SGDM) with the momentum value set to 0.9, representing the contribution from the previous step. The maximum number of epochs to use for training is initially set to 100, and, for the LSTM Network, it is increased to 200. In Table 4.7, the results are summarized, and can it can be seen that the best results are obtained using BiLTSM, even if, for the LSTM, the number of epochs and the number of hidden units were increased.

4.5 Concluding remarks

Several fingerprinting techniques were investigated that are simple to use to identify smartphones based on their *speaker roll-off characteristics*. According to the results, loudspeaker roll-offs offer a reliable fingerprint more resistant to variations in volume levels. In contrast, for some techniques, the volume level may be misleading. The methods that can be utilized to fingerprint smartphones and make them more useful as smart keys were discussed. Existing procedures from the literature were also pointed out. One such specific use case is inside vehicles, which is why in this analysis an in-vehicle head unit was used to collect the audio signals from the smartphones (a calibrated microphone was occasionally used as a reference). The results suggest that in-vehicle head units may be useful in this situation, given the high success rates of the identification.

Chapter 5

Fingerprinting smartphones based on microphones characteristics

This chapter is based on the results of the author that are published in [22] and addresses smartphones fingerprinting based on microphone characteristics. Experiments with 16 identical and 16 different smartphones were done, trying to identify them based on characteristics extracted from the recorded sound.

5.1 Microphone-based fingerprinting, brief motivation

Machine learning classifiers and the frequency domain representation of the recorded sounds are used to assess smartphone fingerprints provided by microphone features. Several conventional machine learning methods have been used, including Ensemble-Subspace Discriminant (ENS), Linear Discriminant (LD), Fine Nearest Neighbor (KNN), Fine Tree (TREE) and Linear Support Vector Machines (SVM).

Vehicle environments, which have recently received a lot of attention as a result of the transformation of the automotive sector toward interconnections with smart devices brought by users, are one particular application area that needs to be investigated. Due to the various sounds and environments, this analysis is concentrated on three separate scenarios, as shown in Figure 5.1 and discussed in what follows.

Scenario A: Fingerprinting smartphones from different brands and models based on human speech: Human speech is used in this case to identify various manufacturers and models of smartphones. The already-existing MOBIPHONE dataset [105], a public voice database that includes 21 smartphones from various manufacturers and models, is used for this scenario. The dataset for each smartphone includes 24 audio samples from 12 male and 12 female speakers. The speakers were picked from the TIMIT database [219]. Ten spoken sentences are included in each recorded file, the first two of which are the same for each speaker and the remaining eight are unique.

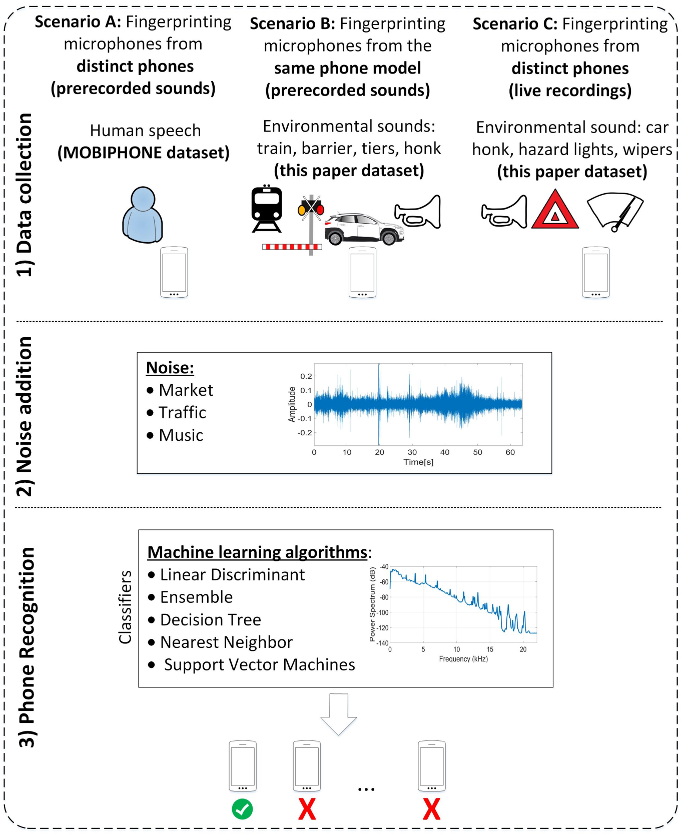

Scenario A: Fingerprinting microphones from distinct phones (prerecorded sounds)
Scenario B: Fingerprinting microphones from the same phone model (prerecorded sounds)
Scenario C: Fingerprinting microphones from distinct phones (live recordings)
1) Data collection
Human speech (MOBIPHONE dataset)
Environmental sounds: train, barrier, tiers, honk (this paper dataset)
Environmental sound: car honk, hazard lights, wipers (this paper dataset)
2) Noise addition
Noise:
• Market
• Traffic
• Music
Amplitude
Time[s]
3) Phone Recognition
Classifiers
Machine learning algorithms:
• Linear Discriminant
• Ensemble
• Decision Tree
• Nearest Neighbor
• Support Vector Machines
Power Spectrum (dB)
Frequency (kHz)
...

Figure 5.1: Overview of the methodology and scenarios used in this chapter

Scenario B. Fingerprinting identical smartphones based on environmental sound using prerecorded sounds: For this scenario, special recordings were done using 16 microphones from the same smartphone (a Samsung Galaxy S6) that were utilized to capture road and vehicle noise that was then played by a high-fidelity audio system. Because it is much simpler to test multiple identical microphone models connected to the same phone inside a controlled environment, these experiments were carried out to ascertain whether the fingerprinting process is influenced by the microphone alone (or by other circuits in the smartphone) or by both. To create the environmental sound, recordings from the SoundArchive `https://www.soundarchive.online/?s=police` database were selected, which are correlated to various situations that are frequently experienced in moving vehicles: the sound of (i) a locomotive signaling departure, (ii) bells jingling as barriers close, (iii) cars screeching as they approach tiers and (iv) a vehicle's horn. Figure 5.2 shows the sounds from SoundArchive played in the indoor experiments (using the same microphones) in the time domain (left) and their power spectrum, or frequency domain representation (right). Two signals, which are the two channels of a stereo recording, are shown on each plot.

Scenario C: Fingerprinting smartphones from different brands and models based on live recordings: For this scenario, a special dataset was created by recording the sound on 16 smartphones both outside and inside a car. Each smartphone records three different sub-scenarios:

1. A vehicle honking in an open area was chosen to prevent reflections from nearby objects. In this instance, as expected in the event of incidental bystander recordings, the smartphones were left outside the vehicle. With each smartphone 400 measurements were done, resulting a total of 6400 measurements in this scenario.

2. In many situations connected to traffic conditions, the vehicle hazard lights inside cars are frequently activated. A total of 4800 measurements were performed for this scenario, 300 for each smartphone.

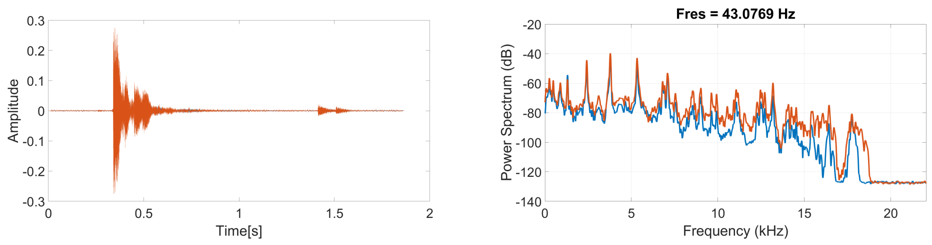
3. It's also normal to hear wiper noise inside cars, but this is usually due to environmental factors. The devices were kept inside the vehicle in these latter two scenarios. A total of 4800 measurements were performed for this scenario, 300 for each smartphone.

Additional noise might be present in the environment in real-world situations. A reason for which the impact of three different types of noises was analyzed on this fingerprinting process. For outdoor recordings, overlaps with music are considered. For this, several songs from the top 10 Spotify charts of 2021 are used. Two environmental noises from the SoundArchive are chosen for inside recordings: (i) sounds of heavy traffic and (ii) sounds of outdoor markets. The representation of these sounds from the SoundArchive in the time domain and frequency domain are shown in Figure 5.3. Since the SoundArchive files are on two-channel, i.e., stereo recordings, each plot has two signals.

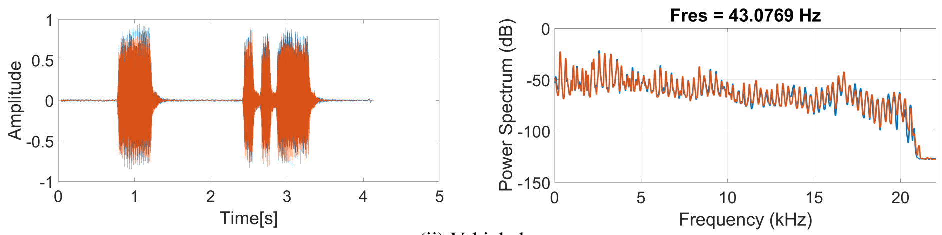
5.2 Setup and methodology

This section summarizes the hardware, setup and software platforms used in the experiments.

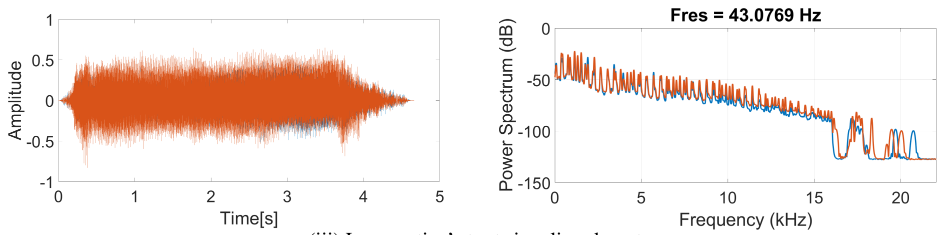
The experiments concentrate on classifying distinct and identical smartphones using microphones in such devices. A Samsung Galaxy S6 smartphone is disassembled and 16 identical flex cables with microphones are purchased to conduct the experiments convincingly and account for variations between identical microphones. The micro USB charging port, jack connector, navigation key and capacitive keys and the microphone are all located on the same board. The capacitive keys are taken off the board to make replacing the flex wires easier. An overview of the devices and measurements is provided in Table 5.1. A total of 32 smartphones have been fingerprinted, 16 of which have identical



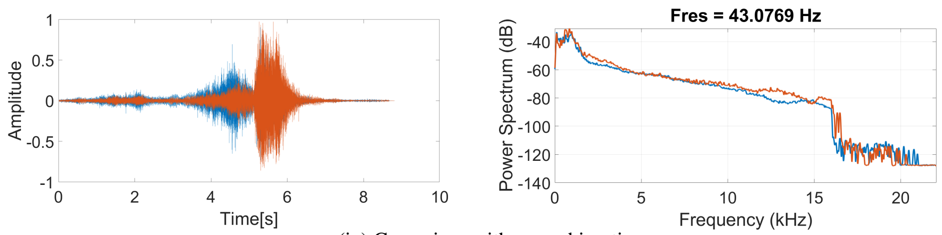
(i) Two bells jingling when barriers close



(ii) Vehicle horn

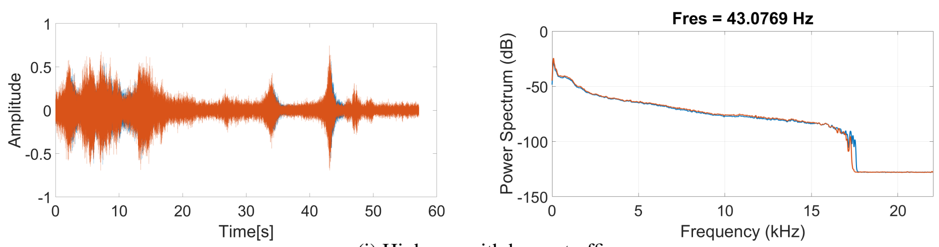


(iii) Locomotive's toot signaling departure

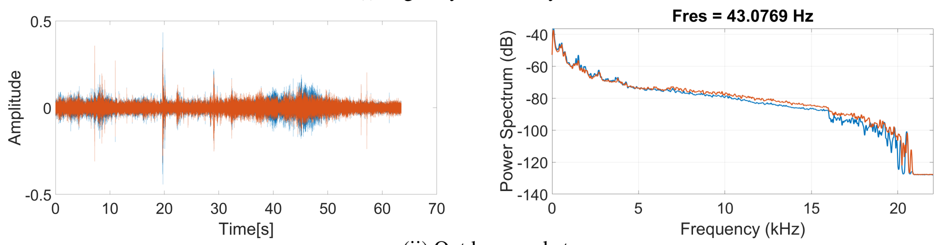


(iv) Car arrives with screeching tire

Figure 5.2: Sounds used in the experiments in time domain (left) and the power spectrum (right)



(i) Highway with heavy traffic



(ii) Outdoor market

Figure 5.3: Noises used in the experiments in time domain (left) and the power spectrum (right)

Table 5.1: Summary of devices and associated measurements

	Phones	Label	No.	Mic.	Meas.	Total
1.	Samsung Galaxy S6	A, C and F to Q	16	1	200	3200
2.	Samsung Galaxy S6 (other)	S6 and S6′	2	1	1000	2000
3.	Allview V1 Viper I	AV	1	1	1000	1000
4.	Samsung Galaxy J5	J5, J5′ and J5″	3	1	1000	3000
5.	One Plus 7 Pro	OP	1	2	1000	1000
6.	Samsung Galaxy Tab S7	S7t	1	2	1000	1000
7.	Leagoo Z10	LE and LE’	2	1	1000	2000
8.	Samsung Galaxy A21s	A21s	1	2	1000	1000
9.	Samsung Galaxy S7	S7	1	1	1000	1000
10.	Samsung Galaxy A3	A3	1	2	1000	1000
11.	Motorola E6 plus	MT	1	1	1000	1000
12.	Google Nexus 7	N7	1	2	1000	1000
13.	LG Optimus P700	LG	1	1	1000	1000
	Total		32			19200

microphones from Samsung Galaxy S6 smartphones that are kept in the same case. The two sides of a disassembled Samsung Galaxy S6 smartphone are shown in Figure 5.4, along with a flex cable nearby. The 16 identical microphones from the Samsung Galaxy S6 on their flex cables are shown in Figure 5.5. The table shows that the remaining 16 devices are distinct smartphones from various manufacturers.

Matlab `https://nl.mathworks.com/products/matlab.html`, a numerical computation environment frequently used for data analysis and model development,

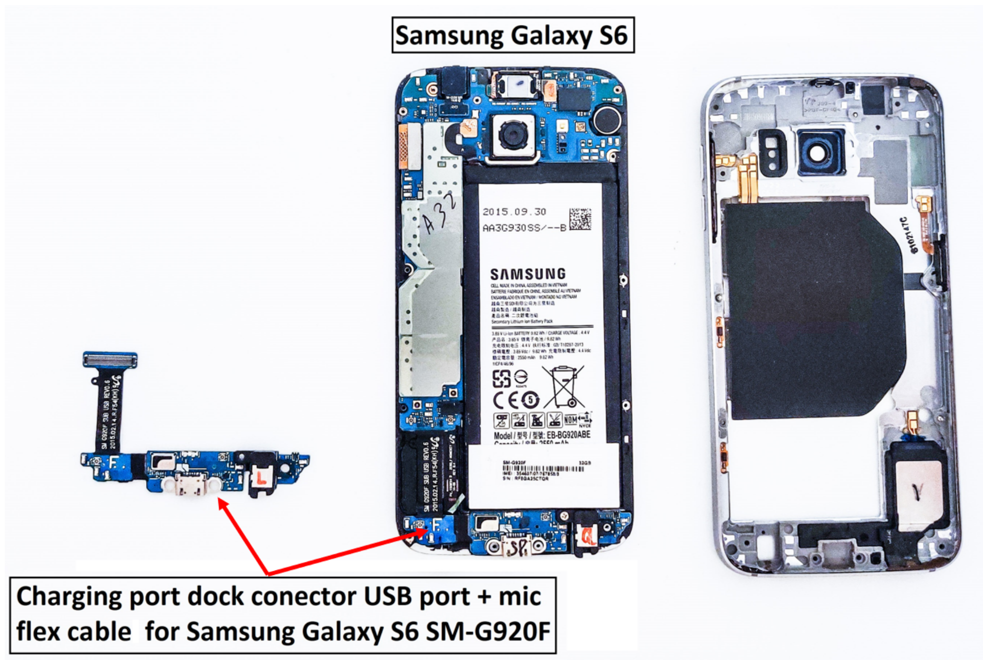

Samsung Galaxy S6
Charging port dock conector USB port + mic flex cable for Samsung Galaxy S6 SM-G920F

Figure 5.4: Samsung Galaxy S6 dissembled, two flex cables with charging USB port dock connector and microphone

is used to analyze the recorded data. The Signal Analyzer application from Matlab 2021a is used for the initial analysis of the recorded data. The Matlab Classification Learner application is used to investigate the classification methods. Additionally, the free room acoustic program, Room EQ Wizard (REW) `http://roomeqwizard.com/`, is used for the first setup calibration.

The recordings from the MOBIPHONE dataset [105] are used for Scenario A. Since rewiring or replacing the smartphone's microphone is required for the experiments in scenario B, where microphones for the identical Samsung Galaxy S6 smartphone are fingerprinted based on environmental noise, indoor measurements with previously recorded noises are done. This is because rewiring is challenging to complete outdoors. Additionally, because each microphone must be independently plugged into the phone, it is impossible to record with several microphones simultaneously on the same phone, resulting in different ambient circumstances.

The indoor experimental setup is shown graphically in Figure 5.6. In the experiments, a high-fidelity audio system is selected that could reproduce sounds with a more linear response since the target is to reproduce a wide frequency spectrum and low-cost speakers cannot handle this and create more distortions. Two professional loudspeakers that can deliver a more accurate low-frequency response are part of the audio system employed in the tests. The positioning of the speakers and the acoustic envi-

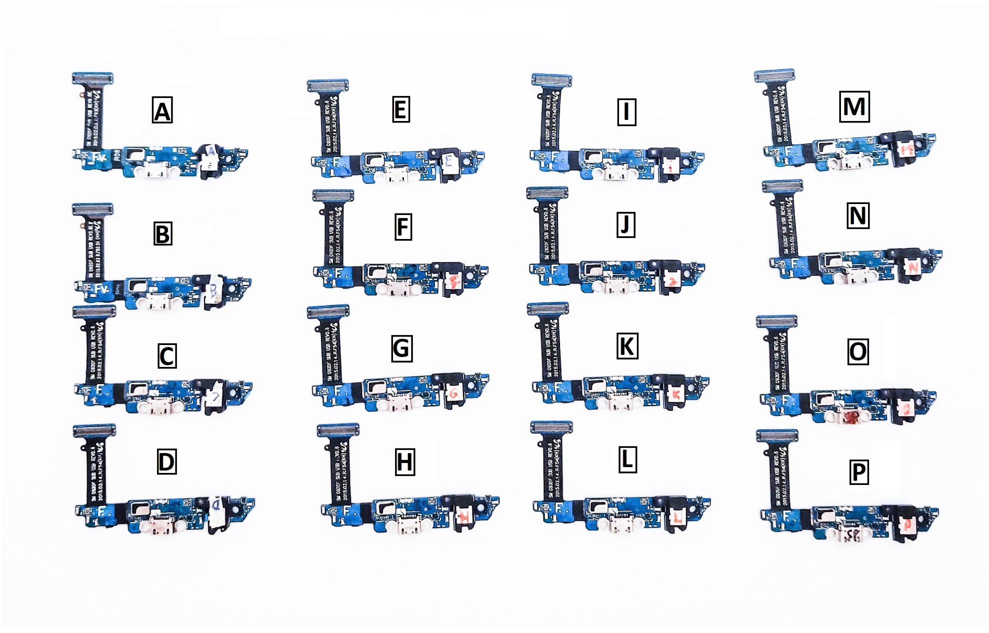


Figure 5.5: 16 flex cables with charging USB port dock connector and microphone for Samsung Galaxy S6

ronment is also important for high-quality reproduction. The recorder and speakers were arranged in an equilateral triangle with 150 cm between them and a 60° interior angle (as advised for stereo reproductions `https://theproaudiofiles.com/better-acoustics-in-your-home-studio/`). The speakers' distance from the rear wall was 50 cm. To prevent sound reflections and reverberations, an acoustic-absorbing material was applied to the front, back and floor, as well as the side walls at mirror locations. Additionally, the room's corners were sealed `http://nzacoustics.com/PolyesterPanelsColoured.htm`. The frequency response of the audio system utilized in this analysis, measured in REW with a linear sweep signal created between 0Hz and 20kHz, is shown in Figure 5.7. This recording was made using the calibrated UMIK-1 Omni-directional USB microphone from miniDSP. In range of +/-5db, the response is adequately linear. For scenario B, each MP3 file from SoundArchive with in-car and traffic noises is played, including: (i) a locomotive's long toot, (ii) barriers with two bells jingling, (iii) a car arriving with screeching tiers and (iv) a vehicle horn. An Android smartphone app is used to record and save the sounds as PCM and WAV files for analysis. The experiments in scenario C were conducted outside using 16 different smartphones from distinct manufacturers, which recorded the following sounds:

1. A vehicle honked 400 times in real time. This experiment was conducted outside.

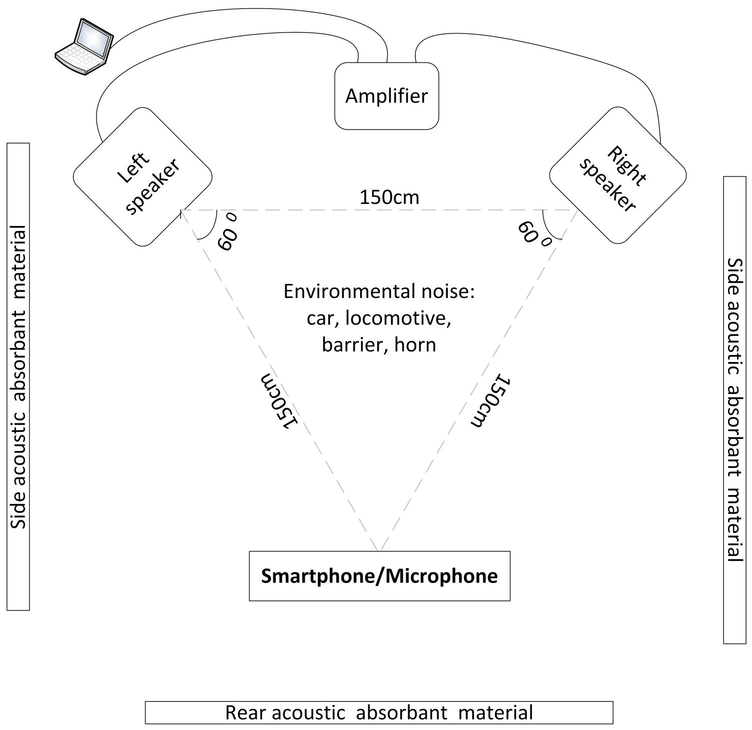

Amplifier
Left speaker
Right speaker
150cm
60 o
60 o
Environmental noise: car, locomotive, barrier, horn
150cm
150cm
Side acoustic absorbant material
Side acoustic absorbant material
Smartphone/Microphone
Rear acoustic absorbant material

Figure 5.6: Suggestive illustration of the indoor experimental setup

The smartphones were placed on a board on the front-right side of the car, 3 meters away from the vehicle, as shown in Figure 5.8. As the vehicle honked, some background noise was audible in the captured files. Since the honk was manually activated 400 times, not all of the honks are identical and some are shorter than others, making the situation more complicated.

2. Hazard lights inside the car blinked 300 times. The car was parked in front of a house, close to a street with no traffic and the engine ran at idle speed during the experiment. The smartphones were positioned side by side on the back seat, as shown in Figure 5.8.

3. The vehicle wipers which operate at a low speed for 300 times. The car was parked in front of the house, near a street with no traffic, the engine was running at idle speed and a garden hose was used to artificially water the windshield during the experiment. Once more, the smartphones were positioned close to one another on the back seat, as shown in Figure 5.8.

Moreover, an Android application ran on the smartphones to record and save the sounds as PCM and WAV files for further analysis.

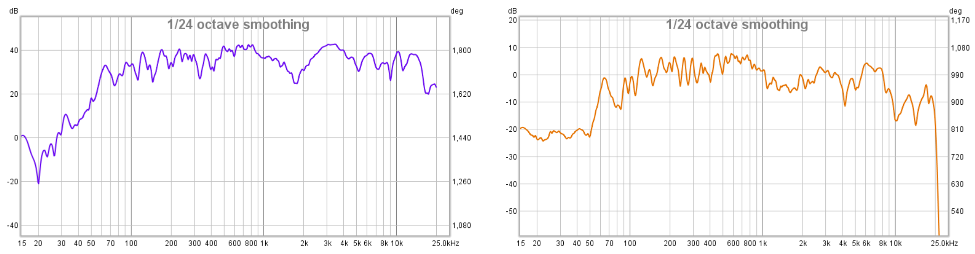


Figure 5.7: Frequency response of the audio system which was used in the experiments validated with miniDSP UMIK-1 microphone (left) and with a smartphone (right)

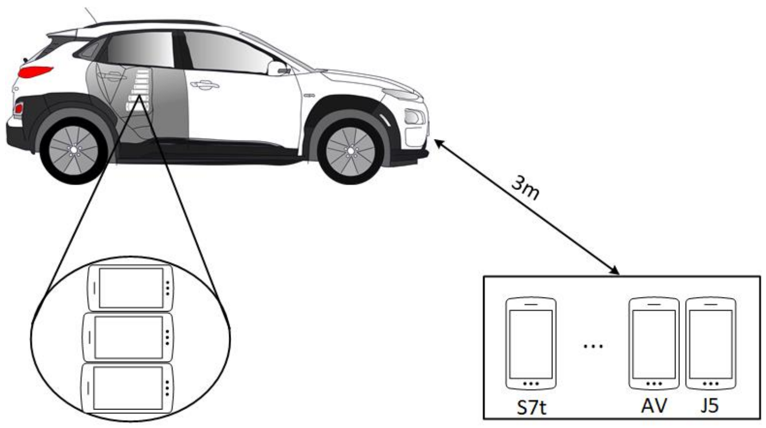


Figure 5.8: Suggestive illustration of the outdoor experimental setup

5.3 Fingerprinting microphones using prerecorded sounds

The conventional machine learning techniques LD, ENS, TREE, KNN and SVM classifiers are used to assess microphone characteristics retrieved from the power spectrum of the recorded data to identify the microphones.

The power spectrum for each signal is extracted and used in the classifiers. The inputs for the classifiers will consist of the 4096 features for each audio signal, as the power spectrum is a vector with 4096 elements.

Additionally, the effects of ambient noise on each dataset used for fingerprinting are analyzed. In other words, the noise is added to the original signal at different SNR levels, i.e., $\mathsf{SigWithNoise} = \mathsf{Sig} + \mathsf{NoiseAmp}$. Here, $\mathsf{SigWithNoise}$ denotes the signal with noise, Sig denotes the original signal (recorded in the time domain using the tested microphones) and $\mathsf{NoiseAmp}$ is the traffic or market noise as downloaded from SoundArchive. The $\mathsf{NoiseAmp}$ is the noise that has been amplified by a particular SNR factor that is calculated as:

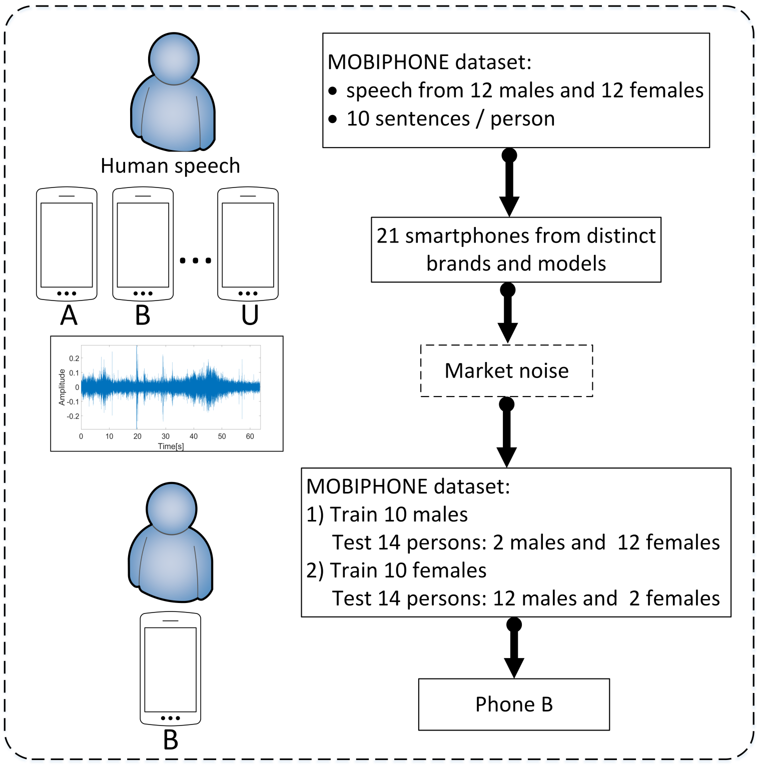

Human speech
A
B
U
MOBIPHONE dataset:
• speech from 12 males and 12 females
• 10 sentences / person
21 smartphones from distinct brands and models
Market noise
MOBIPHONE dataset:
1) Train 10 males
Test 14 persons: 2 males and 12 females
2) Train 10 females
Test 14 persons: 12 males and 2 females
Phone B
B

Figure 5.9: Overview of the procedure for smartphone identification using human speech (Mobiphone Dataset)

$$\mathsf{NoiseAmp} = \mathsf{Noise} \times \mathsf{Fac} \times \frac{\mathsf{MaxNoise}}{\mathsf{MaxSig}}.$$

Here, Noise denotes the time-domain noise signal from SoundArchive, MaxNoise denotes the noise's maximum absolute value, MaxSig denotes the signal's maximum absolute value and Fac is the scalar amplification factor. The SNR is determined by:

$$\mathsf{SNR} = 10 \times \log_{10} \frac{\mathsf{OrigBandPower}}{\mathsf{NoiseBandPower}} [\mathsf{dB}].$$

Where the average power of the original signal (the signal captured by the microphones) is OrigBandPower and the average power of the noise is NoiseBandPower (market or traffic noise from the MP3 file on SoundArchive).

Table 5.2: Precision, recall and accuracy for 5 classifiers (MOBIPHONE dataset)

training	metrics	LD	ENS	TREE	KNN	SVM
mobiphone (10 males)	precision	0.98	0.96	0.69	0.83	0.81
	recall	0.98	0.96	0.76	0.85	0.84
	accuracy	0.98	0.97	0.74	0.76	0.79
mobiphone (10 females)	precision	0.96	0.95	0.72	0.91	0.87
	recall	0.97	0.96	0.74	0.92	0.90
	accuracy	0.99	0.99	0.64	0.86	0.79

5.3.1 Fingerprinting microphones using human speech

The MOBIPHONE dataset [105], which contains 21 smartphones from different manufacturers and models, is used to fingerprint smartphones based on the human voice. There are 24 audio files from 24 speakers, 12 men and 12 women, for each smartphone. The power spectrum, also known as the frequency response, is computed for each audio file. This information is utilized as input for the classifiers.

	A	B	C	D	E	F	G	H	I	J	K	L	M	N	O	P	Q	R	S	T	U
LD	0.	0.	0.007	0.	0.	0.	0.	0.007	0.	0.007	0.	0.	0.	0.003	0.	0.003	0.003	0.	0.	0.	0.
ENS	0.	0.003	0.010	0.	0.	0.	0.003	0.007	0.	0.	0.	0.	0.	0.007	0.	0.010	0.	0.	0.	0.	0.
TREE	0.003	0.003	0.032	0.007	0.007	0.010	0.028	0.017	0.007	0.021	0.024	0.021	0.	0.007	0.007	0.037	0.010	0.	0.031	0.007	0.003
KNN	0.	0.007	0.010	0.	0.003	0.014	0.010	0.010	0.003	0.003	0.	0.	0.003	0.003	0.	0.010	0.007	0.	0.003	0.	0.
SVM	0.	0.007	0.021	0.	0.003	0.007	0.027	0.003	0.007	0.003	0.007	0.	0.003	0.003	0.	0.010	0.017	0.003	0.003	0.	0

(i) FAR's when 10 females were used as training (MOBIPHONE)

	A	B	C	D	E	F	G	H	I	J	K	L	M	N	O	P	Q	R	S	T	U
LD	0.	0.	0.	0.	0.	0.	0.	0.	0.	0.	0.	0.	0.	0.	0.066	0.	0.133	0.066	0.125	0.176	0.
ENS	0.	0.	0.	0.	0.	0.125	0.	0.076	0.	0.	0.	0.	0.	0.	0.066	0.	0.176	0.	0.125	0.176	0.
TREE	0.133	0.187	0.642	0.333	0.142	0.153	0.647	0.250	0.294	0.111	0.222	0.333	0.066	0.200	0.076	0.	0.352	0.517	0.444	0.142	0.
KNN	0.066	0.	0.	0.	0.133	0.285	0.	0.312	0.	0.133	0.	0.	0.	0.	0.	0.083	0.333	0.	0.133	0.125	0.066
SVM	0.	0.142	0.	0.	0.133	0.520	0.	0.381	0.	0.	0.142	0.066	0.	0.	0.	0.083	0.181	0.	0.187	0.	0.176

(ii) FRR's when 10 females were used as training (MOBIPHONE)

Figure 5.10: FARs (up) and FRRs (down) as numerical values for the LD, ENS, TREE, KNN and SVM classifiers when 10 females were used as training (MOBIPHONE)

	A	B	C	D	E	F	G	H	I	J	K	L	M	N	O	P	Q	R	S	T	U
LD	0.	0.	0.	0.	0.	0.010	0.	0.	0.	0.003	0.	0.	0.	0.	0.	0.	0.	0.	0.	0.	0.003
ENS	0.	0.003	0.	0.	0.007	0.	0.	0.	0.	0.	0.	0.003	0.	0.	0.	0.003	0.003	0.	0.	0.	0.007
TREE	0.003	0.003	0.019	0.007	0.017	0.034	0.024	0.024	0.014	0.017	0.021	0.024	0.010	0.014	0.003	0.007	0.014	0.021	0.017	0.014	0.
KNN	0.003	0.021	0.010	0.007	0.	0.014	0.010	0.003	0.003	0.003	0.007	0.007	0.	0.003	0.010	0.007	0.014	0.	0.007	0.017	0.017
SVM	0.	0.021	0.021	0.010	0.017	0.007	0.	0.	0.	0.007	0.010	0.007	0.	0.	0.010	0.014	0.017	0.	0.014	0.024	0.007

(i) FAR's when 10 males were used as training (MOBIPHONE)

	A	B	C	D	E	F	G	H	I	J	K	L	M	N	O	P	Q	R	S	T	U
LD	0.	0.	0.	0.	0.	0.	0.125	0.066	0.	0.	0.	0.	0.	0.	0.	0.066	0.066	0.	0.	0.	0.
ENS	0.	0.	0.	0.	0.	0.125	0.066	0.	0.066	0.	0.	0.	0.	0.066	0.	0.133	0.	0.	0.	0.066	0.
TREE	0.	0.350	0.709	0.368	0.	0.333	0.416	0.461	0.	0.357	0.	0.222	0.	0.230	0.	0.500	0.523	0.111	0.181	0.090	0.176
KNN	0.235	0.200	0.083	0.	0.066	0.230	0.388	0.277	0.071	0.235	0.368	0.142	0.	0.071	0.	0.200	0.230	0.	0.200	0.	0.100
SVM	0.176	0.272	0.272	0.	0.	0.368	0.300	0.481	0.066	0.250	0.312	0.	0.	0.	0.	0.230	0.307	0.	0.166	0.	0.

(ii) FRR's when 10 males were used as training (MOBIPHONE)

Figure 5.11: FARs (up) and FRRs (down) as numerical values for the LD, ENS, TREE, KNN and SVM classifiers when 10 males were used as training (MOBIPHONE)

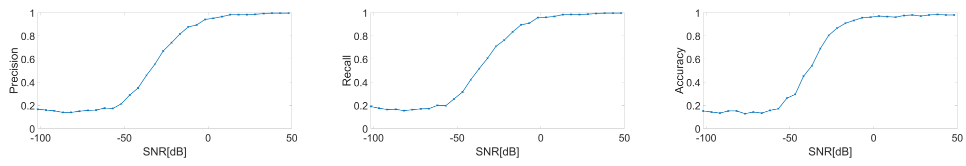


Figure 5.12: Precision (left), recall (middle) and accuracy (right) resulted with linear discriminant with noise between -80dB and 20dB with 5dB increment

Fingerprinting microphones using human speech (clean recordings): The five classifiers stated in the previous section are used to fingerprint various devices using human voice from the MOBIPHONE dataset. Two situations are considered to increase the difficulty of the identification process. First, the power spectrum of the speech of 10 male speakers is employed for training, while the speech of 12 female speakers and the remaining two male speakers is employed for testing. Second, the power spectrum of 10 female speakers is used for training, while 12 male speakers and the two remaining female speakers are used for testing. Figure 5.9 shows the flowchart for these cases. The mean precision, recall and accuracy for each classifier for the two cases in this scenario are displayed in Table 5.2. It is clear that the LD classifier produces the best results, closely followed by the ENS, while the SVM and KNN provide worse outcomes and the TREE classifier produces the worst results, probably due to its propensity for over-fitting. Figure 5.10 shows the FAR (False Acceptance Rate) and FRR (False Rejection Rate) as numerical values for every classifier and microphone for the 10 females used as training samples as an additional performance metric, more particularly focused on the authentication success rate. However, the FAR is relatively low for all classifiers. The Tree classifier on microphone P achieves the highest value of 3.7%. However, the FRR for the TREE classifier on microphones C and G is extremely high, reaching 64%. For the LD classifier, the maximum FAR value on microphones C, H and J is just 0.7% and the maximum FRR value on microphone T is only 17%. Once again, it is clear that the Tree classifier and KNN produce the lowest results, whereas the LD and ENS classifiers produce the best results. Figure 5.11 shows the FAR (up) and FRR (down) for each classifier and microphone for the 10 training males as numerical values. Again, it is clear that the Tree classifier, followed by the KNN classifier, produces the lowest results, while the LD and ENS classifiers produce the best results. The largest value for the Tree classifier on microphone F is 3.4%, but overall, the FAR is relatively low for all classifiers. For the Tree classifier on microphone C, the FRR hits 70%. The maximum FAR and FRR values for the LD classifier are 1% on microphone F and 12.5% on microphone G, respectively.

Fingerprinting microphones using human speech with market noise: Market noise is added to the signals at different SNRs, i.e., from -80dB to 20dB with an increase step of 5dB, to make the fingerprinting process more difficult and equivalent to a real-life scenario in which ambient noise is present. Only the LD classifier is used in this case because it produces the best results and is also the fastest, with a prediction speed of

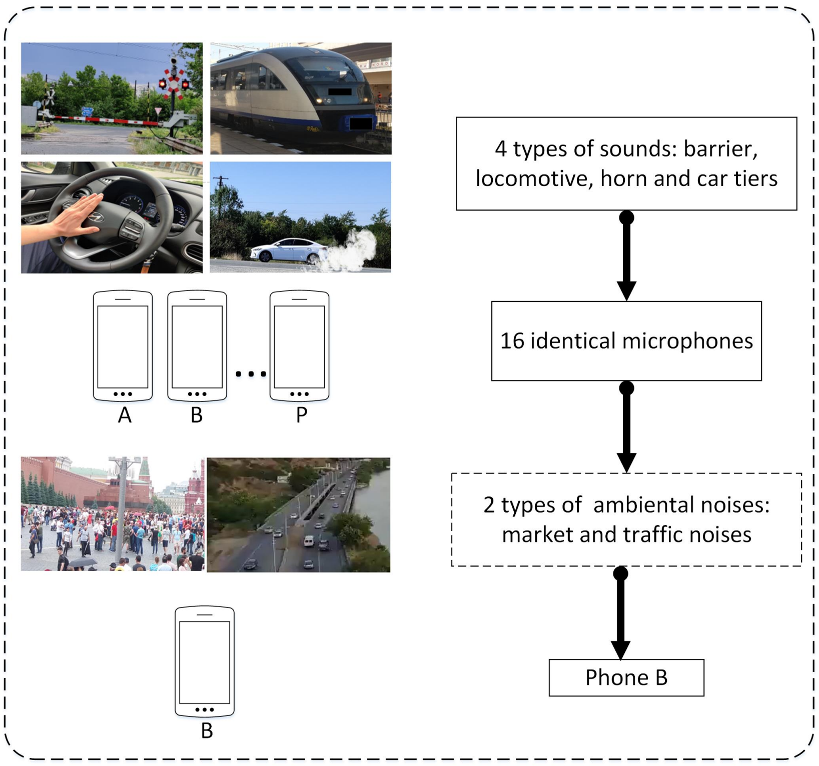

4 types of sounds: barrier, locomotive, horn and car tiers
16 identical microphones
2 types of ambiental noises: market and traffic noises
Phone B
A
B
P
B

Figure 5.13: Overview of the method for smartphone identification based on environmental noise

$\sim$ 87 samples/second and a training time of 18.364 seconds. The mean precision, mean recall and accuracy for various noise levels are shown in Figure 5.12. Identification stops working at SNRs lower than -50dB, increases precision, recall and accuracy from -0.2 to 0.9 at SNRs between -50 and 0dB and ultimately reaches close to 1 at SNRs higher than 0dB.

5.3.2 Fingerprinting identical microphones using environmental noise (indoor experiments)

Again, to get closer to a real-life scenario, the two previous types of noise, i.e., market and traffic, are used at distinct levels. In Figure 5.13 the flowchart of this test scenario is depicted. The dataset contains 16 microphones from the same smartphone model. Four prerecorded ambient noises from the SoundArchive are used: (i) a locomotive indicating departure; (ii) barriers shutting with two bells jingling; (iii) an automobile approaching with screeching tiers; and (iv) the two-tone horn sound of a car. For each microphone, 50 measurements were done while each background sound was playing, totaling 800 mea-

Table 5.3: Precision, recall and accuracy for 5 classifiers

sound type	metrics	LD	ENS	TREE	KNN	SVM
locomotive	precision	1.00	1.00	0.79	0.99	0.99
	recall	1.00	1.00	0.79	0.99	0.99
	accuracy	1.00	1.00	0.91	0.98	0.99
barrier	precision	1.00	1.00	0.89	0.89	0.98
	recall	1.00	1.00	0.90	0.90	0.98
	accuracy	1.00	1.00	0.91	0.92	0.98
car	precision	1.00	1.00	0.79	0.88	0.99
	recall	1.00	1.00	0.82	0.88	0.99
	accuracy	1.00	0.99	0.89	0.83	0.98
horn	precision	1.00	1.00	0.82	0.98	0.99
	recall	1.00	1.00	0.83	0.98	0.99
	accuracy	1.00	1.00	0.90	0.96	1.00

surements for all sounds and 3200 measurements overall. Again, the two prior sources of noise, namely the market and traffic, are added at different intensities to make the simulation more realistic. Figure 5.13 shows the flowchart for this test scenario.

Fingerprinting microphones using environment sounds (clean recordings): For training, 20 measurements were used and for testing, 30 measurements were used. The mean precision, mean recall and accuracy for each classifier for each type of sound are shown in Table 5.3. It is clear that the LD and ENS classifiers, closely followed by the SVM, produce the best results. The KNN produces subpar results and as to be predicted, the TREE classifier again produces the worst results.

	A	B	C	D	E	F	G	H	I	J	K	L	M	N	O	P
LD	0.	0.	0.	0.	0.	0.	0.	0.	0.	0.	0.	0.	0.	0.	0.	0.
ENS	0.	0.	0.	0.	0.	0.	0.	0.	0.	0.	0.	0.	0.	0.	0.	0.
TREE	0.002	0.	0.019	0.	0.042	0.002	0.041	0.045	0.031	0.	0.008	0.006	0.	0.011	0.002	0.004
KNN	0.	0.	0.	0.	0.004	0.	0.	0.	0.	0.	0.	0.	0.	0.	0.	0.
SVM	0.	0.	0.	0.	0.	0.002	0.	0.	0.	0.	0.	0.	0.	0.	0.	0.002

(i) FAR's for 16 identical microphones when locomotive sound was used

	A	B	C	D	E	F	G	H	I	J	K	L	M	N	O	P
LD	0.	0.	0.	0.	0.	0.	0.	0.	0.	0.	0.	0.	0.	0.	0.	0.
ENS	0.	0.	0.	0.	0.	0.	0.	0.	0.	0.	0.	0.	0.	0.	0.	0.
TREE	0.	0.	0.222	0.032	0.620	0.033	0.352	0.357	0.578	0.230	0.133	0.228	0.	0.038	0.383	0.
KNN	0.	0.	0.	0.	0.	0.	0.	0.	0.	0.	0.	0.	0.	0.	0.062	0.
SVM	0.	0.	0.	0.	0.	0.	0.	0.	0.032	0.	0.	0.	0.	0.	0.032	0.

(ii) FRR's for 16 identical microphones when locomotive sound was used

Figure 5.14: FAR (up) and FRR (down) for the LD, ENS, TREE, KNN and SVM classifiers in case of 16 identical microphones when locomotive sound was used

The FAR and FRR for each classifier and microphone on the four testing sounds are shown as numerical values in Figures 5.14, 5.15, 5.16 and 5.17. Figure 5.14 shows the FAR (up) and FRR (down) for the locomotive sound. The overall FAR is quite low for all classifiers, with the TREE classifier for microphone H having the highest value at 4.5%. For the TREE classifier on microphone E, the FRR approaches 62%. The FAR and FRR for all microphones are zero for the LD and ENS classifiers. The FAR (up) and

	A	B	C	D	E	F	G	H	I	J	K	L	M	N	O	P
LD	0.	0.	0.	0.	0.	0.	0.	0.	0.	0.	0.	0.	0.	0.	0.	0.
ENS	0.	0.	0.	0.	0.	0.	0.	0.	0.	0.	0.	0.	0.	0.	0.	0.
TREE	0.002	0.011	0.002	0.008	0.002	0.026	0.013	0.006	0.	0.	0.	0.021	0.002	0.015	0.	0.002
KNN	0.	0.006	0.002	0.002	0.004	0.002	0.013	0.024	0.004	0.	0.006	0.015	0.002	0.008	0.002	0.015
SVM	0.	0.002	0.002	0.004	0.	0.002	0.002	0.	0.002	0.	0.	0.	0.002	0.002	0.	0.

(i) FAR's in case of 16 identical microphones when barrier sound was used

	A	B	C	D	E	F	G	H	I	J	K	L	M	N	O	P
LD	0.	0.	0.	0.	0.	0.	0.	0.	0.	0.	0.	0.	0.	0.	0.	0.
ENS	0.	0.	0.	0.	0.	0.	0.	0.	0.	0.	0.	0.	0.	0.	0.	0.
TREE	0.292	0.038	0.	0.	0.033	0.100	0.040	0.	0.062	0.117	0.062	0.090	0.	0.115	0.361	0.147
KNN	0.	0.	0.	0.	0.151	0.	0.225	0.406	0.066	0.166	0.270	0.	0.	0.037	0.	0.206
SVM	0.	0.033	0.	0.	0.	0.	0.033	0.062	0.033	0.	0.032	0.	0.	0.064	0.	0.032

(ii) FRR's in case of 16 identical microphones when barrier sound was used

Figure 5.15: FAR (up) and FRR (down) for the LD, ENS, TREE, KNN and SVM classifiers in case of 16 identical microphones when barrier sound was used

	A	B	C	D	E	F	G	H	I	J	K	L	M	N	O	P
LD	0.	0.	0.	0.	0.	0.	0.	0.	0.	0.	0.	0.	0.	0.	0.	0.
ENS	0.	0.	0.	0.	0.	0.	0.	0.	0.	0.	0.	0.	0.	0.	0.	0.
TREE	0.	0.019	0.017	0.	0.028	0.004	0.004	0.004	0.011	0.	0.	0.033	0.002	0.008	0.036	0.042
KNN	0.004	0.004	0.013	0.	0.035	0.004	0.006	0.024	0.008	0.	0.011	0.004	0.	0.006	0.	0.002
SVM	0.002	0.	0.	0.	0.002	0.	0.	0.	0.	0.	0.	0.	0.	0.	0.	0.

(i) FAR's in case of 16 identical microphones when car tiers sound was used

	A	B	C	D	E	F	G	H	I	J	K	L	M	N	O	P
LD	0.	0.	0.	0.	0.	0.	0.	0.	0.	0.	0.	0.	0.	0.	0.	0.
ENS	0.	0.	0.	0.	0.	0.	0.	0.	0.	0.	0.	0.	0.	0.	0.	0.
TREE	0.	0.	0.120	0.032	0.190	0.517	0.200	0.243	0.375	0.032	0.210	0.464	0.	0.071	0.187	0.166
KNN	0.066	0.096	0.172	0.	0.416	0.066	0.156	0.486	0.103	0.	0.166	0.066	0.	0.	0.	0.064
SVM	0.	0.032	0.	0.	0.	0.	0.	0.032	0.	0.	0.	0.	0.	0.	0.	0.

(ii) FRR's in case of 16 identical microphones when car tiers sound was used

Figure 5.16: FAR (up) and FRR (down) for the LD, ENS, TREE, KNN and SVM classifiers in case of 16 identical microphones when car tiers sound was used

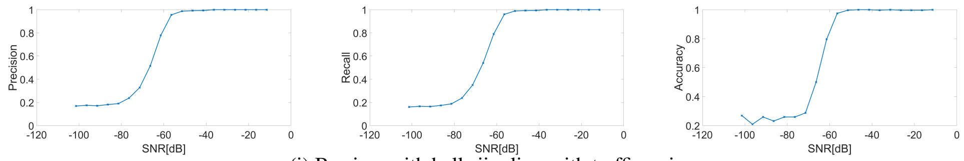
	A	B	C	D	E	F	G	H	I	J	K	L	M	N	O	P
LD	0.	0.	0.	0.	0.	0.	0.	0.	0.	0.	0.	0.	0.	0.	0.	0.
ENS	0.	0.	0.	0.	0.	0.	0.	0.	0.	0.	0.	0.	0.	0.	0.	0.
TREE	0.002	0.013	0.040	0.002	0.	0.009	0.015	0.022	0.006	0.013	0.004	0.021	0.002	0.004	0.011	0.013
KNN	0.	0.	0.	0.	0.	0.	0.	0.	0.	0.	0.	0.004	0.	0.004	0.002	0.002
SVM	0.	0.	0.	0.	0.	0.	0.	0.	0.	0.	0.	0.	0.004	0.002	0.002	0.

(i) FAR's in case of 16 identical microphones when car horn sound was used

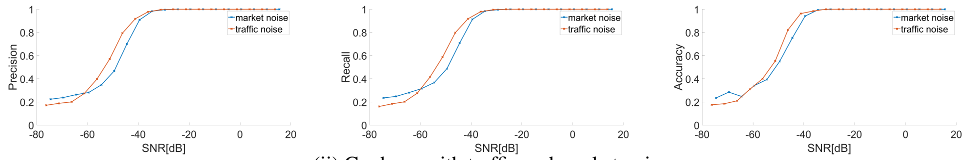
	A	B	C	D	E	F	G	H	I	J	K	L	M	N	O	P
LD	0.	0.	0.	0.	0.	0.	0.	0.	0.	0.	0.	0.	0.	0.	0.	0.
ENS	0.	0.	0.	0.	0.	0.	0.	0.	0.	0.	0.	0.	0.	0.	0.	0.
TREE	0.033	0.076	0.266	0.	0.032	0.458	0.233	0.310	0.205	0.294	0.096	0.166	0.	0.096	0.074	0.250
KNN	0.	0.	0.	0.	0.	0.	0.	0.	0.	0.062	0.032	0.066	0.	0.034	0.	0.
SVM	0.	0.	0.	0.062	0.	0.	0.032	0.	0.	0.032	0.	0.	0.	0.	0.	0.

(ii) FRR's in case of 16 identical microphones when car horn sound was used

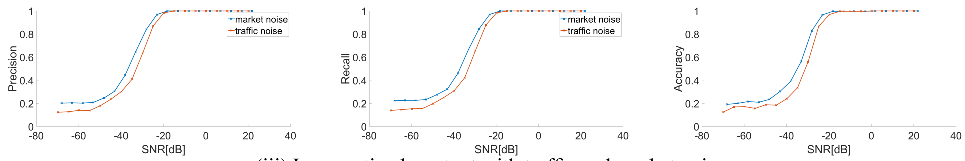
Figure 5.17: FAR (up) and FRR (down) for the LD, ENS, TREE, KNN and SVM classifiers in cased of 16 identical microphones when car horn sound was used



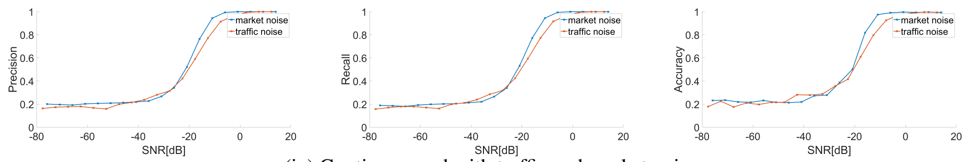
(i) Barriers with bells jingling with traffic noise



(ii) Car horn with traffic and market noise



(iii) Locomotive long toot with traffic and market noise



(iv) Car tiers sound with traffic and market noise

Figure 5.18: Precision (left), recall (middle) and accuracy (right) resulted with linear discriminant classifier with noise between -80dB and 20dB with 5dB increment

FRR (down) for the barrier sound are illustrated in Figure 5.15. Once more, the overall FAR is quite low for all classifiers. The TREE classifier on microphone F achieves the highest value of 2.6%. For the TREE classifier on microphone O, the FRR approaches 36%. Once more, the FAR and FRR for all microphones are zero for the LD and ENS classifiers. The FAR (up) and FRR (down) for the vehicle tiers sound are shown in Figure5.16. With a maximum value of 4.2% for the TREE classifier on microphone P, the FAR is extremely low for all classifiers. For the KNN classifier on microphone H, the FRR reaches 48%. The FAR and FRR are once again zero for all microphones for the LD and ENS classifiers. The FAR (up) and FRR (down) for the car horn sound are shown in Figure 5.17. The TREE classifier on microphone C has a maximum FAR value of 4% and microphone F has a maximum FRR value of 45%. The FAR and FRR are once again zero for all microphones for the LD and ENS classifiers.

Overall, from these data, the FAR and FRR for all microphones are close to zero for the LD and ENS classifiers. The barrier and horn sounds produced worse identification rates for the other classifiers than the locomotive and vehicle tier sounds, with the greatest values for the FAR and FRR.

Fingerprinting microphones using environment sounds with ambient noise: Since ambient noise is present in real-world situations, two types of noise (traffic and market noise) are added to the clean signals at different SNRs, i.e., from -80dB to 20dB, with a 5dB increment step. Only the LD classifier is used in this situation because it produces the best results and has a quick prediction speed. The mean precision (left), mean recall (middle) and accuracy (right) for various noise levels are shown in Figure 5.18. The noise intensity significantly influences the classification of the four sound types. For the barrier sound, identification is ineffective at SNRs lower than -70 dB, increases in precision, recall and accuracy from -0.2 to 0.9 at SNRs between -70 and -63 dB and approaches 1 at SNRs higher than -63 dB. For the horn sound, identification is ineffective at SNRs lower than -60dB, increases in precision, recall and accuracy from -0.2 to 0.9 at SNRs between -60 and -40dB and approaches 1 at SNRs higher than -40dB. The impact of loudness is comparable for locomotive sounds and horn sounds. For the vehicle tiers sound, identification is ineffective at SNRs lower than -35dB, increases in precision, recall and accuracy from -0.2 to 0.9 at SNRs between -35 and -8dB and approaches 1 at SNRs greater than -8dB.

The impact of vehicle tires on microphone identification is the least, but horns and locomotive sounds have a greater impact on fingerprints.

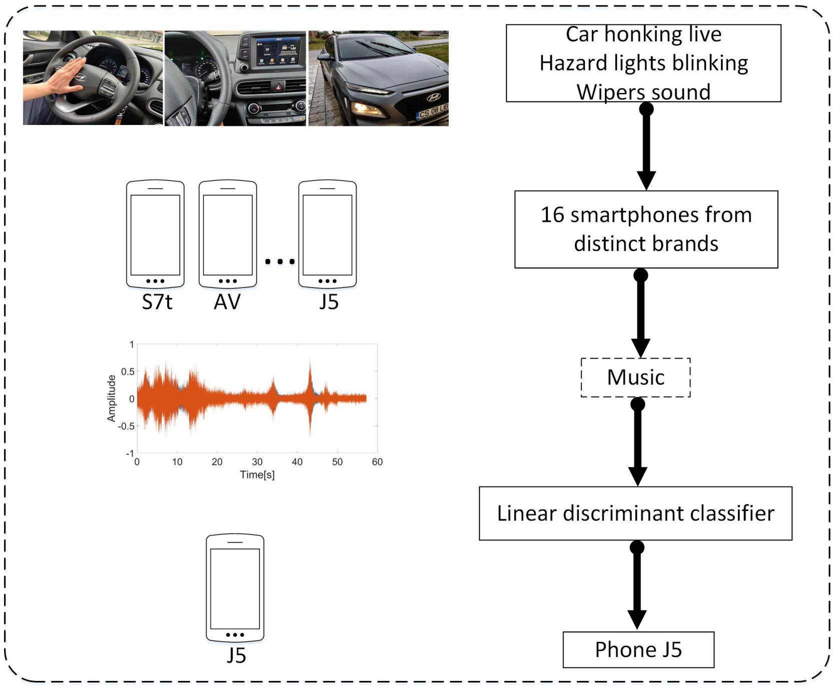


Figure 5.19: Smartphone identification based on live recordings

5.4 Fingerprinting microphones using live recordings

This section examines microphone fingerprinting in the more difficult live recording scenario. Only the LD algorithm is used in this case since it gave the most satisfactory results in all the previous tests.

	A21s	J5	J5'	J5"	LE	LE'	S6	S6'	OP	S7t	AV	LG	A3	S7	N7	MT
LDhorn	0.001	0.001	0.001	0.001	0.001	0.001	0.001	0.	0.001	0.001	0.001	0.001	0.001	0.	0.001	0.001
LDhazard	0.	0.	0.	0.	0.	0.	0.	0.	0.	0.	0.	0.	0.	0.	0.	0.
LDwipers	0.	0.	0.	0.001	0.	0.	0.	0.	0.	0.	0.	0.	0.	0.	0.	0.

(i) FAR's in case of 16 microphones when horn, hazard lights and wipers sounds were used

	A21s	J5	J5'	J5"	LE	LE'	S6	S6'	OP	S7t	AV	LG	A3	S7	N7	MT
LDhorn	0.	0.016	0.016	0.	0.	0.	0.	0.	0.033	0.	0.027	0.055	0.005	0.	0.016	0.
LDhazard	0.	0.	0.	0.	0.	0.	0.	0.	0.	0.	0.	0.	0.	0.	0.	0.
LDwipers	0.	0.	0.	0.	0.	0.	0.	0.	0.	0.	0.006	0.	0.	0.	0.	0.

(ii) FRR's in case of 16 microphones when horn, hazard lights and wipers sounds where used

Figure 5.20: FAR (up) and FRR (down) for the linear discriminant classifier for 16 microphones when horn, hazard lights and wipers sounds where used

	A21s	J5	J5'	J5"	LE	LE'	S6	S6'	OP	S7t	AV	LG	A3	S7	N7	MT
LDhorn	0.002	0.010	0.008	0.004	0.017	0.008	0.013	0.006	0.006	0.011	0.020	0.010	0.008	0.009	0.008	0.020
LDhazard	0.001	0.001	0.006	0.001	0.001	0.001	0.003	0.007	0.002	0.003	0.001	0.001	0.	0.006	0.001	0.001
LDwipers	0.001	0.009	0.007	0.	0.003	0.003	0.006	0.014	0.001	0.007	0.	0.004	0.006	0.020	0.001	0.002

(i) FAR's in case of 16 microphones when horn, hazard lights and wipers sounds were affected by music

	A21s	J5	J5'	J5"	LE	LE'	S6	S6'	OP	S7t	AV	LG	A3	S7	N7	MT
LDhorn	0.016	0.155	0.127	0.077	0.066	0.116	0.100	0.100	0.266	0.083	0.177	0.516	0.083	0.072	0.511	0.061
LDhazard	0.010	0.025	0.050	0.055	0.010	0.005	0.070	0.095	0.055	0.009	0.	0.005	0.005	0.080	0.065	0.
LDwipers	0.	0.113	0.113	0.	0.040	0.040	0.133	0.140	0.053	0.122	0.006	0.066	0.226	0.106	0.053	0.073

(ii) FRR's in case of 16 microphones when horn, hazard lights and wipers sounds were affected by music

Figure 5.21: FARs (up) and FRRs (down) for the linear discriminant classifier for 16 microphones when horn, hazard lights and wipers sounds were affected by music

Scenario C is now analyzed, using 16 smartphones to perform outdoor recordings. For each smartphone, 400 measurements of a vehicle honking, 300 measurements of a vehicle's warning lights blinking sound and 300 measurements of a vehicle's low-speed wiper sound are taken, for a total of 4800 measurements. The power spectrum for each signal is extracted and used for the LD classifier. The input for the classifier are the 4096 features for each audio signal because the power spectrum is an array with 4096 elements. For training, 55% of the measurements are randomly selected, while the remaining 45% of the measurements are used for testing. The sound is amplified with background music noise to further complicate identification. For this, the first three songs from the Top 10 Spotify chart for 2021 are used. This scenario is shown in Figure 5.19.

Results on clean recordings: The FAR (up) and FRR (down) for the LD classifier for 16 microphones, which record horn, hazard lights and wiper sounds, are shown in Figure 5.20. For the horn sound, the FAR and FRR are very low, i.e., the FAR is below 0.1% and the FRR is below 5.5%. In the case of hazard lights and sounds, the FAR and FRR is zero. For wiper sound, the FAR and FRR are both zero, except for smartphones J5" which have a FAR 0.1% and AV, which has an FRR of 0.6%.

Results on recordings affected by noise: The FAR (up) and FRR (down) for the LD classifier for 16 microphones that record horn, hazard lights and wiper sounds influenced by background music are shown in Figure 5.21. For vehicle horn sounds affected by music, the FAR is between 0.2% and 2 and the FRR is between 1.6% and 51.6%. For hazard lights sounds influenced by music, the FAR is below 0.6 and the FRR is below 9.5%. For wipers sounds influenced by music, the FAR is below 2% and the FRR is below 22%.

5.5 Concluding remarks

Using the power spectrum of the recorded signal and various supervised machine learning methods, such as Linear Discriminant, Ensemble-Subspace Discriminant, Fine Tree, Fine KNN and Linear SVM, smartphone microphone fingerprinting was investigated in this chapter. Three fingerprinting use cases based on recorded human speech, artificially generated background sound and live recordings were analyzed. The LD classifier behaved nearly flawlessly in the first two cases. Extra noise was added to each situation to make identification more difficult. The final scenario was the more challenging. When noise was added, the LD produces poor identification results for two specific phones (the LG and Nexus 7). The LD classifier may still be preferred because it utilizes little memory and has a short runtime. Other conventional machine learning classifiers performed worse than LD in terms of accuracy. This type of fingerprints have a wide range of potential uses. For example, verifying ownership of a specific phone to serve as a second authentication token with physical characteristics that cannot be cloned. Nonetheless, this kind of fingerprinting may also be abused by applications that fingerprint devices endangering the privacy of users.

Chapter 6

Fingerprinting smartphones based on camera sensors

This chapter is based on the results of the author from [23]. This work addresses smartphones fingerprinting based on camera sensors and experiments with six identical smartphones trying to identify them based on data extracted from jpeg images.

6.1 Camera-based fingerprinting, brief motivation

Nowadays, the digital camera is an indispensable component of all mobile devices. The use of camera-collected images is a practical method of identifying smartphones. Designing Physical Unclonable Functions (PUF), which for a collection of inputs (considered as challenges), provide a device-specific response based on special and unpredictable circuit differences resulting from the manufacturing process, is one example of a use case. These differences take into consideration the distinct sensor properties brought on by the physical, chemical and geometrical abnormalities in sensors. The authors in [2] established two decades ago the idea of circuit identification based on particular information extracted from the manufacturing process's irregularity and the PUFs emerged slightly later [3]. Numerous PUFs have been developed since then. There are two different kinds of PUFs depending on the quantity of challenge-response pairs (CRPs): strong PUFs (a huge number of CRPs) and weak PUFs (with a small number of CRPs), including memory-based PUFs. The PUFs are frequently employed in systems for device/sensor fingerprinting and device authentication.

Relating to camera sensors, fixed-pattern noise (FPN) is a persistent pixel variation that manifests itself in photos acquired under uniform lighting circumstances as the same pattern of darker and brighter pixels. The FPN can be calculated using the formula $FPN = X - X_{fil}$, where X represents the original picture and X_{fil} represents the filtered image. This process depends on numerous manufacturing flaws that are particular to

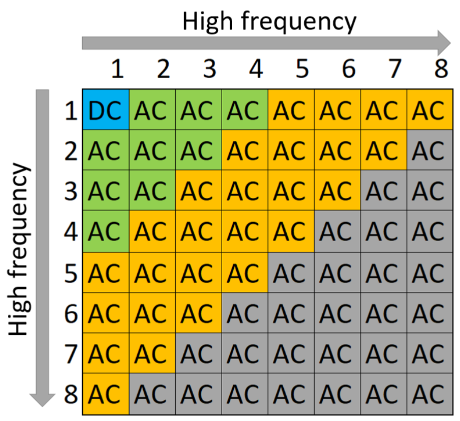
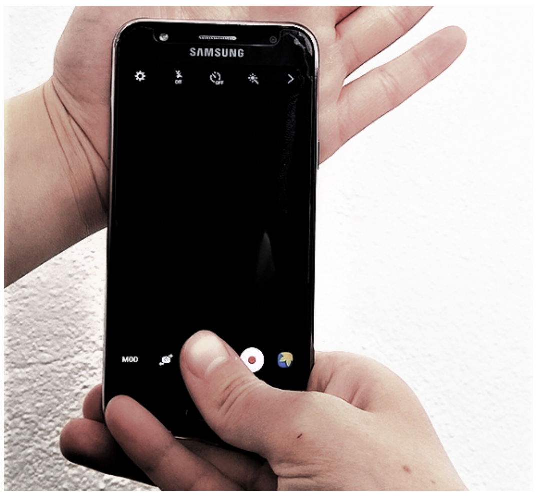

High frequency
1 2 3 4 5 6 7 8
High frequency
1 DC AC AC AC AC AC AC AC
2 AC AC AC AC AC AC AC AC
3 AC AC AC AC AC AC AC AC
4 AC AC AC AC AC AC AC AC
5 AC AC AC AC AC AC AC AC
6 AC AC AC AC AC AC AC AC
7 AC AC AC AC AC AC AC AC
8 AC AC AC AC AC AC AC AC

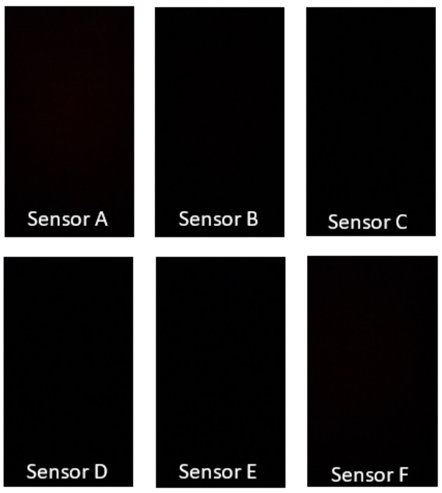
Figure 6.1: DCT coefficients

each sensor in order to remove noise. Additionally, because it possesses features that are particular to a specific device, this noise may be utilized to create a PUF and, as demonstrated in numerous publications like [88], [90] and [91], can also be used as an input to a cryptographically secure function. Dark signal non-uniformity (DSNU), which accounts for differences in pixel offsets in the absence of illumination (dark frame) and photo response non-uniformity (PRNU), which is a fluctuation between pixels in the presence of lighting, are the two types of FPNs (light frames). This study only considers DSNU, which provides greater accuracy without sacrificing usability. The discrete cosine transform (DCT) converts picture pixels from the spatial domain to the frequency domain. The JPEG picture compression technique frequently uses DCT. During JPEG compression, the two-dimensional DCT is applied to the 8x8 non-overleaping chunks of the picture. The result is 64 DCT coefficients for each 8x8 block, the first coefficient coming from the upper left corner is the DC coefficient and the following 63 being the AC coefficients [72]. In Figure 6.1, the DC coefficient is shown in blue, the low-frequency AC coefficients are shown in green, the mid-frequency AC coefficients are marked in orange and the high-frequency AC coefficients are marked in gray. These coefficients correspond to the DCT of an 8x8 image block, the block size for encoding is specified in the ISO/IEC 10918-1:1994 standard for digital compression and coding [72]. Only the low and mid-frequency AC coefficients are used in this analysis since the high-frequency coefficients are affected by JPEG compression and are vulnerable to distortions. In addition, low frequencies are more perceptible to human eyes than high frequencies.

In this chapter, a variety of conventional machine learning algorithms are used, including Nearest Neighbor (KNN), Ensemble-Subspace Discriminant (ENS), Support Vector Machines (SVM), Linear Discriminant (LD), Naive Bayes (NB) and a wide neural network (WNN), to analyze such frequency domain representations from data collected on six identical CMOS sensors. Since a case with identical sensors seems to be more



(i) Experimental use case



(ii) Image produced by each sensor

Figure 6.2: Experimental use case (i) and one image captured by each camera sensor (ii)

demanding, this analysis is concentrated on six identical camera sensors connected to the same smartphone.

6.2 Setup and methodology

A quick summary of the setup and technique is provided next. The experimental use case is shown in Figure 6.2 (i), which depicts a person shooting a dark photo while holding the phone against their palm to fingerprint the sensor. Users may easily acquire this data. Figure 6.2 (ii) illustrates six dark photos taken by the six cameras. Using this method, 50 dark photographs are taken with each camera. The tests were done at approximately 22° Celsius.

A Samsung Galaxy J5 smartphone with 13 MP sensors, f1.9, a 28mm (wide) lens and an AF-capable camera is used in this fingerprinting situation. The disassembled Samsung Galaxy J5 for camera replacement is shown in Figure 6.3. In order to prevent faults caused by the rest of the electronics in the original smartphone and acquire an exact measurement for the defects in each sensor, six identical camera circuits are added to it (these are connected to the same Samsung Galaxy J5 smartphone). Indeed, having different smartphones will also contribute to these variations, but the goal was to have an accurate result regarding the imperfections of each sensor, excluding the rest of the circuitry from the smartphone (which remains identical). The Samsung Galaxy J5 and the six cameras are displayed in Figure 6.4. Using Matlab R2020b, the photos are processed and analyzed. A computer with an Intel(R) Core(TM) i7-6700 CPU running at 3.40 GHz and 32 GB of RAM is used to conduct the analysis.

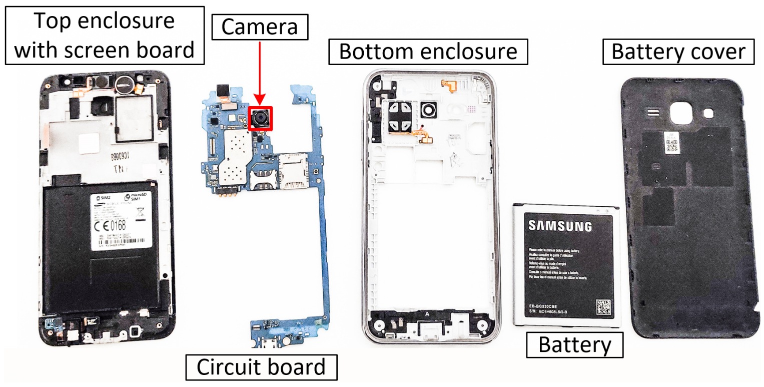


Figure 6.3: Samsung Galaxy J5 dissembled

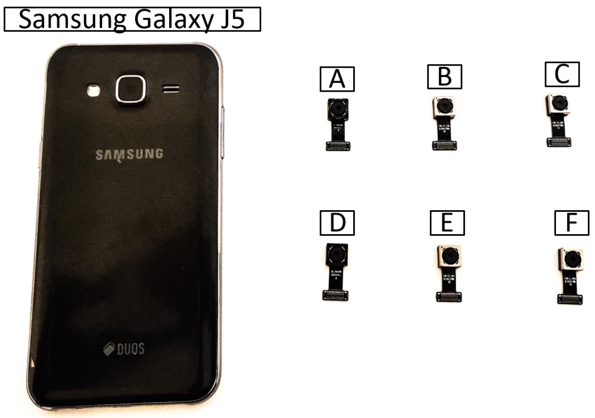


Figure 6.4: Samsung Galaxy J5 and the six cameras that have been disassembled

The focus of this analysis is on sections of the DCT applied to the complete image, eliminating any cropping or scaling that would affect the fingerprint. When only the blue channel is taken into account, as will be shown in a later paragraph, the results are better. Additionally, this reduced the computing time by employing a single channel. The 2-D adaptive noise-removal filter from Matlab called the *wiener2* filter was employed to process the original image. With 10x10 local neighborhoods, this filter calculates the variance and the local mean surrounding each pixel. To recover the pixel variations, the residual noise was computed as the difference between the original picture and the filtered image. The residual noise is divided into 8x8 non-overleaping blocks. The 2-D DCT is computed for each block and the low and mid-frequency AC coefficients are extracted. Each 8x8 block is converted into an array of 35 elements using the zigzag sequence, which is then concatenated to produce the fingerprint. This fingerprint extraction procedure comprises the following seven phases, shown in Figure 6.5: image acquisition

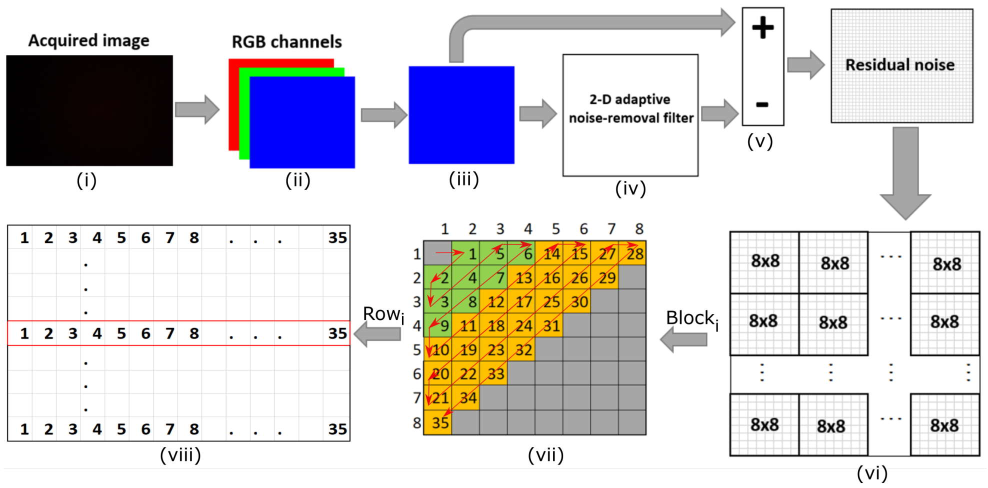


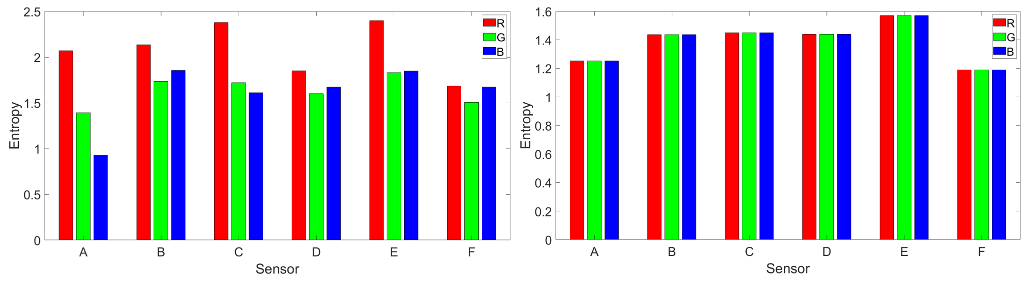
Figure 6.5: Process of extracting fingerprints: from picture capture to AC coefficients

(i), RGB channel separation (ii), blue channel extraction (iii), wiener2 filter application (iv), residual noise computing as the difference between the blue channel image and the filtered image (v), 8x8 block division of the residual noise (vi), 2-D DTC application on each 8x8 block and extraction of the low and mid AC coefficients (vii) are all steps in the process.

The entropy of the data is computed before and after processing the data. The data processing steps are described in Figure 6.5. Both the Shannon entropy and the minimum entropy [220] are employed as metrics, with the latter being a more effective security metric (for applications that plan to use CMOS data as a PUF for authentication) if an adversary merely attempts to guess the data acquired by the sensor using the most frequently used value of the coefficients. For each byte, the probability of occurrence in the array, $p_i, i = 0..255$, is used to calculate the former using the relation $\sum_{i=0,255} -p_i \log p_i$ and the latter as $-\log\max(p_i)$. The array represents either the bytes of the AC coefficients or the RGB bytes in the original image.

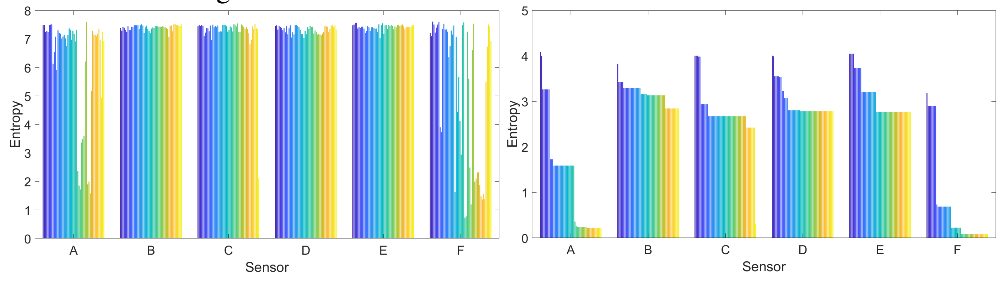
In Figure 6.6, the Shannon (i) and minimum (ii) entropies are shown, calculated from the image's red, green and blue channels (without any processing). The entropy values for the red channel are somewhat more significant than those for the green and blue channels. On the other hand, the blue channel has the lowest entropy, making it a more robust choice for the classification process since it will produce more consistent results. The Shannon entropy for the red channel has a mean value of 2.0855 and a median value of 2.1133. The mean and median values for the green channel are 1.6310 and 1.6354, respectively, while these values are 1.5982 and 1.6024 for the blue channel. For the minimum entropy, the numbers for the three channels are comparable, implying an equivalent minimum security level.

The Shannon and minimum entropy values are computed on the retrieved AC coefficients. In the case of the Shannon entropy, the values are greater than before, often reaching approximately 7 bits for each byte. The Shannon entropy and the minimum entropy per coefficient for 100 randomly chosen rows are shown in Figure 6.7. The entropy is calculated on a 2800-element matrix. It is challenging to capture 100% identical dark pictures when pushing the phone against the palm, so environmental variables are most likely to blame for the entropy fluctuations for sensors A and F. For a matrix of 2800 values, the security level is good enough since the lowest entropy is still often in the region of 2-3 bits for each byte from the coefficients, which is twice as much as in the case of the unprocessed photos.



(i) Mean value of the Shannon entropy (ii) Mean value of the minimum entropy

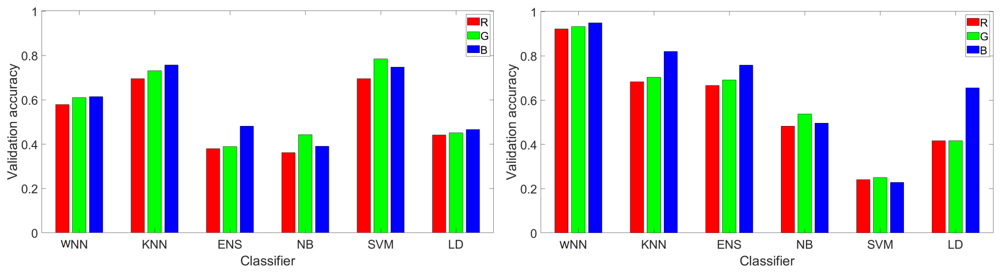
Figure 6.6: Shannon (i) and minimum (ii) entropy calculated on the red, green and blue channels of the image



(i) Shannon entropy for 100 rows (ii) Minimum entropy for 100 rows

Figure 6.7: Shannon (i) and minimum (ii) entropy when 100 rows were randomly selected

The blue channel was chosen instead of the green channel, which previous works have often employed for camera identification, because this early investigation revealed that this channel produces superior results. In Figure 6.8, bar charts of the validation accuracy for 100 and 1000 randomly chosen rows for all classifiers and all three channels are shown as evidence for this claim. Each channel is identified using the red, green and blue colors that go with it. The data was split into 80% for training and the remaining 20% for testing. The blue channel performs best with NN, KNN, ENS and LD, whereas the green channel performs best with NB and SVM. For 1000 samples, NB and SVM produce



(i) Validation accuracy for 100 rows (ii) Validation accuracy for 1000 rows

Figure 6.8: Validation accuracy for 100 (i) and 1000 (ii) randomly selected rows for all classifiers and all channels

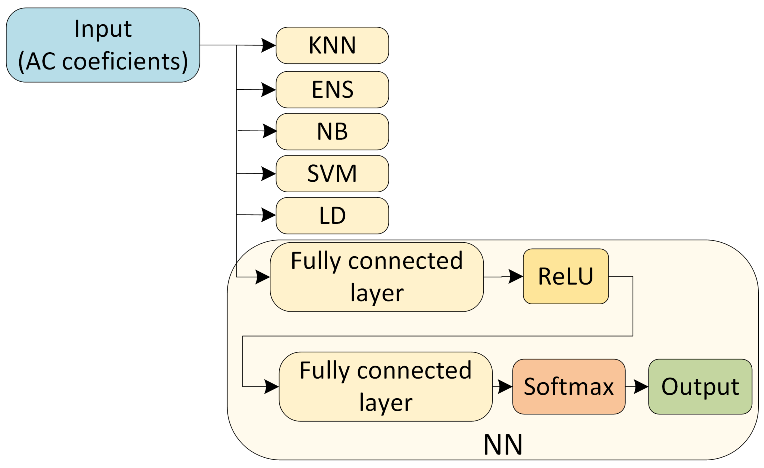


Figure 6.9: The multi-layer fully-connected neural network and the classifiers use the AC coefficients as input

the lowest results for all channels, making them unsuitable as classification algorithms for this task.

Following image processing, a bi-dimensional array of 149640 rows and 35 columns is generated for each image. Each of the 8x8 matrices created for each picture corresponds to one of the 149640 rows and the number of retrieved AC coefficients is 35. Due to two factors prediction time and memory requirements, the classification based on the whole array is impractical because of the size of the output array produced after processing (out-of-bounds errors may result from several classifiers). Samples of 100 or 1000 rows are used for each image and device. Among the 149640 rows, the 100 rows (or 1000 rows in the second scenario) were randomly chosen, although the selection process remained constant across all of the experiment's photos. Consequently, each picture is converted into a bi-dimensional array of 100 or 1000 rows and 35 columns to do the classification. Also, 10,000 rows are attempted to be utilized, but the increases in accuracy are not significant and the size of the dataset caused out-of-memory errors for several classifiers, including LD, SVM and NB, while significantly increasing the classification time for others. Another approach involved taking 100 or 1000 element matrices from the top left corner of each image, although the results are less successful. The random selection of the matrix, which was previously used, seems to produce the most relevant results. As input, 100 (or 1000) rows and 35 columns are used for each of the classification algorithms, i.e., WNN, KNN, ENS, NB, SVM and LD, along with the low and mid AC coefficients acquired after using the 2-D DCT on the 8x8 blocks of residual noise retrieved from the photos. The inputs of the classification methods are shown in Figure 6.9.

6.3 Identification of CMOS sensors using machine learning

The identification of CMOS sensors using a variety of classifiers, including linear discriminant (LD), Naive Bayes (NB), Support Vector Machine (SVM), Nearest Neighbor (KNN), Ensemble-Subspace Discriminant (ENS) and a multi-layer fully-connected neural network (NN) is discussed in this section.

Seven different sizes of training sets are used for each classifier, starting with 20% of the images for training (the remaining 80% are used for testing), increasing the training portion of the image usage until it reaches 80% and then decreasing the testing portion of the image usage until it reaches 20%. The increment step used for this was 10%.

6.3.1 Results for 100 rows from each image

The validation accuracy for each of the six classifiers and all test situations, with 100 randomly chosen rows for each image, is shown in Figure 6.10. The values from this figure are given as average values for each classifier for all the sensors. As anticipated, each classifier's validation accuracy increases with the training data percentage. Following SVM in validation accuracy for all training percentages are KNN and WNN. With NB, the poorest validation accuracy was attained. Since the results are inconsistent, the precision and recall are also discussed for each sensor.

Figures 6.11, 6.12, 6.13, 6.14, 6.15 and 6.16 show the accuracy (up) and recall (down) for each CMOS sensor for all the investigated scenarios and all classifiers in the case of 100 randomly chosen matrices as 3D-bar charts and numerical values for a more precise representation.

WNN: Figure 6.11 shows the precision (up) and recall (down) for 6 CMOS sensors for the WNN classifier for 100 randomly selected matrices. The precision values are often below 50% for sensors with training percentages below 60%, whereas sensors with training percentages above 80% have precision values as low as 60% for sensor C, 80%

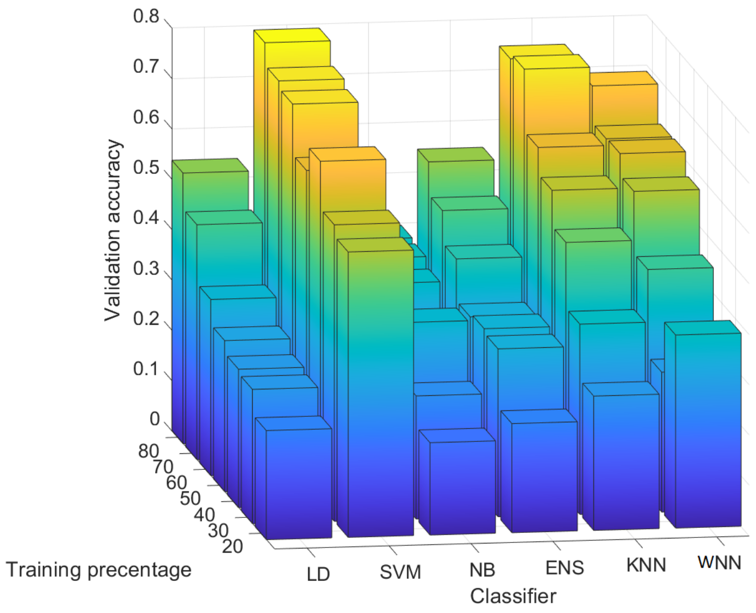


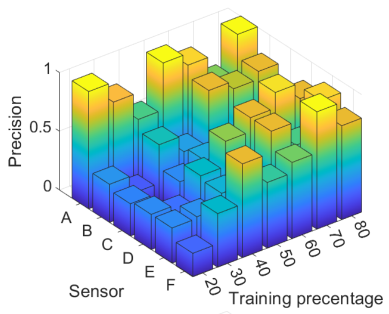
Figure 6.10: Validation accuracy when 100 randomly selected rows were used

for sensors B, D and F, 90% for sensor E and 100% for sensor A. Sensor D achieved a 100% recall rate independent of the training percentage. Additionally, sensor E's recall is almost 100% for all training percentages. The sensors with the lowest recall rates were A and F, but the results generally improved with increasing training percentages in this case, reaching a maximum recall of 66% for A and 61% for F at 80% training.

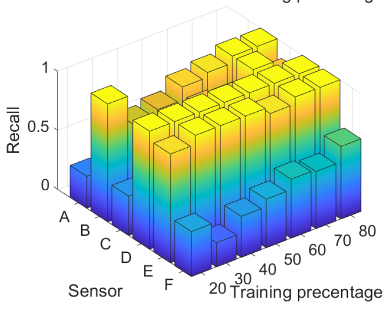
KNN: Figure 6.12 shows that the precision for KNN is comparable to the results from WNN. Precision results are poor for training percentages below 60%, while for training percentages above 70%, the lowest precision rises to 66% for sensor C. Regardless of the training percentage, sensor B has a recall of about 100%. While sensors A and F have recalls below 70% for all training percentages, sensors C, D and E have recalls close to 100% for all training percentages.

ENS: The precision and recall are lower for all evaluated training percentages than for KNN and WNN, as shown in Figure 6.13. Since the result was not a real number due to a division by zero, the recall value in the case of 20% training for sensor A is marked with `NaN`. (there were no true positives and false negatives).

SVM: Figure 6.14 shows that for all training percentages, the precision is highest for sensors A, B, C, D and E, often nearing 100%. However, sensor F reaches its maximum value of 30% at training percentages of 40%, which is not satisfactory. For sensors A, B, C and D, the recall is above 53% for all training percentages, demonstrating an average performance. The recall for sensor E is approximately 60% to 70% for larger training set sizes, which is once again average, while the recall for sensor F is between 0% and 66%, which is poor.

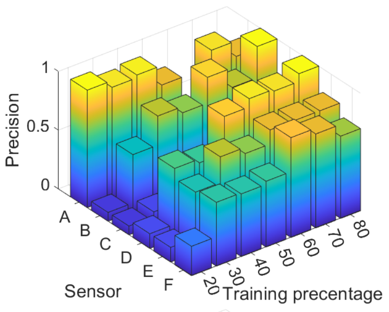


x	A	B	C	D	E	F
20	1.0	0.3250	0.2750	0.3000	0.3000	0.2000
30	0.8285	0.1714	0.0285	0.2285	0.3142	0.4571
40	0.6000	0.5000	0.3000	0.4666	0.4333	0.8000
50	1.0	0.3600	0.4000	0.6800	0.3200	0.5600
60	0.9000	0.8000	0.4500	0.8000	0.8000	0.6500
70	0.6666	0.7333	0.4000	0.9333	0.6000	1.0
80	1.0	0.8000	0.6000	0.8000	0.9000	0.8000

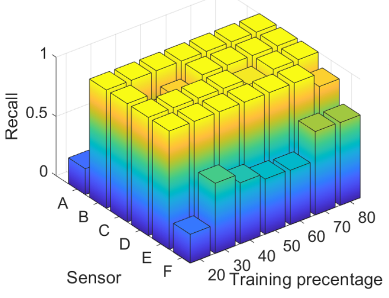


x	A	B	C	D	E	F
20	0.2702	1.0	0.3333	1.0	0.9230	0.3809
30	0.2815	0.7500	0.5000	1.0	1.0	0.2051
40	0.3529	0.7894	0.6428	1.0	1.0	0.3478
50	0.3623	0.9000	0.8333	1.0	1.0	0.4117
60	0.5454	0.9411	0.8181	1.0	0.9411	0.5000
70	0.5263	1.0	1.0	1.0	1.0	0.4838
80	0.6666	1.0	0.8571	1.0	1.0	0.6153

Figure 6.11: Precision (up) and recall (down) for 6 CMOS sensors for WNN classifier when 100 randomly selected matrices were used

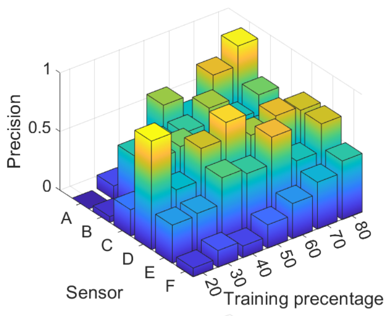


x	A	B	C	D	E	F
20	1.0	0.0750	0.0750	0.1250	0.1205	0.2500
30	0.9428	0.4857	0.1142	0.6000	0.5142	0.5428
40	0.9666	0.7333	0.2666	0.5666	0.7333	0.5333
50	0.8000	0.6800	0.2000	0.8800	0.6800	0.5600
60	0.3500	0.9000	0.4000	0.9500	0.8000	0.8500
70	0.9333	0.7333	0.6666	0.9333	0.8000	0.8000
80	0.8000	1.0	0.5000	1.0	0.8000	0.7000

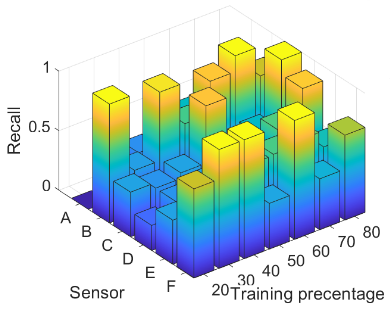


x	A	B	C	D	E	F
20	0.2185	1.0	1.0	1.0	1.0	0.2439
30	0.2820	1.0	0.8000	1.0	1.0	0.5937
40	0.3670	1.0	0.8888	1.0	1.0	0.5161
50	0.3389	1.0	1.0	1.0	1.0	0.4666
60	0.3333	1.0	1.0	0.9500	1.0	0.4594
70	0.5384	1.0	1.0	1.0	1.0	0.7058
80	0.5000	1.0	1.0	1.0	0.8888	0.7000

Figure 6.12: Precision (up) and recall (down) for 6 CMOS sensors for KNN classifier when 100 randomly selected matrices were used

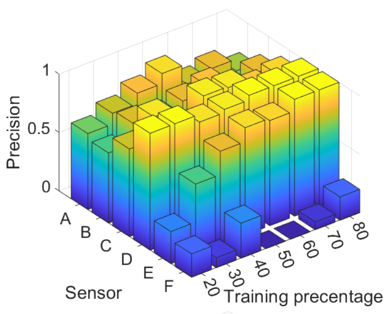


x	A	B	C	D	E	F
20	0.	0.0500	0.2250	0.9250	0.3250	0.0750
30	0.1142	0.4857	0.5714	0.4285	0.3428	0.1428
40	0.1666	0.5000	0.6000	0.7000	0.5666	0.1000
50	0.6400	0.5200	0.2400	0.8400	0.5200	0.2000
60	0.4500	0.6500	0.4500	0.6500	0.7500	0.3000
70	0.7333	0.4666	0.4666	0.7333	0.4666	0.4000
80	0.9000	0.6000	0.5000	0.7000	0.7000	0.5000

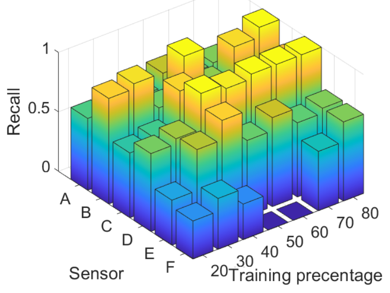


x	A	B	C	D	E	F
20	NaN	1.0	0.3750	0.2114	0.3714	0.7500
30	0.1538	0.3617	0.3278	0.3260	0.4800	1.0
40	0.3125	0.9375	0.3750	0.3818	0.4047	1.0
50	0.4102	0.5909	0.8571	0.4468	0.5909	0.3846
60	0.4736	0.8666	0.4090	0.3939	0.6000	1.0
70	0.4400	1.0	0.4666	0.6111	0.6363	0.4285
80	0.5000	0.7500	1.0	0.8750	0.5000	0.7142

Figure 6.13: Precision (up) and recall (down) for 6 CMOS sensors for ENS classifier when 100 randomly selected matrices were used

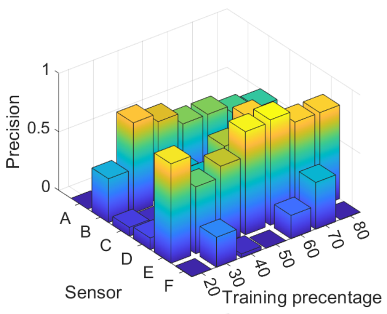


x	A	B	C	D	E	F
20	0.6500	0.6000	0.7500	1.0	0.2750	0.2000
30	0.7428	0.7714	0.9428	1.0	0.6000	0.0857
40	0.8333	0.7333	0.6333	0.9333	0.8000	0.3000
50	0.9200	0.8000	0.9600	0.9600	0.9200	0.
60	0.7500	0.8500	0.9500	1.0	0.9000	0.
70	0.8000	0.7333	0.8666	1.0	1.0	0.0666
80	0.7000	0.8000	0.9000	1.0	1.0	0.2000

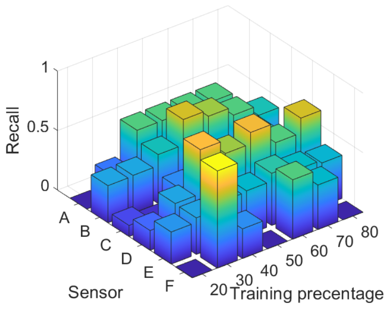


x	A	B	C	D	E	F
20	0.6046	0.8888	0.5454	0.6557	0.3793	0.3200
30	0.6666	0.9310	0.5892	0.7000	0.7241	0.4285
40	0.6250	0.6470	0.9047	1.0	0.8888	0.3000
50	0.7419	1.0	0.6315	0.9600	0.6388	NaN
60	0.6521	0.6800	0.7037	1.0	0.7500	0.
70	0.6666	0.9166	0.6842	1.0	0.6250	0.5000
80	0.5384	1.0	0.8181	1.0	0.6666	0.6666

Figure 6.14: Precision (up) and recall (down) for 6 CMOS sensors for SVM classifier when 100 randomly selected matrices were used



x	A	B	C	D	E	F
20	0.	0.3750	0.0750	0.1000	0.8500	0.
30	0.0285	0.7714	0.0285	0.1428	0.5714	0.2571
40	0.0333	0.7000	0.4000	0.4333	0.6666	0.0333
50	0.0800	0.6000	0.1600	0.6400	0.8800	0.
60	0.0500	0.6000	0.2500	0.8000	0.9000	0.2000
70	0.0666	0.5333	0.3333	0.4666	0.8000	0.4000
80	0.	0.5000	0.4000	0.5000	0.8000	0.



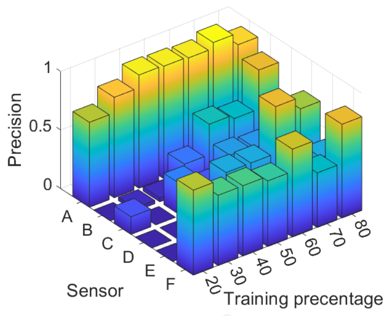
x	A	B	C	D	E	F
20	0	0.3125	0.0937	0.1666	0.2698	NaN
30	0.2000	0.3176	0.0357	0.3125	0.3076	0.8181
40	0.5000	0.4117	0.2449	0.6842	0.3636	0.2500
50	0.5000	0.6250	0.1142	0.5714	0.3728	NaN
60	0.5000	0.5714	0.2083	0.6666	0.4390	0.5000
70	0.5000	0.5000	0.4545	0.5000	0.3871	0.3750
80	0.	0.4166	0.2352	0.6250	0.3636	NaN

Figure 6.15: Precision (up) and recall (down) for 6 CMOS sensors for NB classifier when 100 randomly selected matrices were used

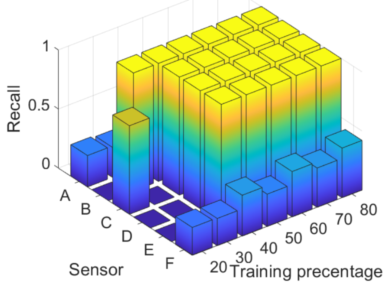
NB: Compared to the other classifiers, as shown in Figure 6.15, the precision and recall are the worst. Sensors A through F have a precision fluctuating in the range of 0-90% for all training percentages, which is an inconsistent result, while sensor E is more reliable at about 80%. The recall is below 81% for all sensors and training percentages, typically between 30% and 50%, which is again not good.

LD: According to Figure 6.16, with a few exceptions, sensors B, C, D and E frequently have precision below 50%, which is poor, while sensors A and F have a greater precision between 46% and 100% for all training percentages. Sensors A and F, which had greater precision but now have the worst recall at about 23% – 42%, represent a case where the recall is reversed. Given that most of the samples for sensors A and F are rejected, even though sensors B through E have a 100% recall rate, this seems to be less valuable.

Summarizing the results for 100 randomly chosen matrices, sensors B, C, D and E are easier to identify with NN, KNN and LD than sensors A and F. Recall rates of 60% or below for various devices and classifiers indicate that it is necessary to increase the size of the feature vectors, which is done next. The results are still not what we hoped for and are going to be improved next by using larger matrices.



x	A	B	C	D	E	F
20	0.7250	0.	0.1500	0.	0.	0.7250
30	0.8571	0.0285	0.	0.	0.	0.6000
40	0.9666	0.0333	0.2666	0.0666	0.2666	0.6000
50	0.9600	0.2400	0.1600	0.4000	0.1600	0.5600
60	0.9500	0.5000	0.3500	0.4000	0.2500	0.7500
70	1.0	0.4666	0.3333	0.8000	0.3333	0.4666
80	0.9000	0.8000	0.1000	0.7000	0.4000	0.8000



x	A	B	C	D	E	F
20	0.2660	NaN	0.7500	NaN	NaN	0.2357
30	0.2542	1.0	NaN	NaN	NaN	0.2307
40	0.2660	1.0	1.0	1.0	1.0	0.3461
50	0.3200	1.0	1.0	1.0	1.0	0.2745
60	0.3725	1.0	1.0	1.0	1.0	0.3846
70	0.3750	1.0	1.0	1.0	1.0	0.3333
80	0.4285	1.0	1.0	1.0	1.0	0.4210

Figure 6.16: Precision (up) and recall (down) for 6 CMOS sensors for LD classifier when 100 randomly selected matrices were used

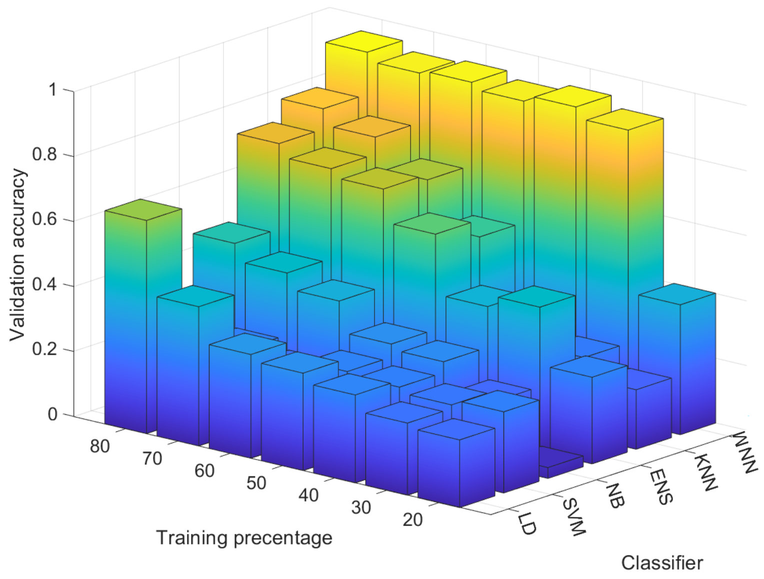
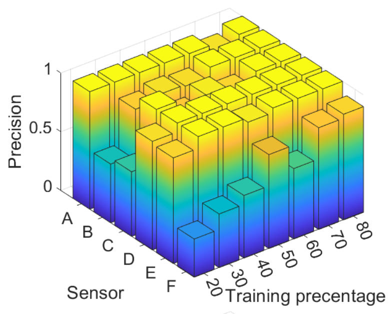
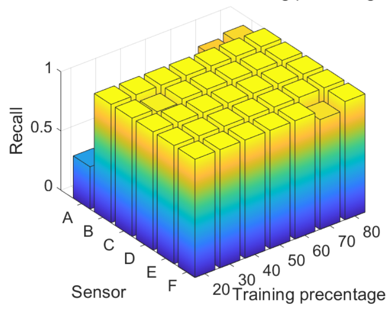


Figure 6.17: Validation accuracy when 1000 randomly selected rows were used



x	A	B	C	D	E	F
20	1.0	0.5000	0.5500	0.9250	0.9000	0.3250
30	1.0	0.8857	1.0	1.0	1.0	0.4571
40	1.0	0.9333	0.8666	1.0	1.0	0.5333
50	1.0	0.9200	0.9600	1.0	0.9600	0.8000
60	1.0	1.0	0.9000	1.0	1.0	0.6000
70	0.9333	1.0	1.0	1.0	1.0	0.8666
80	1.0	1.0	1.0	1.0	1.0	0.9000



x	A	B	C	D	E	F
20	0.3571	1.0	1.0	1.0	1.0	1.0
30	0.6140	1.0	0.9722	1.0	1.0	1.0
40	0.6000	1.0	1.0	1.0	1.0	1.0
50	0.7352	1.0	1.0	1.0	1.0	1.0
60	0.6666	1.0	1.0	1.0	1.0	1.0
70	0.8750	1.0	1.0	1.0	1.0	0.9285
80	0.9090	1.0	1.0	1.0	1.0	1.0

Figure 6.18: Precision (up) and recall (down) for 6 CMOS sensors for WNN classifier when 1000 randomly selected matrices were used

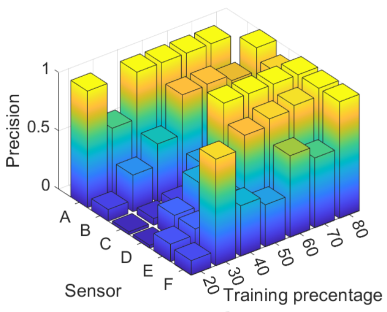
6.3.2 Results for 1000 rows from each image

The validation accuracy for the six classifiers and all test scenarios using 1000 randomly chosen rows from each image is illustrated in Figure 6.17. Again, the values in this figure are presented as average values and since the results are inconsistent, the precision and recall are discussed for each sensor. As anticipated, the validation accuracy for each classifier generally rises with the proportion of training data. As is evident, the WNN, KNN and ENS have the highest validation accuracy for all training percentages. SVM achieved the worst validation accuracy.

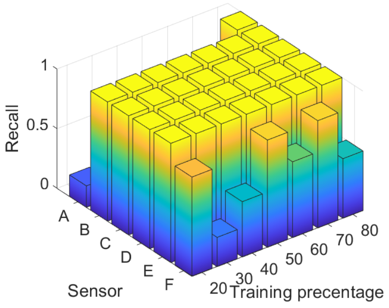
Figures 6.18, 6.19, 6.20, 6.21, 6.22 and 6.23 show the precision (up) and recall (down) as 3D barcharts and numerical values (right).

WNN: Figure 6.18 shows that sensors A – E have a precision above 86%, with a training percentage above 30%, which is good. Sensor F is identified with precision between 45% and 90%. Sensors A through F are detected with high precision between 90% and 100% for 80% training. Regarding recall, the recall for sensors B – F is nearly or exactly 100% for all training percentages. With increasing training percentages, the recall for sensor A rises from 35% at 20% training to 90% at 80% training. For all sensors, these results are good.

KNN: Compared to the WNN results, the results presented in Figure 6.19, for KNN, are less favorable. Precision is poor for training percentages below 60%, while for training percentages between 60% and 70% the precision is from 50% to 100%. Sensor A has



x	A	B	C	D	E	F
20	1.0	0.1	0.0250	0.0250	0.1500	0.1250
30	0.6000	0.3142	0.0571	0.2000	0.2000	0.9142
40	1.0	0.5000	0.1000	0.4000	0.4000	0.4333
50	1.0	0.8400	0.3600	1.0	0.8400	0.3600
60	1.0	0.8500	0.5000	1.0	0.9000	0.7000
70	1.0	0.8000	0.6666	1.0	0.9333	0.6000
80	0.1000	1.0	0.9000	1.0	1.0	1.0



x	A	B	C	D	E	F
20	0.1801	1.0	1.0	1.0	1.0	0.8333
30	0.3962	1.0	1.0	1.0	1.0	0.2461
40	0.2727	1.0	1.0	1.0	1.0	0.4642
50	0.3906	1.0	1.0	1.0	1.0	0.9000
60	0.6060	1.0	1.0	1.0	1.0	0.6363
70	0.5172	1.0	1.0	1.0	1.0	0.9000
80	1.0	1.0	1.0	1.0	1.0	0.5000

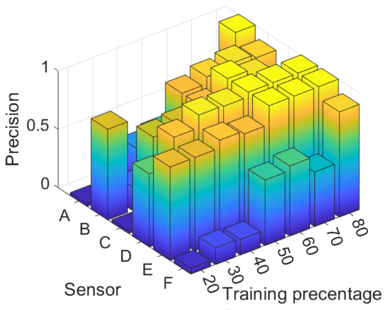
Figure 6.19: Precision (up) and recall (down) for 6 CMOS sensors for KNN classifier when 1000 randomly selected matrices were used

a very poor precision of 10% for an 80% training, sensor C has a precision of 90%, while sensors B, D, E and F reach a precision of 100%. Sensors B through E have a 100% recall for all training percentages, while sensors A and F have a variable recall between 18% and 100%. Overall, the KNN's results are not too poor, but they are not very consistent. For example, sensor A's precision went from 10% to 100%.

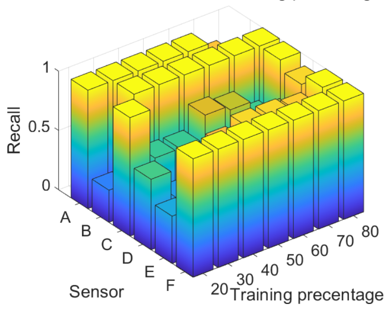
ENS: The precision achieved in the case of the ENS is comparable to the precision for KNN as sown in Figure 6.20. The precision is poor for sensors A and F for training percentages under 70%, but it reaches 100% for sensors B, C, D and E. For sensors A, D and E, precision is 100% for training 80%, while for sensors B and F the precision reach 90% and only 70% for sensor C. For sensor F, the recall is 100% for all training percentages, while sensor B reaches the same 100% (except 20% training, which does not seem like enough). At 80% training, the recall for A, C, D and E is between 71% and 100%. D and E perform poorly, with results hardly reaching even 50%.

SVM: Figure 6.21 shows that the SVM classifier is quickly discarded because the precision for sensors C and D is zero for all training percentages. In some training sets, sensors A and B also result in a precision of 0%, but sensor F has a precision of less than 35%. Even if sensor E achieves 100% accuracy, this happens because all sensors are being incorrectly assigned as sensor E. The recall rate for all sensors is below 66%, which is a problem.

NB: The results depicted in Figure 6.22 are comparable to those of the SVM. Sensors A and F are classified with a precision close to 0% for all training percentages, whereas

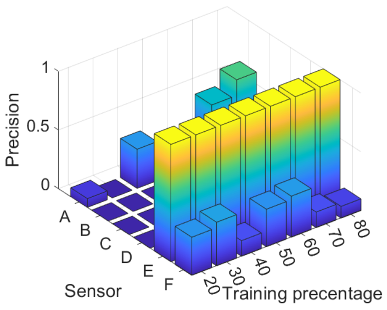


x	A	B	C	D	E	F
20	0.0250	0.7750	0.0250	0.6250	0.8000	0.0500
30	0.0285	0.2000	0.7428	0.8571	0.8000	0.1428
40	0.3333	0.5666	0.7666	0.9666	0.8666	0.1333
50	0.3200	0.8400	0.8400	0.9600	0.8400	0.5600
60	0.2500	0.9000	1.0	1.0	1.0	0.6000
70	0.4000	0.9333	0.9333	1.0	1.0	0.4666
80	1.0	0.9000	0.7000	1.0	1.0	0.9000

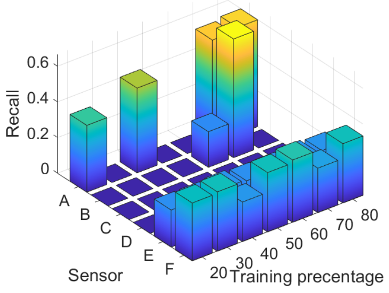


x	A	B	C	D	E	F
20	1.0	0.2719	1.0	0.5952	0.4000	1.0
30	1.0	1.0	0.5777	0.3614	0.4058	1.0
40	1.0	1.0	0.5750	0.3718	0.8387	1.0
50	1.0	1.0	0.7777	0.4210	0.9130	1.0
60	1.0	1.0	0.7407	0.5555	0.9090	1.0
70	0.8571	1.0	0.5833	0.7142	0.8823	1.0
80	0.7142	1.0	1.0	0.9090	1.0	1.0

Figure 6.20: Precision (up) and recall (down) for 6 CMOS sensors for ENS classifier when 1000 randomly selected matrices were used

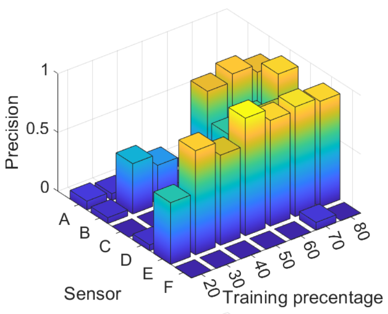


x	A	B	C	D	E	F
20	0.0750	0.	0.	0.	1.0	0.3250
30	0.	0.	0.	0.	1.0	0.3714
40	0.3333	0.	0.	0.	1.0	0.1333
50	0.	0.	0.	0.	1.0	0.3200
60	0.	0.0500	0.	0.	1.0	0.3500
70	0.4666	0.1333	0.	0.	1.0	0.1333
80	0.6000	0.	0.	0.	1.0	0.1000

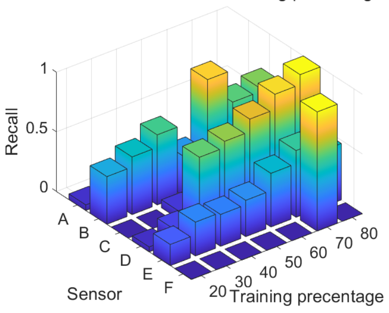


x	A	B	C	D	E	F
20	0.3750	NaN	NaN	NaN	0.2083	0.3250
30	NaN	0.	NaN	NaN	0.2083	0.3333
40	0.4761	0.	NaN	NaN	0.2173	0.2222
50	NaN	0.	NaN	NaN	0.2100	0.3333
60	0.	0.2000	NaN	NaN	0.2127	0.3500
70	0.5833	0.6666	NaN	NaN	0.2238	0.2500
80	0.6000	NaN	0.	NaN	0.2173	0.3333

Figure 6.21: Precision (up) and recall (down) for 6 CMOS sensors for SVM classifier when 1000 randomly selected matrices were used



x	A	B	C	D	E	F
20	0.0750	0.0500	0.	0.0500	0.5250	0.
30	0.0571	0.4285	0.	0.1142	0.8857	0.
40	0.	0.3333	0.1000	0.4000	0.7666	0.
50	0.	0.0800	0.2800	0.5200	1.0	0.
60	0.	0.8000	0.5500	0.7500	0.9000	0.
70	0.	0.8666	0.6000	0.7333	0.9333	0.0666
80	0.1000	0.8000	0.9000	0.5000	0.9000	0.



x	A	B	C	D	E	F
20	0.04687	0.4000	0.	0.0416	0.1779	NaN
30	0.1176	0.4838	0.	0.1081	0.2844	NaN
40	0.	0.5882	0.0666	0.6315	0.2674	NaN
50	0.	0.3333	0.1794	0.6842	0.3048	NaN
60	0.	0.8888	0.2894	0.7894	0.4500	NaN
70	0.	0.6190	0.3103	0.9166	0.5600	1.0
80	0.2500	0.6666	0.4500	1.0	0.4736	NaN

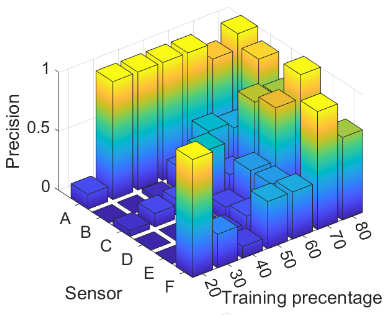
Figure 6.22: Precision (up) and recall (down) for 6 CMOS sensors for NB classifier when 1000 randomly selected matrices were used

sensors B -– E are identified with a precision between 0% and 93%. With a few exceptions, the recall values are typically between 0% and 60%, which is not very good.

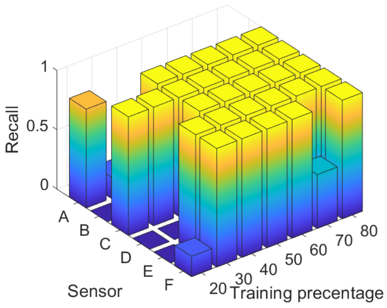
LD: Figure 6.23 shows that sensors B–F typically provide precision below 40% for all training percentages, which is poor, while sensor A typically provides a precision of 100%. On the other side, sensor A, which had superior precision, now has the worst recall, typically around 20%, with some exceptions. Although sensors B-F have a 100% recall rate, this seems to be of little help given that most samples for sensor A are rejected and that the precision for sensors B – F is also poor.

The WNN produced the best results out of 1000 randomly chosen matrices, followed closely by the KNN, although at a large margin. While SVM and NB no longer worked for 1000 randomly chosen matrices, ENS continues to classify for high training percentages. Sensors A and F are the ones that produce the worst results. Even for these two, the recall at 80% training is 90.9 – 100% and the precision is 90 – 100% with the WNN. This demonstrates that the WNN is capable of distinguishing the sensors correctly.

The SVM, KNN and WNN all have the best validation accuracy for 100 rows. While NB produces the lowest results, ENS and LD produce similar results. Only five of the six cameras are accurately identified, even though SVM has the maximum accuracy. This implies that identification requires more than 100 rows. Last but not least, traditional machine learning methods may outperform neural networks in situations with inadequate data. WNN produced the most excellent results for all training percentages when applied to 1000 rows, followed by KNN and ENS. Even when compared to NB, SVM produces



x	A	B	C	D	E	F
20	0.1250	0.	0.0500	0.	0.	1.0
30	1.0	0.	0.1142	0.	0.0571	0.2857
40	1.0	0.0333	0.1333	0.0333	0.1000	0.1000
50	1.0	0.2000	0.4800	0.1200	0.1600	0.4000
60	1.0	0.4500	0.2000	0.3500	0.3500	0.4000
70	0.8666	0.4000	0.7333	0.8000	0.1333	1.0
80	1.0	0.9000	0.7000	1.0	0.3000	0.7000



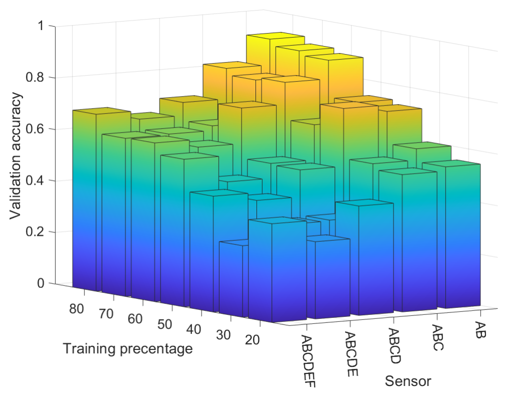
x	A	B	C	D	E	F
20	0.8333	NaN	1.0	NaN	NaN	0.1724
30	0.1804	NaN	1.0	NaN	1.0	1.0
40	0.1785	1.0	1.0	1.0	1.0	1.0
50	0.2155	1.0	1.0	1.0	1.0	1.0
60	0.2352	1.0	1.0	1.0	1.0	1.0
70	0.5000	1.0	1.0	1.0	1.0	0.4545
80	0.4166	1.0	1.0	1.0	1.0	1.0

Figure 6.23: Precision (up) and recall (down) for 6 CMOS sensors for LD classifier when 1000 randomly selected matrices were used

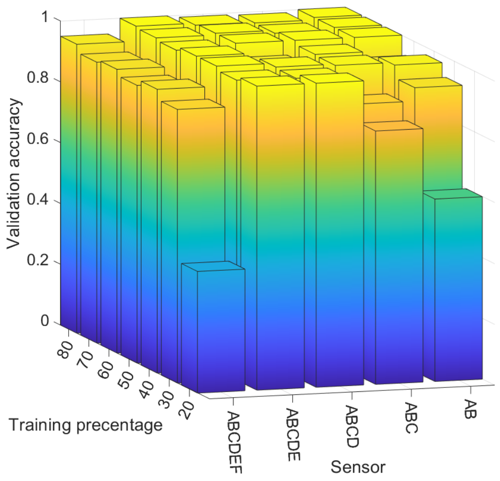
the lowest results. With 1000 samples, the WNN exceeds conventional machine learning techniques.

Although each classifier is run ten times, choosing different random rows each time, the outcomes remained consistent. Over-fitting may have contributed to the typical machine learning algorithms' poor performance in the case of 1000 rows. The authors of [221] and [222] have also observed performance deterioration for SVM due to over-fitting in a different setting. This may help to explain why SVM was the top classifier for 100 rows but fell to last place for 1000 rows. To improve the results, PCA (principal component analysis) is also used, but no changes are made. The greatest demands for training time, up to 60 minutes for 1000 matrices, are made by NB. Other classifiers needed less than 8 minutes for 1000 rows and roughly 1 minute for 100 rows.

The WNN classifier is evaluated for two, three, four and five sensors to examine the validation accuracy for classifying various sensors. The classification results for two sensors (A and B), three (A, B and C), four (A, B, C and D), five (A, B, C, D and E) and finally all six sensors (A, B, C, D, E and F) are shown in Figure 6.24. The results for 100 rows are shown in Figure 6.24 (i) and the results for 1000 rows are shown in Figure 6.24 (ii). As more sensors are added, the validation accuracy drops, as shown in Figure 6.24 (i), particularly for the case of 100 rows. Figure 6.24 (ii) illustrates that when the number of rows is extended to 1000, the fluctuation in validation accuracy for classifying 2 – 6 sensors is not significant. The accuracy decreases as the number of sensors increases, but the results are still good if enough data is added, i.e., 1000 rows of AC coefficients.



(i) Validation accuracy for 100 rows



(ii) Validation accuracy for 1000 rows

Figure 6.24: Validation accuracy for 2, 3, 4, 5 and 6 sensors when 100 randomly selected rows were used (i) and 1000 randomly selected rows used (ii) for WNN classifier

6.4 Concluding remarks

This chapter addressed smartphone fingerprinting using the low and mid-frequency AC coefficients from the DCT of dark photos. For this purpose, users must take a picture while holding their phone in their palm to capture black images. For this purpose, 50 photos were taken using six identical cameras from Samsung Galaxy J5 phones. According to the investigation, the blue channel is more effective at recognizing the camera. Six machine learning algorithms were employed to recognize the devices. The wide neural network (WNN), which had an accuracy of 97% for 1000 samples and about 70% for 100 samples, had the best results. The conventional KNN method also produced promising results, achieving an accuracy rate of about 80% for both 100 and 1000 samples.

Chapter 7

Extensions outside the smartphone domain, ECU fingerprinting

Due to the interest of the author in the area of automotive security, a side objective of the current reasearch was the application of fingerprinting technologies on Electronic Control Units. Since a dataset containing physical fingerprints is already public [223], the application of the previous machine learning toolset from Matlab was immediate. This chapter presents the results of the author in this direction, which are currently accepted for publication in [26], and it is no surprise that these techniques that yield good results for smartphones, perform well in this area too.

7.1 Fingerprinting technologies in the automotive domains

The concept of electronic component fingerprinting is used in various domains. Not unexpectedly, the first area to be explored was that of personal computers. The authors in [224] discuss remote physical device fingerprinting based on the deviation of clock skews. They analyze the TCP and ICMP timestamps and extract the fingerprints for Windows and Linux devices. Physical-layer fingerprinting of Ethernet devices was proposed in [225].

A more recent application of fingerprinting technologies is in the area of automotive electronics. Nowadays, vehicles are equipped with several hundreds of ECUs that communicate between them using CAN, FlexRay, Ethernet, LIN, etc. In figure 7.1 a modern car equipped with several ECUs is depicted. ECU fingerprinting has been discussed in many research papers published in recent years. There are several methods for ECU fingerprinting proposed by researchers in order to mitigate the attacks on in-vehicle communication networks. Voltage-based ECU fingerprinting has been proposed many years ago [226] and since then a significant number of research works followed. The authors of [227] propose a system for identifying attacker ECUs based on voltage measured on

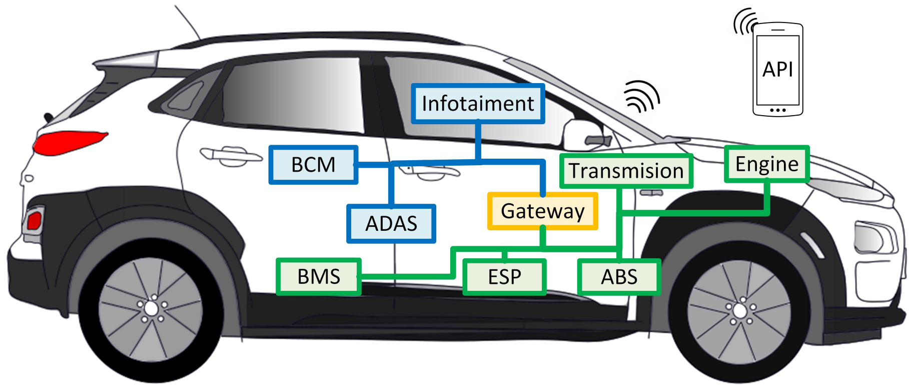


Figure 7.1: A modern car equiped with several ECUs

the in-vehicle network. The voltage profiles are used as fingerprints in order to identify the attacker ECUs. The voltage of CAN signals is used in [228] to detect intrusions and can also separate between errors and bus-off attacks. Three machine learning algorithms, i.e., KNN, SVM and logistic regression are used in [229] for ECU fingerprinting. The classifiers are trained on statistical features extracted from the voltage information. The authors in [230] discuss ECU fingerprinting based on voltage for the CAN nodes using machine learning algorithms. It is worth mentioning that using voltage for ECU fingerprinting may encounter several limitations in the real life because the battery voltage, engine state or temperature may influence the voltage levels on CAN nodes.

Other methods have been also considered for ECU fingerprinting. The authors in [231] propose a method for ECU fingerprinting based on temperature variations. They analyze how the temperature impacts the clock offset of an ECU. The experiments were performed at temperatures ranging from 0°C to 20°C. Several ECUs are near the engine and the engine temperature influences their temperature. The authors in [231] heated the engine to 80°C, measured the temperature of several ECUs and observed that the temperature was different from one ECU to another. The engine temperature can be influenced by several factors, e.g., ambient temperature, wind, fuel type, load, etc. Moreover, the placement of ECUs inside the vehicle can differ from vehicle to vehicle. The authors in [232] discuss ECU fingerprinting based on the clock behavior of the ECU. They analyze the intervals of periodic messages using the Recursive Least Square algorithm and then detect the abnormal shifts in the clock skews by performing a cumulative sum. Statistical features extracted from the CAN transceiver of each ECU are used to extract fingerprints in [233], [234]. Vehicle identification based on RFID (Radio Frequency Identification) was discussed in [235]. The authors in [236] discuss vehicle fingerprinting based on CAN bus data. The authors in [237], use a multi-layer perceptron neural network and hyper-parameter tuning to fingerprint ECUs. They select several statistical features extracted from the clock of the CAN-H and train the neural network on them. The material imper-

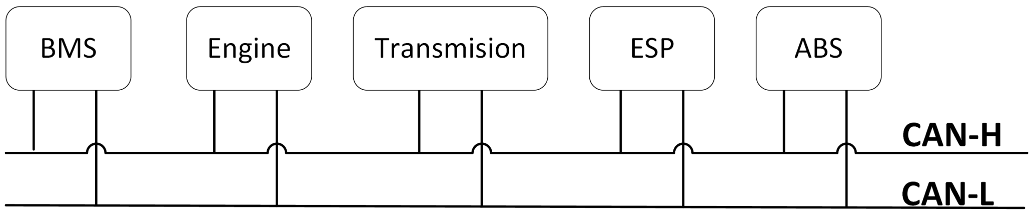


Figure 7.2: An in-vehicle CAN bus with 5 ECUs

fections and semiconductor impurities are used in [238], as features to train the Gaussian Naive Bayes classifier. This method reaches an accuracy of 100% in ECU classification. The following section depicts a method for fingerprinting ECUs using machine learning algorithms.

In Figure 7.2, an in-vehicle CAN bus with five ECUs is illustrated. The CAN bus has two wires called CAN-H and CAN-L. An internal architecture of an in-vehicle ECU is depicted in Figure 7.3. The main components of an ECU are the power supply, a microcontroller, a memory, I/O interfaces, a CAN controller and a CAN driver. The ECU fingerprinting can be done based on the bus voltage characteristics as suggested by the fingerprint placed on the bus (other components, like the local clock of the controller, may be used but are out of for the current analysis). The bus voltage on an in-vehicle CAN network is depicted in Figure 7.4. When the voltages on CAN-H and CAN-L are equal, i.e., 2.5V, the bit is recessive, i.e., 1 binary, and when CAN-H is 3.75V and CAN-L is 1.25V, the bit is dominant, i.e., 0 binary. In Figure 7.5, the voltages from 3 bits collected from 3 distinct ECUs from a Hyundai i20 according to the dataset from [223] are illustrated. There are visible variations between the three bits, which makes the separation between the three ECUs feasible.

7.2 Machine learning on physical layer signals

In this section, the experimental findings on ECU identification using physical layer data are presented. The results demonstrate that while single voltage attributes like maximum or minimum voltages may be helpful for smaller ECU pools, they become insufficient as the pool grows. Traditional machine learning algorithms perform better in this scenario, but only neural networks appear to be able to distinguish between samples accurately.

7.2.1 The ECUPrint dataset

A complete physical layer dataset, ECUPrint [223], containing information collected from 10 vehicles, nine passenger vehicles and one heavy-duty vehicle complying with the J1939 standard, has just been made available to the public. It is also important to note

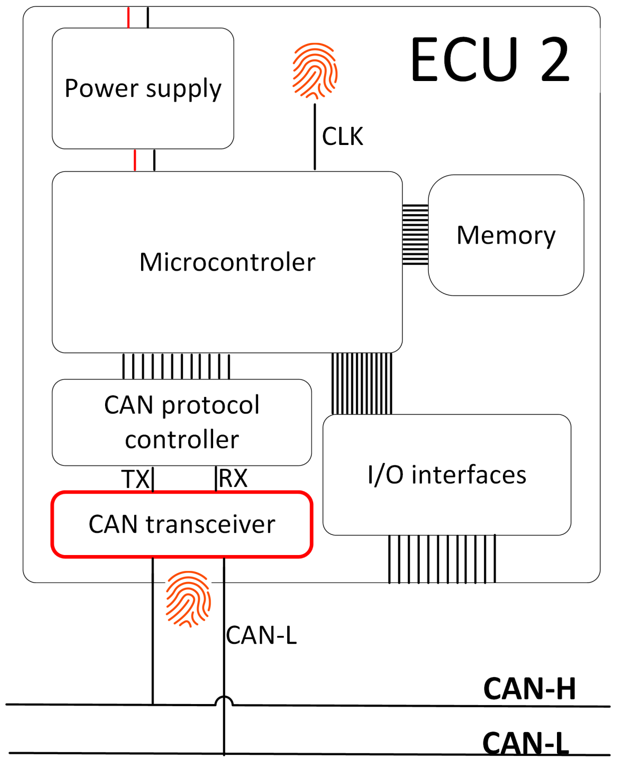


Figure 7.3: Internal architecture of an in-vehicle ECU

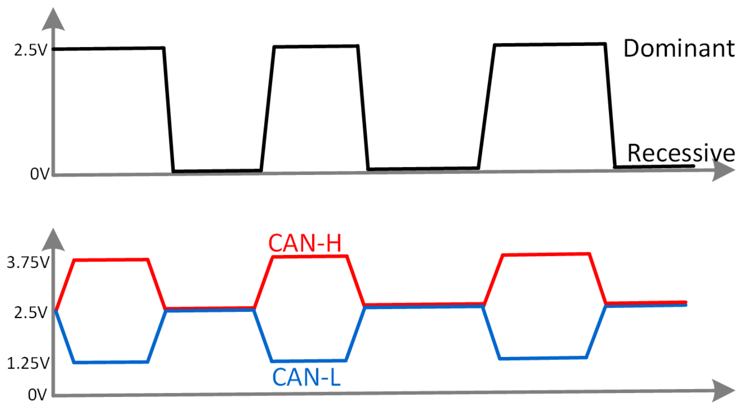


Figure 7.4: Bus voltage on in-vehicle CAN network

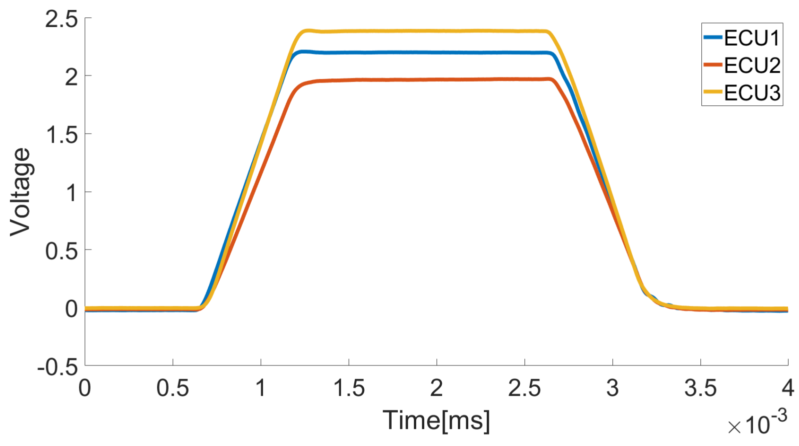


Figure 7.5: Bus voltage for a dominant CAN bit, i.e., the differential voltage between CAN-H and CAN-L

that the ECUPrint paper [223] promotes the use of physical fingerprints for forensics, i.e., the identification of vehicles that may be the target of theft, VIN cloning, or the illegal replacement/modification of in-vehicle ECUs, an application of physical fingerprints that has not previously been taken into consideration. The authors suggest using four features, each of which can be retrieved from isolated bits, that is, a dominant bit split between two recessive bits: 1) mean voltage, 2) peak voltage, 3) bit time and 4) plateau time. The report notes that using just one feature out of the four results in significant overlaps amongst ECUs and more features should be combined. The classification of the ECUs is sufficiently accurate, with just minor overlaps when all four criteria are used. Machine learning techniques enable recognition with very high accuracy, above 99.9%, as will be demonstrated later.

Between three and nine ECUs are present in each car in the dataset, each with a unique ID. There are between 20 and 20187 sample records for each ECU in the collection due to the extraction of multiple bits from each ID. One measurement consists of 2000 sampling points for the nine-passenger vehicles, whereas for the heavy-duty vehicle, it consists of 2700 sampling points (this happens because of the distinct data rate from the heavy-duty truck, i.e., 250 Kbps vs. 500 Kbps in regular cars). Only data from the nine passenger vehicles, which have a combined 51 ECUs, will be used in this study. The number of sampled bits may differ for each ECU, but each measurement has exactly 2000 sample points.

7.2.2 Results using neural networks and conventional classifiers

Using the ECUPrint dataset, the performance of five machine learning algorithms is analyzed: Linear Discriminant (LD), Decision Trees (Tree), Support Vector Machines

(SVM), K-Nearest Neighbors (KNN) and a simple neural network (NN). These algorithms are all available in the Matlab toolset [239].

Using different amounts of data for training and testing, the performance of the five machine learning techniques stated above is analyzed. The classifier implementation offered by Matlab 2021a is used. The tests were run on a laptop with an Intel Core i7-9850H processor and 32GB RAM.

The first analysis shows that using machine learning classifiers with two voltage features, i.e., two sampling points from a single bit, is insufficient. Four features (the average voltage, peak value, bit time and plateau time) would be necessary for such separation, according to the authors of [223]. The accuracy of the KNN classifiers was only 48.21% when identifying a node using the maximum and minimum voltage features. The average values for FAR, FRR, precision and recall were 1.06%, 57.29%, 42.71% and 45.75%, respectively, when using 80% of the data for training. The confusion matrix for a KNN is shown in Figure 7.6 when 20% of the data are used for testing and 80% of the data are used for training when using maximum and minimum voltage as the two features. The ECU classes are represented by the true and predicted class axes, with the letter designating a particular vehicle and the number identifying a specific ECU inside a vehicle. For example, ECU1 from automobile A is A1, ECU2 from car B is B2, etc. The diagonal elements of the confusion matrix are marked in green as being the correctly identified ECUs and the outer elements are highlighted in shades of red denoting incorrectly identified ECUs. The ECUs may be successfully identified locally within a single vehicle. However, there are definite overlaps among the 51 ECUs from various automobiles. Using only these two features, the KNN classifier outperformed the other classifiers in terms of performance. More than two voltage features are needed to separate an ECU.

In order to cover all 2000 data points for each bit taken from the ECUPrint dataset, the input of classifiers was extended. The FAR produced using 20% and 80% training examples for each of the 51 ECUs when applying all five machine learning algorithms is presented as bar charts in Figure 7.7. In order to improve the training percentages up to 80%, the dataset from each ECU is randomly split into 20% training data and 80% testing data. The Tree and LD classifiers perform poorly when 20% of the dataset is used for training, with FARs of 1.3% for Tree and 1.4% for LD, respectively. KNN, SVM and NN show better results. The FAR for KNN is up to 0.046%, whereas the FAR for SVM is up to 0.05%. The results for NN are marginally better in terms of FAR, with values ranging from 0 to 0.01%. The FAR values rise for two ECUs when the training data percentage is increased to 80%, with the Tree classifier reaching 0.28% for one of them. For the remaining classifiers, improvements can be seen, with the FAR increasing for the NN classifier to 0.006%.

The FRR for 20% and 80% of the data used as training examples is shown in Figure 7.8. When only 20% of the dataset is used for training, the FRR for the Tree and LD classifiers is considerably too high, approaching 100% for several ECUs. The results for KNN, SVM and NN are more encouraging, with FRRs of up to 39% for KNN, less than

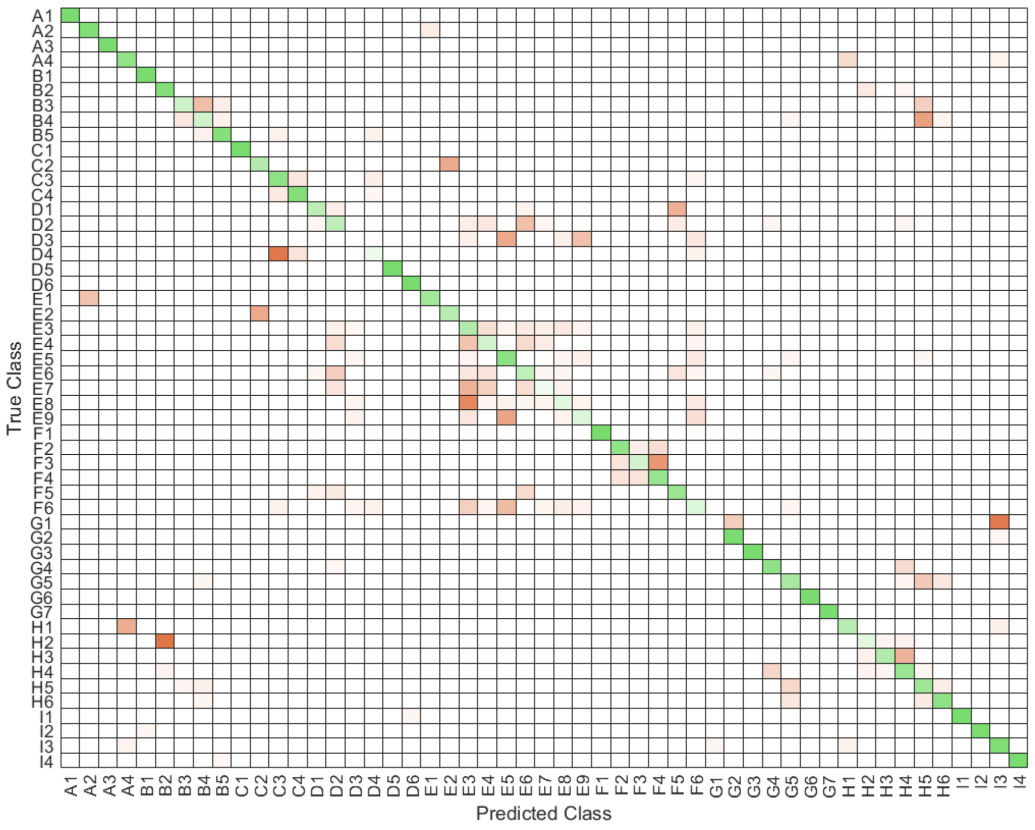
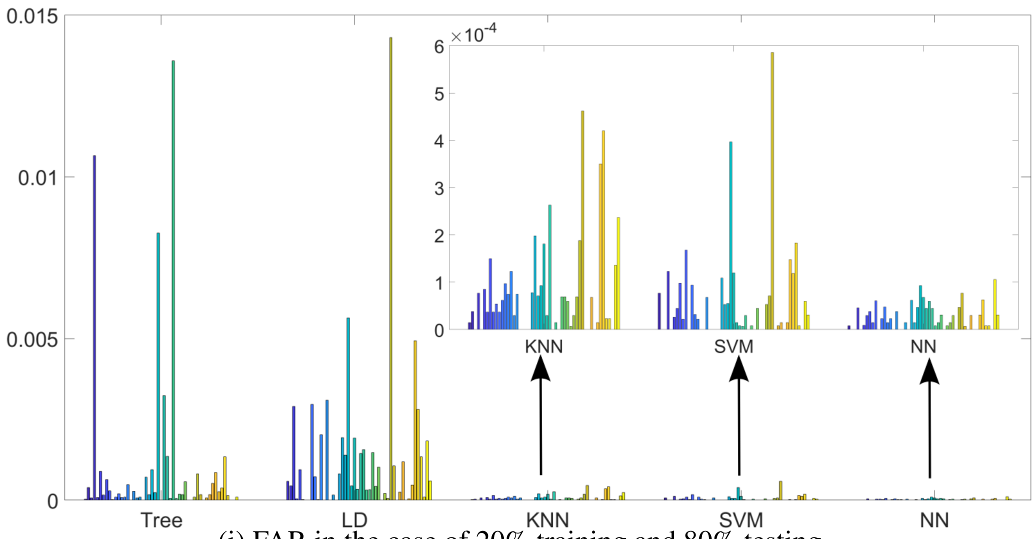
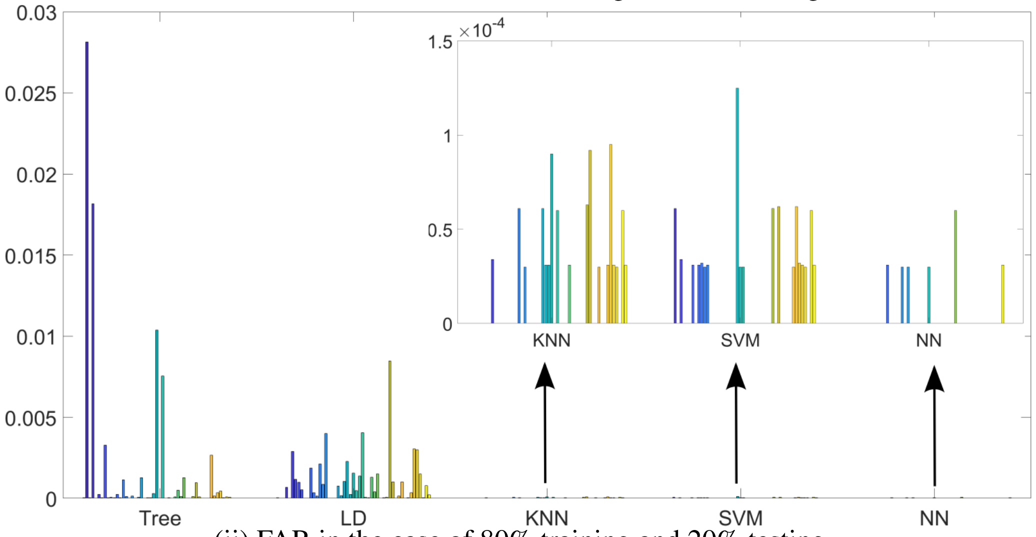


Figure 7.6: For the 51 ECUs, the confusion matrix for KNN at 80% training on two features (max and min value)



(i) FAR in the case of 20% training and 80% testing



(ii) FAR in the case of 80% training and 20% testing

Figure 7.7: FAR in the case of 20% and 80% training for 51 ECUs

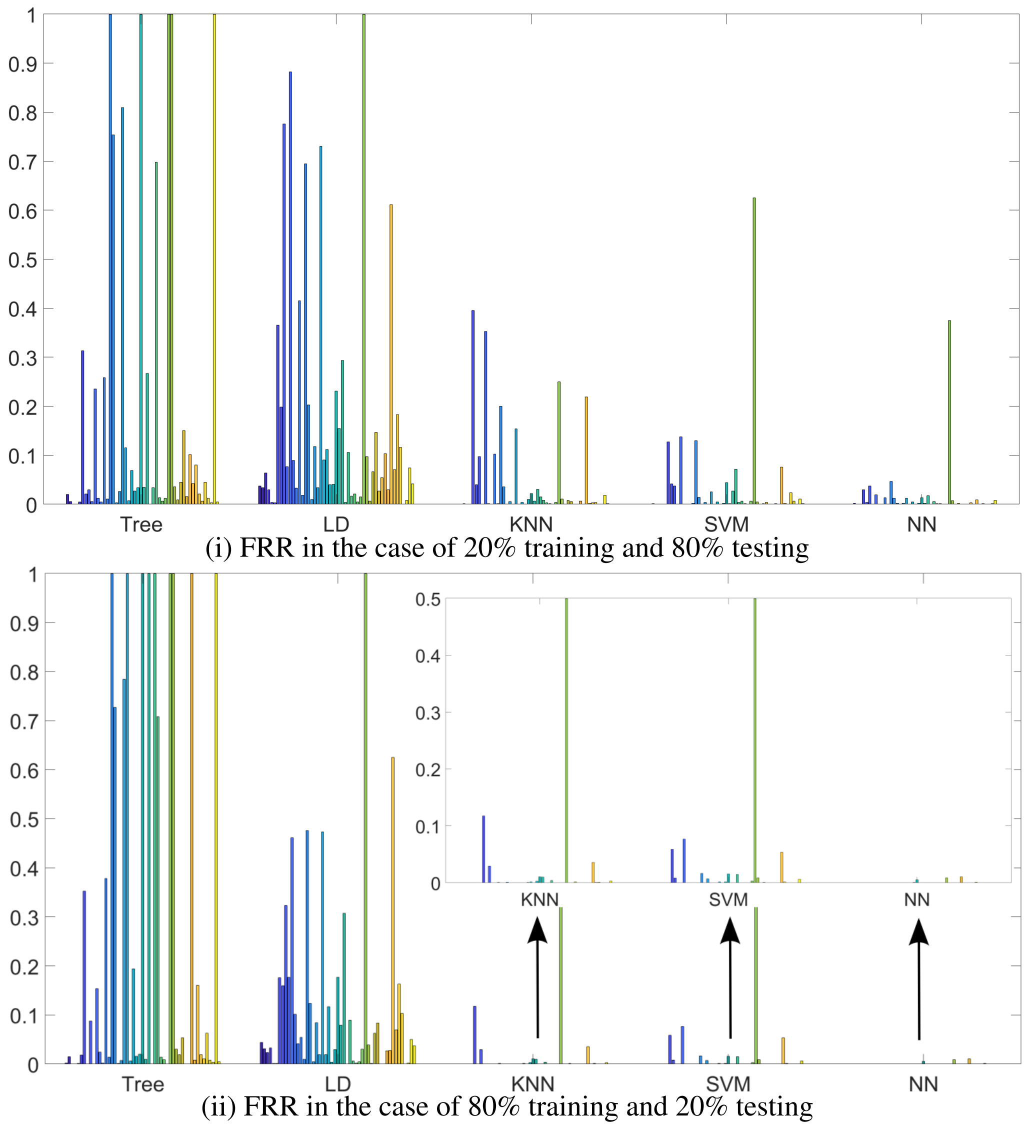


Figure 7.8: FRR for 20% and 80% training for 51 ECUs

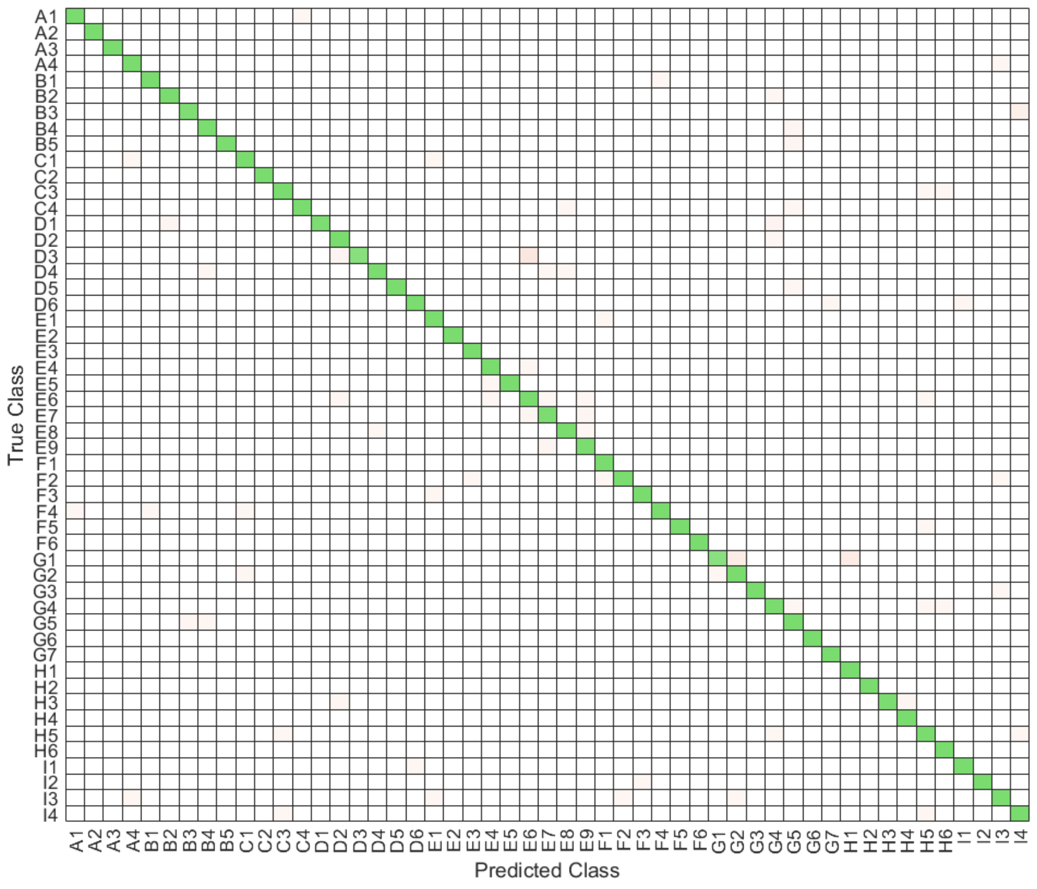


Figure 7.9: Confusion matrix in the case of NN with 20% training for the 51 ECUs

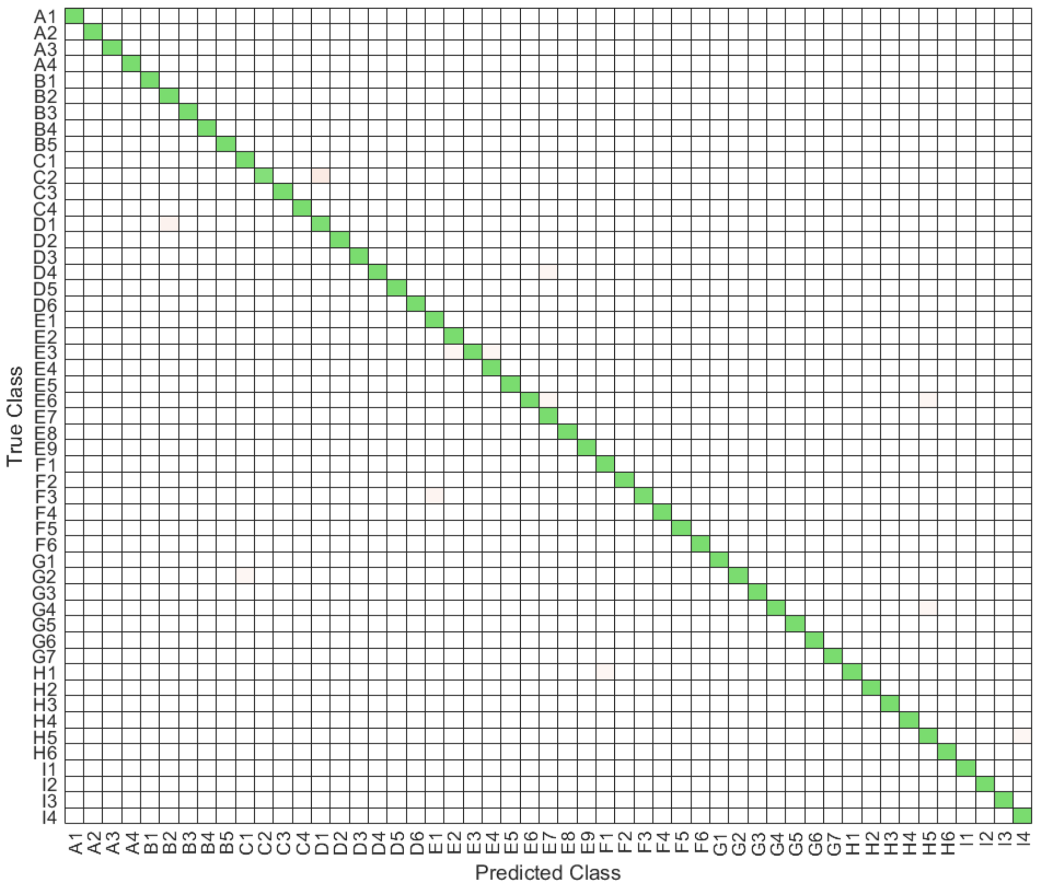


Figure 7.10: Confusion matrix in the case of NN with 80% training for the 51 ECUs

Table 7.1: Results for the five classifiers at 20% – 80% training

Trn.	Alg.	Acc.	FAR			FRR			Precision			Recall			F1-score		
			Min	Avg	Max	Min	Avg	Max	Min	Avg	Max	Min	Avg	Max	Min	Avg	Max
20%	Tree	0.949	0	$< 10^{-3}$	0.013	0	0.184	1.0	0	0.815	1.0	0.418	NaN	1.0	0.319	NaN	0.999
	LD	0.940	0	0.001	0.0143	0	0.168	1.0	0	0.831	1.0	0.161	NaN	1.0	0.201	NaN	0.995
	KNN	0.994	0	$< 10^{-4}$	$< 10^{-3}$	0	0.039	0.395	0.604	0.960	1.0	0.868	0.985	1.0	0.736	0.971	1.0
	SVM	0.995	0	$< 10^{-4}$	$< 10^{-3}$	0	0.0280	0.625	0.375	0.971	1.0	0.951	0.994	1.0	0.545	0.980	1.0
	NN	0.998	0	$< 10^{-4}$	$< 10^{-4}$	0	0.012	0.375	0.625	0.987	1.0	0.833	0.992	1.0	0.714	0.989	1.0
40%	Tree	0.951	0	$< 10^{-3}$	0.013	0	0.194	1.0	0	0.805	1.0	0.279	NaN	1.0	0.229	NaN	1.0
	LD	0.948	0	0.001	0.0128	0	0.132	1.0	0	0.867	1.0	0.394	NaN	1.0	0.434	NaN	1.0
	KNN	0.997	0	$< 10^{-4}$	$< 10^{-3}$	0	0.023	0.250	0.750	0.976	1.0	0.939	0.992	1.0	0.841	0.983	1.0
	SVM	0.997	0	$< 10^{-4}$	$< 10^{-3}$	0	0.018	0.416	0.583	0.981	1.0	0.921	0.994	1.0	0.736	0.987	1.0
	NN	0.999	0	$< 10^{-4}$	$< 10^{-4}$	0	0.001	0.031	0.968	0.998	1.0	0.857	0.996	1.0	0.923	0.997	1.0
60%	Tree	0.941	0	0.001	0.018	0	0.236	1.0	0	0.763	1.0	0.269	NaN	1.0	0.205	NaN	0.999
	LD	0.953	0	$< 10^{-3}$	0.008	0	0.127	1.0	0	0.872	1.0	0.127	NaN	1.0	0.142	NaN	1.0
	KNN	0.998	0	$< 10^{-4}$	$< 10^{-3}$	0	0.0169	0.192	0.807	0.983	1.0	0.898	0.992	1.0	0.875	0.987	1.0
	SVM	0.998	0	$< 10^{-4}$	$< 10^{-3}$	0	0.011	0.375	0.625	0.988	1.0	0.928	0.996	1.0	0.769	0.991	1.0
	NN	0.999	0	$< 10^{-4}$	$< 10^{-4}$	0	0.002	0.027	0.972	0.997	1.0	0.973	0.998	1.0	0.923	0.997	1.0
80%	Tree	0.928	0	0.001	0.028	0	0.253	1.0	0	0.746	1.0	0.343	NaN	1.0	0.265	NaN	1.0
	LD	0.951	0	$< 10^{-3}$	0.008	0	0.117	1.0	0	0.882	1.0	0.275	NaN	1.0	0.360	NaN	1.0
	KNN	0.998	0	$< 10^{-4}$	$< 10^{-3}$	0	0.0141	0.500	0.500	0.985	1.0	0.984	0.998	1.0	0.666	0.990	1.0
	SVM	0.999	0	$< 10^{-4}$	$< 10^{-3}$	0	0.015	0.500	0.500	0.984	1.0	0.973	0.998	1.0	0.666	0.989	1.0
	NN	0.999	0	$< 10^{-4}$	$< 10^{-4}$	0	$< 10^{-3}$	0.010	0.989	0.999	1.0	0.666	0.993	1.0	0.800	0.995	1.0

20% for SVM (except one node, for which the FRR is 62%) and less than 10% for NN (except one ECU, for which the FRR is 37.5%). While the results for LD indicate only a small increases, for the remaining 20% of the training data, the FRR for the Tree classifier hits 100% for even more ECUs. Except for one node with an FRR of 50%, the KNN and SVM algorithms now reveal significant gains with FRR values below 12%. NN proved to be the most reliable, with FRRs of around 1% for 80% training.

The confusion matrix produced by employing NN with 20% of the data used for training is shown in Figure 7.9. The ECUs could occasionally be misidentified at this lower training percentage, but this is generally a rare occurrence. To complete the picture, the confusion matrix for NN is shown in Figure 7.10, where 80% of the data is employed for training and the remaining 20% for testing. Misidentifications in this situation are highly uncommon; for instance, C2 is rarely mistaken for D1 and D1 is rarely mistaken for B2. The misidentification rate will virtually reduce to zero when multiple bits are employed because this classification is based on data from a single bit while several bits are accessible in each frame.

The results for all metrics, including the minimum, average and maximum values of FAR, FRR, precision, recall and F1-score for the five classifiers when utilizing 20%, 40%, 60% and 80% of the dataset for training, are summarized as numerical values in Table 7.1. For some ECUs, the total of true positives and false negatives is zero. Hence the value NaN denotes division by zero. The results improve when the training percentage is increased, but less than one might anticipate. This shows that a small amount of data should be adequate. The best classifiers are KNN, SVM and NN. The NN surpassed the other classifiers with an accuracy above 99.9%.

7.3 Concluding remarks

This final chapter containing the author's results, tied to briefly analyze how fingerprinting technologies can be extended to other components, such as in-vehicle ECUs. Five machine learning algorithms: Linear Discriminant (LD), Decision Trees (Tree), SVM, KNN and a wide neural network (NN), were used to fingerprint 51 ECUs from several vehicles. When all features were used, the NN reached an accuracy of 99.9%, while when only two features were used (which was performed only for KNN), the classification result was not good giving only 48.21% accuracy. Consequently, good results for physical layer-based fingerprinting can be achieved with neural networks. However, some issues still need to be resolved before such methods can be implemented in existing in-vehicle networks, e.g., voltage fluctuations due to temperature changes, battery discharge, engine state, etc.

Chapter 8

Conclusion

The thesis addressed fingerprinting technologies for smartphone embedded transducers, i.e., accelerometers, loudspeakers, microphones and camera sensors. As a result of this thesis, several relevant datasets were built containing data collected from transducers using dozens of smartphones and distinct experimental scenarios. The collected data was analyzed in order to extract sensor characteristics and several machine learning algorithms were used to perform the classification. Existing research publications on smartphone fingerprinting were also analyzed as related work, along with the techniques they used to identify devices and the corresponding results. As a final application to transducer fingerprinting, in line with many other recent research works, this thesis also discussed the identification of ECUs (Electronic Control Unit) based on CAN voltage levels with many of the machine learning classifiers previously used for smartphones. As a final conclusion, a brief summary of the findings from each chapter now follows.

Chapter 2 briefly presented the principles of operation behind smartphone transducers, i.e., accelerometers, loudspeakers, microphones and camera sensors. The most popular feature extraction methods, popular classification algorithms and an overview of evaluation metrics were presented. This chapter also illustrated some application scenarios and preventive measures against the exploitation of smartphone fingerprinting as a privacy leak. Regarding each sensor presented in the thesis, the related works showed the following. Accelerometers were widely used for device authentication (pairing) and it is surprising that only a few works discussed smartphone fingerprinting based on accelerometers. Loudspeakers were employed much less frequently in research than microphones were. It is possible that less research was done on device fingerprinting based on audio signals from loudspeakers because, while such data is simple to evaluate, it is more challenging to collect. There are several publicly available datasets for microphones (the majority of them focusing on speech recognition and criminal investigations), which are also utilized for device identification based on microphone characteristics. To the best of the author's knowledge, the only public dataset available for loudspeaker identification is the result of the research done for this thesis. The topic of camera fingerprinting was addressed by the largest body of research works compared to all other sensors, based on the works surveyed in this thesis. This is expected given that consumers frequently submit images to several websites, making such data relatively easy to gather and likely leading to privacy concerns. Additionally, photos can be used to extract a wide variety of samples and attributes and numerous public datasets were available.

Chapter 3 discussed smartphone pairing in several transportation modes based on accelerometer data and also smartphone fingerprinting based on accelerometers. Accelerometer signals differ substantially depending on the mode of mobility. The findings from this thesis demonstrated that acquiring enough entropy from the accelerometer data was possible in order to create a session key in all transportation environments. Low-entropy extractions can also lead to secure session keys by relying on guessing-resilient protocols (that allow matching such values without exposing them to a brute-force adversarial search). Most of the filtering methods employed gave comparable results. However, simple scaling of the accelerometer measurements was the most suitable choice due to its simplicity. The entropy was increased by extending the feature vectors with sigma-delta modulation, but this required more computations because more features had to be traded. Given the variations in transportation situations, specific parameters may be advantageous depending on the case and the trade-off between the level of security and pairing probability. By addressing the adversarial advantage and the pairing success rate, a more precise image of this approach was provided. Using a variety of machine learning classifiers (including NN, KNN, SVM, Ensemble and Decision Trees) on seven time-domain variables (including Kurtosis, Skewness, SNR, STD, RMS, peak value and SINAD), this chapter also discussed smartphone fingerprinting based on accelerometer sensors. The Ensemble classifier provided maximum recognition accuracy of 100% for the dataset, which includes instances from five separate and identical phones. The results demonstrated that classical machine learning algorithms can produce good results for fingerprinting smartphones based on accelerometer sensors. Using more sophisticated deep learning architectures seem unnecessary, especially when training data is limited.

Chapter 4 approached smartphone fingerprinting based on loudspeaker characteristics. Smartphones can be fingerprinted using the methods summarized in this thesis, possibly making them useful as smart keys identifiable based on physical characteristics. In this thesis, an effective fingerprinting technique was investigated that can be quickly applied to identify smartphones based on loudspeaker roll-off characteristics. According to the findings, loudspeaker roll-offs offer a reliable fingerprint that is more resistant to variations in volume levels. In contrast, for some techniques, the volume level may be deceptive. While the slope of the roll-offs alone was adequate to identify different smartphone models, deep-learning methods were required for a more thorough analysis of loudspeakers coming from the same smartphone models. The discrimination between such identical loudspeakers can be done with an accuracy of 90–99% using the LSTM and BiLSTM neural network designs. One specific application scenario was the use of smartphones inside vehicles, which is why most of the experiments conducted in this chapter employed a car's head unit to record the smartphone sounds. Regardless of the recorder, the identification had a high success rate, indicating that in-vehicle infotainment units are usable in such scenarios.

Chapter 5 investigated smartphone microphone fingerprinting employing the signal power spectrum and various supervised machine learning methods (including Linear Discriminant, Ensemble-Subspace Discriminant, Fine Tree, Fine KNN and Linear SVM). Three critical contexts for fingerprinting were investigated: human speech, synthetically simulated environmental sound and live recordings. Additional noise was introduced to all scenarios to make classification more difficult. The LD classifier acted perfectly in the first two cases. These fingerprints have a wide range of potential uses. For example, verifying ownership of a specific phone to serve as a second authentication token with physical characteristics that cannot be cloned. However, such fingerprinting could also be abused by mobile applications to fingerprint devices without access to device-unique identifiers. Malicious apps (or libraries hidden within) with high-fidelity access to microphone samples already have a more significant impact on security and privacy [161] than the additional device fingerprint.

In Chapter 6, smartphone fingerprinting based on their camera sensors was also investigated using the low and mid-frequency AC coefficients obtained from the DCT of dark photos. The investigation showed that the blue channel is more effective at recognizing the camera. A dark picture needs to be taken for this purpose, by holding the smartphone against the user's palm. Six machine learning algorithms were employed to identify the devices. For the classification, 50 photos were taken using six identical cameras from Samsung Galaxy J5 smartphones. The wide neural network (WNN), which had an accuracy of 97% for 1000 samples and roughly 70% for 100 samples, had the best results. The conventional KNN algorithm also gave promising outcomes, with an accuracy of about 80% for both 100 and 1000 samples.

Last but not least, in Chapter 7, the author analyzed and tested how fingerprinting techniques can be extended to other components, in this case, in-vehicle ECUs. Five machine learning algorithms: Linear Discriminant (LD), Decision Trees (Tree), SVM, KNN and a wide neural network (NN), which were also used for smartphones, were also used here to fingerprint 51 ECUs based on a publicly available dataset. The classifiers were tested on all features from the dataset. When all features were used, NN reached an accuracy of 99.9%, while when only two features were used, the ECUs performance of the KNN was not so good. Suggesting, as already known in the literature, that a reduced number of features is not sufficient for accurate classification.

To sum up, this thesis provided positive results for the classification of smartphones based on four transducers: accelerometers, loudspeakers, microphones and cameras. One of the main findings of this research was that traditional machine learning algorithms can give even better results than more complicated deep neural network architectures for sensor fingerprinting. Comprehensive datasets were also publicly released for loudspeaker and microphone data evolving more than 60 smartphones. The results of this PhD work have been submitted and accepted for publication in relevant ISI journals and conferences.

Nevertheless, the research done for this thesis can be considered as a basis for future investigations. There are several directions that can be further explored. For example, studying how sensors are affected by other environmental elements, such as temperature or various kinds of noise, is possible. Also, more experiments concerning specific in-vehicle use cases and details, such as passenger/phone positions, as well as other environmental noises can be investigated. Practical deployments inside cars require more experiments, which may be considered in upcoming works. Other interesting scenarios may consist of authentication at electric vehicle charging stations which are now a common presence and may be subjected to adversarial attacks which can be circumvented by using physical characteristics that are hard or impossible to clone.

List of Figures

List of Tables

Bibliography

[1] T. Petroc, "Number of smartphone subscriptions worldwide from 2016 to 2021, with forecasts from 2022 to 2027," 2023. [Online]. Available: https://www.statista.com/statistics/330695/number-of-smartphone-users-worldwide/

[2] K. Lofstrom, W. R. Daasch, and D. Taylor, "Ic identification circuit using device mismatch," in *2000 IEEE International Solid-State Circuits Conference. Digest of Technical Papers (Cat. No. 00CH37056)*. IEEE, 2000, pp. 372–373.

[3] B. Gassend, D. Clarke, M. Van Dijk, and S. Devadas, "Silicon physical random functions," in *Proceedings of the 9th ACM conference on Computer and communications security*, 2002, pp. 148–160.

[4] V. K. Khanna, "Remote fingerprinting of mobile phones," *IEEE Wireless Communications*, vol. 22, no. 6, pp. 106–113, 2015.

[5] G. Baldini and G. Steri, "A survey of techniques for the identification of mobile phones using the physical fingerprints of the built-in components," *IEEE Communications Surveys & Tutorials*, vol. 19, no. 3, pp. 1761–1789, 2017.

[6] N. Kanjan, K. Patil, S. Ranaware, and P. Sarokte, "A comparative study of fingerprint matching algorithms," *International Research Journal of Engineering and Technology*, vol. 4, no. 11, pp. 1892–1896, 2017.

[7] K. Ren, Z. Qin, and Z. Ba, "Toward hardware-rooted smartphone authentication," *IEEE Wireless Communications*, vol. 26, no. 1, pp. 114–119, 2019.

[8] S. J. Alsunaidi and A. M. Almuhaideb, "Investigation of the optimal method for generating and verifying the smartphone's fingerprint: A review," *Journal of King Saud University-Computer and Information Sciences*, 2020.

[9] P. M. S. Sánchez, J. M. J. Valero, A. H. Celdrán, G. Bovet, M. G. Pérez, and G. M. Pérez, "A survey on device behavior fingerprinting: Data sources, techniques, application scenarios, and datasets," *IEEE Communications Surveys & Tutorials*, 2021.

[10] M. Mohanty, M. Zhang, M. R. Asghar, and G. Russello, "e-prnu: Encrypted domain prnu-based camera attribution for preserving privacy," *IEEE Transactions on Dependable and Secure Computing*, vol. 18, no. 1, pp. 426–437, 2019.

[11] A. Konstantinidis, G. Chatzimilioudis, D. Zeinalipour-Yazti, P. Mpeis, N. Pelekis, and Y. Theodoridis, "Privacy-preserving indoor localization on smartphones," *IEEE Transactions on Knowledge and Data Engineering*, vol. 27, no. 11, pp. 3042–3055, 2015.

[12] S. A. Anand, C. Wang, J. Liu, N. Saxena, and Y. Chen, "Spearphone: a lightweight speech privacy exploit via accelerometer-sensed reverberations from smartphone loudspeakers," in *Proceedings of the 14th ACM Conference on Security and Privacy in Wireless and Mobile Networks*, 2021, pp. 288–299.

[13] P. Hu, H. Zhuang, P. S. Santhalingam, R. Spolaor, P. Pathak, G. Zhang, and X. Cheng, "Accear: Accelerometer acoustic eavesdropping with unconstrained vocabulary," in *2022 IEEE Symposium on Security and Privacy (SP)*. IEEE, 2022, pp. 1757–1773.

[14] C. Wang, F. Lin, T. Liu, Z. Liu, Y. Shen, Z. Ba, L. Lu, W. Xu, and K. Ren, "mmphone: Acoustic eavesdropping on loudspeakers via mmwave-characterized piezoelectric effect," in *IEEE INFOCOM 2022-IEEE Conference on Computer Communications*. IEEE, 2022, pp. 820–829.

[15] A. Mongia, V. M. Gunturi, and V. Naik, "Detecting activities at metro stations using smartphone sensors," in *2018 10th International Conference on Communication Systems & Networks (COMSNETS)*. IEEE, 2018, pp. 57–65.

[16] K. A. Nguyen, Y. Wang, G. Li, Z. Luo, and C. Watkins, "Realtime tracking of passengers on the london underground transport by matching smartphone accelerometer footprints," *Sensors*, vol. 19, no. 19, p. 4184, 2019.

[17] N. Roy, H. Wang, and R. Roy Choudhury, "I am a smartphone and i can tell my user's walking direction," in *Proceedings of the 12th annual international conference on Mobile systems, applications, and services*, 2014, pp. 329–342.

[18] A. Anjum and M. U. Ilyas, "Activity recognition using smartphone sensors," in *2013 ieee 10th consumer communications and networking conference (ccnc)*. IEEE, 2013, pp. 914–919.

[19] X. Su, H. Tong, and P. Ji, "Activity recognition with smartphone sensors," *Tsinghua science and technology*, vol. 19, no. 3, pp. 235–249, 2014.

[20] B. Groza, A. Berdich, C. Jichici, and R. Mayrhofer, "Secure accelerometer-based pairing of mobile devices in multi-modal transport," *IEEE Access*, vol. 8, pp. 9246–9259, 2020.

[21] A. Berdich, B. Groza, R. Mayrhofer, E. Levy, A. Shabtai, and Y. Elovici, "Sweep-to-unlock: Fingerprinting smartphones based on loudspeaker roll-off characteristics," *IEEE Transactions on Mobile Computing*, 2021.

[22] A. Berdich, B. Groza, E. Levy, A. Shabtai, Y. Elovici, and R. Mayrhofer, "Fingerprinting smartphones based on microphone characteristics from environment affected recordings," *IEEE Access*, vol. 10, pp. 122 399–122 413, 2022.

[23] A. Berdich and B. Groza, "Smartphone camera identification from low-mid frequency dct coefficients of dark images," *Entropy*, vol. 24, no. 8, p. 1158, 2022.

[24] A. Berdich, P. Iosif, C. Burlacu, A. Anistoroaei, and B. Groza, "Fingerprinting smartphone accelerometers with traditional classifiers and deep learning networks," in *2023 IEEE 17th International Symposium on Applied Computational Intelligence and Informatics (SACI)*, 2023.

[25] A. Berdich and B. Groza, "Secure by design autonomous emergency braking systems in accordance with iso 21434," *Machine Learning and Optimization Techniques for Automotive Cyber-Physical Systems*, 2023.

[26] S. Murvay, A. Berdich, and B. Groza, "Physical layer intrusion detection and localization on can bus," *Machine Learning and Optimization Techniques for Automotive Cyber-Physical Systems*, 2023.

[27] B. Groza, H. Gurban, L. Popa, A. Berdich, and S. Murvay, "Car-to-smartphone interactions: Experimental setup, risk analysis and security technologies," in *5th International Workshop on Critical Automotive Applications: Robustness & Safety*, 2019.

[28] A. Berdich, A. Anistoroaei, B. Groza, H. Gurban, S. Murvay, and D. Iercan, "Antares-anonymous transfer of vehicle access rights from external cloud services," in *2020 IEEE 91st Vehicular Technology Conference (VTC2020-Spring)*. IEEE, 2020, pp. 1–5.

[29] B. Groza, T. Andreica, A. Berdich, P. Murvay, and E. H. Gurban, "Prestvo: Privacy enabled smartphone based access to vehicle on-board units," *IEEE Access*, vol. 8, pp. 119 105–119 122, 2020.

[30] A. Anistoroaei, A. Berdich, P. Iosif, and B. Groza, "Secure audio-visual data exchange for android in-vehicle ecosystems," *Applied Sciences*, vol. 11, no. 19, p. 9276, 2021.

[31] C. Jichici, A. Berdich, A. Musuroi, and B. Groza, "Control system level intrusion detection on j1939 heavy-duty vehicle buses."

[32] L. Popa, A. Berdich, and B. Groza, "Cartwin—development of a digital twin for a real-world in-vehicle can network," *Applied Sciences*, vol. 13, no. 1, p. 445, 2022.

[33] A. Berdich and B. Groza, "Cyberattacks on adaptive cruise controls and emergency braking systems: Adversary models, impact assessment and countermeasures."

[34] A. Berdich, B. Groza, and R. Mayrhofer, "A survey on fingerprinting technologies for smartphones based on embedded transducers."

[35] D. Litwiller, "Ccd vs. cmos," *Photonics spectra*, vol. 35, no. 1, pp. 154–158, 2001.

[36] S. Dey, N. Roy, W. Xu, R. R. Choudhury, and S. Nelakuditi, "AccelPrint: Imperfections of Accelerometers Make Smartphones Trackable." in *NDSS*, 2014.

[37] Z. Ding and M. Ming, "Accelerometer-based mobile device identification system for the realistic environment," *IEEE Access*, vol. 7, pp. 131 435–131 447, 2019.

[38] A. Das, N. Borisov, and E. Chou, "Every move you make: Exploring practical issues in smartphone motion sensor fingerprinting and countermeasures." *Proc. Priv. Enhancing Technol.*, vol. 2018, no. 1, pp. 88–108, 2018.

[39] A. Das, N. Borisov, and M. Caesar, "Exploring ways to mitigate sensor-based smartphone fingerprinting," *arXiv preprint arXiv:1503.01874*, 2015.

[40] ——, "Tracking mobile web users through motion sensors: Attacks and defenses." in *NDSS*, 2016.

[41] T. Hupperich, H. Hosseini, and T. Holz, "Leveraging sensor fingerprinting for mobile device authentication," in *International Conference on Detection of Intrusions and Malware, and Vulnerability Assessment*. Springer, 2016, pp. 377–396.

[42] I. Amerini, R. Becarelli, R. Caldelli, A. Melani, and M. Niccolai, "Smartphone fingerprinting combining features of on-board sensors," *IEEE Transactions on Information Forensics and Security*, vol. 12, no. 10, pp. 2457–2466, 2017.

[43] G. Baldini, G. Steri, I. Amerini, and R. Caldelli, "The identification of mobile phones through the fingerprints of their built-in magnetometer: An analysis of the portability of the fingerprints," in *2017 International Carnahan Conference on Security Technology (ICCST)*. IEEE, 2017, pp. 1–6.

[44] T. Van Goethem, W. Scheepers, D. Preuveneers, and W. Joosen, "Accelerometer-based device fingerprinting for multi-factor mobile authentication," in *International Symposium on Engineering Secure Software and Systems*. Springer, 2016, pp. 106–121.

[45] J. Lukas, J. Fridrich, and M. Goljan, “Digital camera identification from sensor pattern noise,” *IEEE Transactions on Information Forensics and Security*, vol. 1, no. 2, pp. 205–214, 2006.

[46] T. Gloe, M. Kirchner, A. Winkler, and R. B¨ohme, “Can we trust digital image forensics?” in *Proceedings of the 15th ACM international conference on Multimedia*, 2007, pp. 78–86.

[47] M. Chen, J. Fridrich, M. Goljan, and J. Lukás, “Determining image origin and integrity using sensor noise,” *IEEE Transactions on information forensics and security*, vol. 3, no. 1, pp. 74–90, 2008.

[48] Q.-T. Phan, G. Boato, and F. G. De Natale, “Image clustering by source camera via sparse representation,” in *Proceedings of the 2nd International Workshop on Multimedia Forensics and Security*, 2017, pp. 1–5.

[49] F. Marra, G. Poggi, C. Sansone, and L. Verdoliva, “Blind prnu-based image clustering for source identification,” *IEEE Transactions on Information Forensics and Security*, vol. 12, no. 9, pp. 2197–2211, 2017.

[50] D. Valsesia, G. Coluccia, T. Bianchi, and E. Magli, “User authentication via prnu-based physical unclonable functions,” *IEEE Transactions on Information Forensics and Security*, vol. 12, no. 8, pp. 1941–1956, 2017.

[51] R. Deka, C. Galdi, and J.-L. Dugelay, “Hybrid g-prnu: Optimal parameter selection for scale-invariant asymmetric source smartphone identification,” *Electronic Imaging*, vol. 2019, no. 5, pp. 546–1, 2019.

[52] V. Bruni, A. Salvi, and D. Vitulano, “Joint correlation measurements for prnu-based source identification,” in *International Conference on Computer Analysis of Images and Patterns*. Springer, 2019, pp. 245–256.

[53] M. S. Behare, A. Bhalchandra, and R. Kumar, “Source camera identification using photo response noise uniformity,” in *2019 3rd International conference on Electronics, Communication and Aerospace Technology (ICECA)*. IEEE, 2019, pp. 731–734.

[54] A. T. Erozan, M. Hefenbrock, M. Beigl, J. Aghassi-Hagmann, and M. B. Tahoori, “Image puf: A physical unclonable function for printed electronics based on optical variation of printed inks,” *Cryptology ePrint Archive*, 2019.

[55] S. Taspinar, M. Mohanty, and N. Memon, “Camera fingerprint extraction via spatial domain averaged frames,” *IEEE Transactions on Information Forensics and Security*, vol. 15, pp. 3270–3282, 2020.

[56] J. Bernacki, “On robustness of camera identification algorithms,” *Multimedia Tools and Applications*, vol. 80, no. 1, pp. 921–942, 2021.

[57] M. Iuliani, M. Fontani, and A. Piva, “A leak in prnu based source identification—questioning fingerprint uniqueness,” *IEEE Access*, vol. 9, pp. 52 455–52 463, 2021.

[58] S. Ikram and H. Malik, “Microphone identification using higher-order statistics,” in *Audio engineering society conference: 46th international conference: audio forensics*. Audio Engineering Society, 2012.

[59] F. Kurniawan, M. S. M. Rahim, M. S. Khalil, and M. K. Khan, “Statistical-based audio forensic on identical microphones,” *International Journal of Electrical and Computer Engineering*, vol. 6, no. 5, p. 2211, 2016.

[60] Z. Zhou, W. Diao, X. Liu, and K. Zhang, “Acoustic fingerprinting revisited: Generate stable device id stealthily with inaudible sound,” in *Proceedings of the 2014 ACM SIGSAC Conference on Computer and Communications Security*, 2014, pp. 429–440.

[61] A. Lawgaly and F. Khelifi, “Sensor pattern noise estimation based on improved locally adaptive dct filtering and weighted averaging for source camera identification and verification,” *IEEE Transactions on Information Forensics and Security*, vol. 12, no. 2, pp. 392–404, 2016.

[62] D. Valsesia, G. Coluccia, T. Bianchi, and E. Magli, “Large-scale image retrieval based on compressed camera identification,” *IEEE Transactions on Multimedia*, vol. 17, no. 9, pp. 1439–1449, 2015.

[63] J. R. Corripio, D. A. González, A. S. Orozco, L. G. Villalba, J. Hernandez-Castro, and S. J. Gibson, “Source smartphone identification using sensor pattern noise and wavelet transform,” 2013.

[64] M. Tiwari and B. Gupta, “Efficient prnu extraction using joint edge-preserving filtering for source camera identification and verification,” in *2018 IEEE Applied Signal Processing Conference (ASPCON)*. IEEE, 2018, pp. 14–18.

[65] L. Debiasi, E. Leitet, K. Norell, T. Tachos, and A. Uhl, “Blind source camera clustering of criminal case data,” in *2019 7th International Workshop on Biometrics and Forensics (IWBF)*. IEEE, 2019, pp. 1–6.

[66] D. Cozzolino, F. Marra, D. Gragnaniello, G. Poggi, and L. Verdoliva, “Combining prnu and noiseprint for robust and efficient device source identification,” *EURASIP Journal on Information Security*, vol. 2020, no. 1, pp. 1–12, 2020.

[67] S. Mandelli, D. Cozzolino, P. Bestagini, L. Verdoliva, and S. Tubaro, "Cnn-based fast source device identification," *IEEE Signal Processing Letters*, vol. 27, pp. 1285–1289, 2020.

[68] C.-T. Li, X. Lin, K. A. Kotegar *et al.*, "Beyond prnu: Learning robust device-specific fingerprint for source camera identification," *arXiv preprint arXiv:2111.02144*, 2021.

[69] B. N. Sarkar, S. Barman, and R. Naskar, "Blind source camera identification of on-line social network images using adaptive thresholding technique," in *Proceedings of International Conference on Frontiers in Computing and Systems*. Springer, 2021, pp. 637–648.

[70] R. Rouhi, F. Bertini, and D. Montesi, "No matter what images you share, you can probably be fingerprinted anyway," *Journal of Imaging*, vol. 7, no. 2, 2021. [Online]. Available: https://www.mdpi.com/2313-433X/7/2/33

[71] J. Bernacki, "A survey on digital camera identification methods," *Forensic Science International: Digital Investigation*, vol. 34, p. 300983, 2020.

[72] I. IEC, "Information technology-digital compression and coding of continuous-tone still images: Requirements and guidelines," *Standard, ISO IEC*, pp. 10 918–1, 1994.

[73] A. Roy, R. S. Chakraborty, V. U. Sameer, and R. Naskar, "Camera source identification using discrete cosine transform residue features and ensemble classifier." in *CVPR Workshops*, 2017, pp. 1848–1854.

[74] B. Gupta and M. Tiwari, "Improving performance of source-camera identification by suppressing peaks and eliminating low-frequency defects of reference spn," *IEEE Signal processing letters*, vol. 25, no. 9, pp. 1340–1343, 2018.

[75] A. El-Yamany, H. Fouad, and Y. Raffat, "A generic approach cnn-based camera identification for manipulated images," in *2018 IEEE International Conference on Electro/Information Technology (EIT)*. IEEE, 2018, pp. 0165–0169.

[76] G. Xu and Y. Q. Shi, "Camera model identification using local binary patterns," in *2012 IEEE International Conference on Multimedia and Expo*. IEEE, 2012, pp. 392–397.

[77] N. Zandi and F. Razzazi, "Source camera identification using wlbp descriptor," in *2020 International Conference on Machine Vision and Image Processing (MVIP)*. IEEE, 2020, pp. 1–6.

[78] ¨O. ESKİDERE, "Source microphone identification from speech recordings based on a gaussian mixture model," *Turkish Journal of Electrical Engineering & Computer Sciences*, vol. 22, no. 3, pp. 754–767, 2014.

[79] R. Aggarwal, S. Singh, A. K. Roul, and N. Khanna, "Cellphone identification using noise estimates from recorded audio," in *2014 International Conference on Communication and Signal Processing*. IEEE, 2014, pp. 1218–1222.

[80] C. Hanilçi and T. Kinnunen, "Source cell-phone recognition from recorded speech using non-speech segments," *Digital Signal Processing*, vol. 35, pp. 75–85, 2014.

[81] ¨O. Eskidere and A. Karatutlu, "Source microphone identification using multitaper mfcc features," in *2015 9th International Conference on Electrical and Electronics Engineering (ELECO)*. IEEE, 2015, pp. 227–231.

[82] Y. Li, X. Zhang, X. Li, Y. Zhang, J. Yang, and Q. He, "Mobile phone clustering from speech recordings using deep representation and spectral clustering," *IEEE Transactions on Information Forensics and Security*, vol. 13, no. 4, pp. 965–977, 2017.

[83] X. Li, D. Yan, L. Dong, and R. Wang, "Anti-forensics of audio source identification using generative adversarial network," *IEEE Access*, vol. 7, pp. 184 332–184 339, 2019.

[84] V. A. Hadoltikar, V. R. Ratnaparkhe, and R. Kumar, "Optimization of mfcc parameters for mobile phone recognition from audio recordings," in *2019 3rd International conference on Electronics, Communication and Aerospace Technology (ICECA)*. IEEE, 2019, pp. 777–780.

[85] A. Das, N. Borisov, and M. Caesar, "Fingerprinting smart devices through embedded acoustic components," *arXiv preprint arXiv:1403.3366*, 2014.

[86] ——, "Do you hear what i hear?: Fingerprinting smart devices through embedded acoustic components," in *Proceedings of the 2014 ACM SIGSAC Conference on Computer and Communications Security*. ACM, 2014, pp. 441–452.

[87] T. Qin, R. Wang, D. Yan, and L. Lin, "Source cell-phone identification in the presence of additive noise from cqt domain," *Information*, vol. 9, no. 8, p. 205, 2018.

[88] Y. Kim and Y. Lee, "Campuf: physically unclonable function based on cmos image sensor fixed pattern noise," in *Proceedings of the 55th Annual Design Automation Conference*, 2018, pp. 1–6.

[89] X. Lu, L. Hong, and K. Sengupta, "Cmos optical pufs using noise-immune process-sensitive photonic crystals incorporating passive variations for robustness," *IEEE Journal of Solid-State Circuits*, vol. 53, no. 9, pp. 2709–2721, 2018.

[90] Y. Cao, S. S. Zalivaka, L. Zhang, C.-H. Chang, and S. Chen, "Cmos image sensor based physical unclonable function for smart phone security applications," in *2014 International Symposium on Integrated Circuits (ISIC)*. IEEE, 2014, pp. 392–395.

[91] Y. Cao, L. Zhang, S. S. Zalivaka, C.-H. Chang, and S. Chen, "Cmos image sensor based physical unclonable function for coherent sensor-level authentication," *IEEE Transactions on Circuits and Systems I: Regular Papers*, vol. 62, no. 11, pp. 2629–2640, 2015.

[92] Z. Ding, W. Zhou, and Z. Zhou, "Configuration-based fingerprinting of mobile device using incremental clustering," *IEEE Access*, vol. 6, pp. 72 402–72 414, 2018.

[93] A. Maiti, V. Gunreddy, and P. Schaumont, "A systematic method to evaluate and compare the performance of physical unclonable functions," in *Embedded systems design with FPGAs*. Springer, 2013, pp. 245–267.

[94] R. Arjona, M. A. Prada-Delgado, J. Arcenegui, and I. Baturone, "Using physical unclonable functions for internet-of-thing security cameras," in *Interoperability, Safety and Security in IoT*. Springer, 2017, pp. 144–153.

[95] Y. Zheng, X. Zhao, T. Sato, Y. Cao, and C.-H. Chang, "Ed-puf: event-driven physical unclonable function for camera authentication in reactive monitoring system," *IEEE Transactions on Information Forensics and Security*, vol. 15, pp. 2824–2839, 2020.

[96] K. San Choi, E. Y. Lam, and K. K. Wong, "Source camera identification using footprints from lens aberration," in *Digital Photography II*, vol. 6069. International Society for Optics and Photonics, 2006, p. 60690J.

[97] B. Xu, X. Wang, X. Zhou, J. Xi, and S. Wang, "Source camera identification from image texture features," *Neurocomputing*, vol. 207, pp. 131–140, 2016.

[98] V. U. Sameer, A. Sarkar, and R. Naskar, "Source camera identification model: Classifier learning, role of learning curves and their interpretation," in *2017 International Conference on Wireless Communications, Signal Processing and Networking (WiSPNET)*. IEEE, 2017, pp. 2660–2666.

[99] A. Rashidi and F. Razzazi, "Single image camera identification using i-vectors," in *2017 7th International Conference on Computer and Knowledge Engineering (ICCKE)*. IEEE, 2017, pp. 406–410.

[100] Y. Huang, L. Cao, J. Zhang, L. Pan, and Y. Liu, "Exploring feature coupling and model coupling for image source identification," *IEEE Transactions on Information Forensics and Security*, vol. 13, no. 12, pp. 3108–3121, 2018.

[101] M. H. Al Banna, M. A. Haider, M. J. Al Nahian, M. M. Islam, K. A. Taher, and M. S. Kaiser, "Camera model identification using deep cnn and transfer learning approach," in *2019 International Conference on Robotics, Electrical and Signal Processing Techniques (ICREST)*. IEEE, 2019, pp. 626–630.

[102] P. R. M. Júnior, L. Bondi, P. Bestagini, S. Tubaro, and A. Rocha, "An in-depth study on open-set camera model identification," *IEEE Access*, vol. 7, pp. 180 713–180 726, 2019.

[103] R. Buchholz, C. Kraetzer, and J. Dittmann, "Microphone classification using fourier coefficients," in *International Workshop on Information Hiding*. Springer, 2009, pp. 235–246.

[104] H. Q. Vu, S. Liu, X. Yang, Z. Li, and Y. Ren, "Identifying microphone from noisy recordings by using representative instance one class-classification approach," *Journal of networks*, 2012.

[105] C. Kotropoulos and S. Samaras, "Mobile phone identification using recorded speech signals," in *2014 19th International Conference on Digital Signal Processing*. IEEE, 2014, pp. 586–591.

[106] J. Zeng, S. Shi, X. Yang, Y. Li, Q. Lu, X. Qiu, and H. Zhu, "Audio recorder forensic identification in 21 audio recorders," in *2015 IEEE International Conference on Progress in Informatics and Computing (PIC)*. IEEE, 2015, pp. 153–157.

[107] L. Zou, Q. He, and X. Feng, "Cell phone verification from speech recordings using sparse representation," in *2015 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*. IEEE, 2015, pp. 1787–1791.

[108] G. Baldini and I. Amerini, "Smartphones identification through the built-in microphones with convolutional neural network," *IEEE Access*, vol. 7, pp. 158 685–158 696, 2019.

[109] G. Baldini, I. Amerini, and C. Gentile, "Microphone identification using convolutional neural networks," *IEEE Sensors Letters*, vol. 3, no. 7, pp. 1–4, 2019.

[110] Y. Jiang and F. H. Leung, "Source microphone recognition aided by a kernel-based projection method," *IEEE Transactions on Information Forensics and Security*, vol. 14, no. 11, pp. 2875–2886, 2019.

[111] C. Jin, R. Wang, and D. Yan, “Source smartphone identification by exploiting encoding characteristics of recorded speech,” *Digital Investigation*, vol. 29, pp. 129–146, 2019.

[112] G. Baldini and I. Amerini, “An evaluation of entropy measures for microphone identification,” *Entropy*, vol. 22, no. 11, p. 1235, 2020.

[113] D. Bykhovsky, “Recording device identification by enf harmonics power analysis,” *Forensic science international*, vol. 307, p. 110100, 2020.

[114] X. Zhou, X. Zhuang, H. Tang, M. Hasegawa-Johnson, and T. S. Huang, “Novel gaussianized vector representation for improved natural scene categorization,” *Pattern Recognition Letters*, vol. 31, no. 8, pp. 702–708, 2010.

[115] D. P. Chowdhury, S. Bakshi, P. K. Sa, and B. Majhi, “Wavelet energy feature based source camera identification for ear biometric images,” *Pattern Recognition Letters*, vol. 130, pp. 139–147, 2020.

[116] J. Tian, J. Zhang, X. Li, C. Zhou, R. Wu, Y. Wang, and S. Huang, “Mobile device fingerprint identification using gyroscope resonance,” *IEEE Access*, 2021.

[117] I. Amerini, P. Bestagini, L. Bondi, R. Caldelli, M. Casini, and S. Tubaro, “Robust smartphone fingerprint by mixing device sensors features for mobile strong authentication,” *Electronic Imaging*, vol. 2016, no. 8, pp. 1–8, 2016.

[118] Y. Cheng, X. Ji, J. Zhang, W. Xu, and Y.-C. Chen, “Demicpu: Device fingerprinting with magnetic signals radiated by cpu,” in *Proceedings of the 2019 ACM SIGSAC Conference on Computer and Communications Security*, 2019, pp. 1149–1170.

[119] A. Tuama, F. Comby, and M. Chaumont, “Camera model identification with the use of deep convolutional neural networks,” in *2016 IEEE International workshop on information forensics and security (WIFS)*. IEEE, 2016, pp. 1–6.

[120] H. Yao, T. Qiao, M. Xu, and N. Zheng, “Robust multi-classifier for camera model identification based on convolution neural network,” *IEEE Access*, vol. 6, pp. 24 973–24 982, 2018.

[121] D. Freire-Obregón, F. Narducci, S. Barra, and M. Castrillón-Santana, “Deep learning for source camera identification on mobile devices,” *Pattern Recognition Letters*, vol. 126, pp. 86–91, 2019.

[122] P. Yang, R. Ni, Y. Zhao, and W. Zhao, “Source camera identification based on content-adaptive fusion residual networks,” *Pattern Recognition Letters*, vol. 119, pp. 195–204, 2019.

[123] D. Cozzolino and L. Verdoliva, "Noiseprint: A cnn-based camera model fingerprint," *IEEE Transactions on Information Forensics and Security*, vol. 15, pp. 144–159, 2019.

[124] X. Ding, Y. Chen, Z. Tang, and Y. Huang, "Camera identification based on domain knowledge-driven deep multi-task learning," *IEEE Access*, vol. 7, pp. 25 878–25 890, 2019.

[125] M. Zhao, B. Wang, F. Wei, M. Zhu, and X. Sui, "Source camera identification based on coupling coding and adaptive filter," *IEEE Access*, vol. 8, pp. 54 431–54 440, 2019.

[126] A. M. Rafi, T. I. Tonmoy, U. Kamal, Q. J. Wu, and M. K. Hasan, "Remnet: remnant convolutional neural network for camera model identification," *Neural Computing and Applications*, vol. 33, no. 8, pp. 3655–3670, 2021.

[127] D. Dal Cortivo, S. Mandelli, P. Bestagini, and S. Tubaro, "Cnn-based multi-modal camera model identification on video sequences," *Journal of Imaging*, vol. 7, no. 8, p. 135, 2021.

[128] V. Verma and N. Khanna, "Cnn-based system for speaker independent cell-phone identification from recorded audio." in *CVPR Workshops*, 2019, pp. 53–61.

[129] X. Lin, J. Zhu, and D. Chen, "Subband aware cnn for cell-phone recognition," *IEEE Signal Processing Letters*, vol. 27, pp. 605–609, 2020.

[130] M. A. Qamhan, H. Altaheri, A. H. Meftah, G. Muhammad, and Y. A. Alotaibi, "Digital audio forensics: Microphone and environment classification using deep learning," *IEEE Access*, vol. 9, pp. 62 719–62 733, 2021.

[131] J. Monaco, "Device fingerprinting with peripheral timestamps," in *2022 2022 IEEE Symposium on Security and Privacy (SP) (SP).* Los Alamitos, CA, USA: IEEE Computer Society, may 2022, pp. 243–258. [Online]. Available: https://doi.ieeecomputersociety.org/10.1109/SP46214.2022.00015

[132] A. Krizhevsky, I. Sutskever, and G. E. Hinton, "Imagenet classification with deep convolutional neural networks," *Advances in neural information processing systems*, vol. 25, pp. 1097–1105, 2012.

[133] C. Szegedy, W. Liu, Y. Jia, P. Sermanet, S. Reed, D. Anguelov, D. Erhan, V. Vanhoucke, and A. Rabinovich, "Going deeper with convolutions," in *Proceedings of the IEEE conference on computer vision and pattern recognition*, 2015, pp. 1–9.

[134] K. He, X. Zhang, S. Ren, and J. Sun, "Deep residual learning for image recognition," in *Proceedings of the IEEE conference on computer vision and pattern recognition*, 2016, pp. 770–778.

[135] Y. Liu, Z. Zou, Y. Yang, N.-F. B. Law, and A. A. Bharath, "Efficient source camera identification with diversity-enhanced patch selection and deep residual prediction," *Sensors*, vol. 21, no. 14, p. 4701, 2021.

[136] V. U. Sameer, I. Dali, and R. Naskar, "A deep learning based digital forensic solution to blind source identification of facebook images," in *International Conference on Information Systems Security*. Springer, 2018, pp. 291–303.

[137] Y. Lee, J. Li, and Y. Kim, "Micprint: acoustic sensor fingerprinting for spoof-resistant mobile device authentication," in *Proceedings of the 16th EAI International Conference on Mobile and Ubiquitous Systems: Computing, Networking and Services*, 2019, pp. 248–257.

[138] D. Chen, N. Zhang, Z. Qin, X. Mao, Z. Qin, X. Shen, and X.-Y. Li, "S2m: A lightweight acoustic fingerprints-based wireless device authentication protocol," *IEEE Internet of Things Journal*, vol. 4, no. 1, pp. 88–100, 2016.

[139] S. Wang, C. Chen, and J. Ma, "Accelerometer based transportation mode recognition on mobile phones," in *Wearable Computing Systems (APWCS), 2010 Asia-Pacific Conference on*. IEEE, 2010, pp. 44–46.

[140] S. Reddy, M. Mun, J. Burke, D. Estrin, M. Hansen, and M. Srivastava, "Using mobile phones to determine transportation modes," *ACM Transactions on Sensor Networks (TOSN)*, vol. 6, no. 2, p. 13, 2010.

[141] T. Feng and H. J. Timmermans, "Transportation mode recognition using GPS and accelerometer data," *Transportation Research Part C: Emerging Technologies*, vol. 37, pp. 118–130, 2013.

[142] M. T. Ahvanooey, M. X. Zhu, Q. Li, W. Mazurczyk, K.-K. R. Choo, B. B. Gupta, and M. Conti, "Modern authentication schemes in smartphones and iot devices: An empirical survey," *IEEE Internet of Things Journal*, pp. 1–1, 2021.

[143] F. Lorenz, L. Thamsen, A. Wilke, I. Behnke, J. Waldmüller-Littke, I. Komarov, O. Kao, and M. Paeschke, "Fingerprinting analog iot sensors for secret-free authentication," in *2020 29th International Conference on Computer Communications and Networks (ICCCN)*. IEEE, 2020, pp. 1–6.

[144] D. A. Johnson and M. M. Trivedi, "Driving style recognition using a smartphone as a sensor platform," in *2011 14th International IEEE Conference on Intelligent Transportation Systems (ITSC)*. IEEE, 2011, pp. 1609–1615.

[145] M. Van Ly, S. Martin, and M. M. Trivedi, "Driver classification and driving style recognition using inertial sensors," in *Intelligent Vehicles Symposium (IV), 2013 IEEE*. IEEE, 2013, pp. 1040–1045.

[146] N. Kalra and D. Bansal, “Analyzing driver behavior using smartphone sensors: a survey,” *Int. J. Electron. Electr. Eng*, vol. 7, no. 7, pp. 697–702, 2014.

[147] P. Singh, N. Juneja, and S. Kapoor, “Using mobile phone sensors to detect driving behavior,” in *Proceedings of the 3rd ACM Symposium on Computing for Development*. ACM, 2013, p. 53.

[148] J. Yu, Z. Chen, Y. Zhu, Y. J. Chen, L. Kong, and M. Li, “Fine-grained abnormal driving behaviors detection and identification with smartphones,” *IEEE transactions on mobile computing*, vol. 16, no. 8, pp. 2198–2212, 2017.

[149] M. Muaaz and R. Mayrhofer, “Accelerometer based gait recognition using adapted gaussian mixture models,” in *Proceedings of the 14th International Conference on Advances in Mobile Computing and Multi Media*. ACM, 2016, pp. 288–291.

[150] A. Mednis, G. Strazdins, R. Zviedris, G. Kanonirs, and L. Selavo, “Real time pothole detection using android smartphones with accelerometers,” in *Distributed Computing in Sensor Systems and Workshops (DCOSS), 2011 International Conference on*. IEEE, 2011, pp. 1–6.

[151] K. Chen, M. Lu, X. Fan, M. Wei, and J. Wu, “Road condition monitoring using on-board three-axis accelerometer and GPS sensor,” 2011.

[152] G. Hu, Z. He, and R. Lee, “Smartphone impostor detection with built-in sensors and deep learning,” *arXiv preprint arXiv:2002.03914*, 2020.

[153] J. Hua, Z. Shen, and S. Zhong, “We can track you if you take the metro: Tracking metro riders using accelerometers on smartphones,” *IEEE Transactions on Information Forensics and Security*, vol. 12, no. 2, pp. 286–297, 2016.

[154] R. Rouhi, F. Bertini, D. Montesi, X. Lin, Y. Quan, and C.-T. Li, “Hybrid clustering of shared images on social networks for digital forensics,” *IEEE Access*, vol. 7, pp. 87 288–87 302, 2019.

[155] A. Pandya and R. K. Shukla, “New perspective of nanotechnology: role in preventive forensic,” *Egyptian Journal of Forensic Sciences*, vol. 8, no. 1, pp. 1–11, 2018.

[156] F. Obodoeze, F. Ozioko, F. Okoye, C. Mba, T. Ozue, and E. Ofoegbu, “The escalating nigeria national security challenge: Smart objects and internet-of-things to the rescue,” *International journal of Computer Networking and Communication (IJCNAC)*, vol. 1, no. 1, pp. 81–94, 2013.

[157] S. Negi, M. Jayachandran, and S. Upadhyay, “Deep fake: An understanding of fake images and videos,” 2021.

[158] Lionbridge, "Deepfakes a threat to individuals and national security," https://lionbridge.ai/articles/deepfakes-a-threat-to-individuals-and-national-security/, [Online; accessed 6-November-2022].

[159] E. Altinisik and H. T. Sencar, "Camera model identification using container and encoding characteristics of video files," *arXiv preprint arXiv:2201.02949*, 2022.

[160] I. Castillo Camacho and K. Wang, "A comprehensive review of deep-learning-based methods for image forensics," *Journal of Imaging*, vol. 7, no. 4, p. 69, 2021.

[161] I. Shumailov, L. Simon, J. Yan, and R. Anderson, "Hearing your touch: A new acoustic side channel on smartphones," 2019.

[162] W. Dai, M. Qiu, L. Qiu, L. Chen, and A. Wu, "Who moved my data? privacy protection in smartphones," *IEEE Communications Magazine*, vol. 55, no. 1, pp. 20–25, 2017.

[163] Y. Yang, X. Du, and Z. Yang, "Pradroid: Privacy risk assessment for android applications," in *2021 IEEE 5th International Conference on Cryptography, Security and Privacy (CSP)*. IEEE, 2021, pp. 90–95.

[164] S. Small, S. Khalid, P. Dhiman, S. Chan, D. Jackson, A. Doherty, and A. Price, "Impact of reduced sampling rate on accelerometer-based physical activity monitoring and machine learning activity classification," *Journal for the Measurement of Physical Behaviour*, vol. 4, no. 4, pp. 298–310, 2021.

[165] H. Bojinov, Y. Michalevsky, G. Nakibly, and D. Boneh, "Mobile device identification via sensor fingerprinting," *arXiv preprint arXiv:1408.1416*, 2014.

[166] A. Hafeez, K. M. Malik, and H. Malik, "Exploiting frequency response for the identification of microphone using artificial neural networks," in *Audio Engineering Society Conference: 2019 AES International Conference on Audio Forensics*. Audio Engineering Society, 2019.

[167] D. Luo, P. Korus, and J. Huang, "Band energy difference for source attribution in audio forensics," *IEEE Transactions on Information Forensics and Security*, vol. 13, no. 9, pp. 2179–2189, 2018.

[168] C. Hanilci, F. Ertas, T. Ertas, and Ö. Eskidere, "Recognition of brand and models of cell-phones from recorded speech signals," *IEEE Transactions on Information Forensics and Security*, vol. 7, no. 2, pp. 625–634, 2011.

[169] A. E. Dirik, H. T. Sencar, and N. Memon, "Digital single lens reflex camera identification from traces of sensor dust," *IEEE Transactions on Information Forensics and Security*, vol. 3, no. 3, pp. 539–552, 2008.

[170] Y. Zheng, Y. Cao, and C.-H. Chang, "A new event-driven dynamic vision sensor based physical unclonable function for camera authentication in reactive monitoring system," in *2016 IEEE Asian Hardware-Oriented Security and Trust (AsianHOST)*. IEEE, 2016, pp. 1–6.

[171] B. Hadwiger and C. Riess, "The forchheim image database for camera identification in the wild," in *International Conference on Pattern Recognition*. Springer, 2021, pp. 500–515.

[172] C. Chen and M. C. Stamm, "Robust camera model identification using demosaicing residual features," *Multimedia Tools and Applications*, vol. 80, no. 8, pp. 11 365–11 393, 2021.

[173] C. You, H. Zheng, Z. Guo, T. Wang, and X. Wu, "Multiscale content-independent feature fusion network for source camera identification," *Applied Sciences*, vol. 11, no. 15, p. 6752, 2021.

[174] A. Lawgaly, F. Khelifi, and A. Bouridane, "Image sharpening for efficient source camera identification based on sensor pattern noise estimation," in *2013 Fourth International Conference on Emerging Security Technologies*. IEEE, 2013, pp. 113–116.

[175] Q.-T. Phan, G. Boato, and F. G. De Natale, "Accurate and scalable image clustering based on sparse representation of camera fingerprint," *IEEE Transactions on Information Forensics and Security*, vol. 14, no. 7, pp. 1902–1916, 2018.

[176] S. Khan and T. Bianchi, "Fast image clustering based on camera fingerprint ordering," in *2019 IEEE International Conference on Multimedia and Expo (ICME)*. IEEE, 2019, pp. 766–771.

[177] X. Meng, K. Meng, and W. Qiao, "A survey of research on image data sources forensics," in *Proceedings of the 2020 3rd International Conference on Artificial Intelligence and Pattern Recognition*, 2020, pp. 174–179.

[178] S. Gupta, N. Mohan, and M. Kumar, "A study on source device attribution using still images," *Archives of Computational Methods in Engineering*, vol. 28, pp. 2209–2223, 2021.

[179] R. Mayrhofer and H. Gellersen, "Shake well before use: Intuitive and secure pairing of mobile devices," *IEEE Transactions on Mobile Computing*, vol. 8, no. 6, pp. 792–806, 2009.

[180] 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*. ACM, 2012, pp. 161–168.

[181] Y. Liu and J. Niu, “Overlapped-shaking: A local authentication method for mobile applications,” in *Computing, Communications and IT Applications Conference (ComComAp), 2014 IEEE*. IEEE, 2014, pp. 93–97.

[182] D. Bichler, G. Stromberg, M. Huemer, and M. Löw, “Key generation based on acceleration data of shaking processes,” in *International Conference on Ubiquitous Computing*. Springer, 2007, pp. 304–317.

[183] S. M. Bellovin and M. Merritt, “Encrypted key exchange: Password-based protocols secure against dictionary attacks,” in *Research in Security and Privacy, 1992. Proceedings., 1992 IEEE Computer Society Symposium on*. IEEE, 1992, pp. 72–84.

[184] M. Bellare, D. Pointcheval, and P. Rogaway, “Authenticated key exchange secure against dictionary attacks,” in *International conference on the theory and applications of cryptographic techniques*. Springer, 2000, pp. 139–155.

[185] D. P. Jablon, “Extended password key exchange protocols immune to dictionary attack,” in *Proceedings of IEEE 6th Workshop on Enabling Technologies: Infrastructure for Collaborative Enterprises*. IEEE, 1997, pp. 248–255.

[186] P. S. Teh, N. Zhang, A. B. J. Teoh, and K. Chen, “A survey on touch dynamics authentication in mobile devices,” *Computers & Security*, vol. 59, pp. 210–235, 2016.

[187] V. M. Patel, R. Chellappa, D. Chandra, and B. Barbello, “Continuous user authentication on mobile devices: Recent progress and remaining challenges,” *IEEE Signal Processing Magazine*, vol. 33, no. 4, pp. 49–61, 2016.

[188] R. Mayrhofer, J. Fuss, and I. Ion, “UACAP: A Unified Auxiliary Channel Authentication Protocol,” *IEEE Transactions on Mobile Computing*, vol. 12, p. 710–721, April 2013.

[189] M. K. Chong, R. Mayrhofer, and H. Gellersen, “A Survey of User Interaction for Spontaneous Device Association,” *ACM Computing Surveys*, 2014.

[190] M. Fomichev, F. Alvarez, D. Steinmetzer, P. Gardner-Stephen, and M. Hollick, “Survey and Systematization of Secure Device Pairing,” *IEEE Communications Surveys & Tutorials*, September 2017.

[191] A. Kumar, N. Saxena, G. Tsudik, and E. Uzun, “A comparative study of secure device pairing methods,” *Pervasive and Mobile Computing*, vol. 5, no. 6, pp. 734–749, 2009.

[192] D. Kirovski, M. Sinclair, and D. Wilson, “The Martini Synch: Using Accelerometers for Device Pairing,” Technical Report MSR-TR-2007-123, Microsoft Research, Tech. Rep., 2007.

[193] G. Revadigar, C. Javali, W. Xu, A. V. Vasilakos, W. Hu, and S. Jha, “Accelerometer and fuzzy vault-based secure group key generation and sharing protocol for smart wearables,” *IEEE Transactions on Information Forensics and Security*, vol. 12, no. 10, pp. 2467–2482, 2017.

[194] D. Schürmann, A. Brüsch, N. Nguyen, S. Sigg, and L. Wolf, “Moves like Jagger: Exploiting variations in instantaneous gait for spontaneous device pairing,” *Pervasive and Mobile Computing*, vol. 47, pp. 1–12, 2018.

[195] F. Sun, C. Mao, X. Fan, and Y. Li, “Accelerometer-based speed-adaptive gait authentication method for wearable iot devices,” *IEEE Internet of Things Journal*, vol. 6, no. 1, pp. 820–830, 2018.

[196] M. Cao, L. Wang, H. Xu, D. Chen, C. Lou, N. Zhang, Y. Zhu, and Z. Qin, “Sec-d2d: A secure and lightweight d2d communication system with multiple sensors,” *IEEE Access*, vol. 7, pp. 33 759–33 770, 2019.

[197] A. Studer, T. Passaro, and L. Bauer, “Don’t bump, shake on it: The exploitation of a popular accelerometer-based smart phone exchange and its secure replacement,” in *Proceedings of the 27th Annual Computer Security Applications Conference*. ACM, 2011, pp. 333–342.

[198] H. Yüzugüzel, J. Niemi, S. Kiranyaz, M. Gabbouj, and T. Heinz, “ShakeMe: Key generation from shared motion,” in *Computer and Information Technology; Ubiquitous Computing and Communications; Dependable, Autonomic and Secure Computing; Pervasive Intelligence and Computing (CIT/IUCC/DASC/PICOM), 2015 IEEE International Conference on*. IEEE, 2015, pp. 2130–2133.

[199] R. Mayrhofer, “The candidate key protocol for generating secret shared keys from similar sensor data streams,” in *European Workshop on Security in Ad-hoc and Sensor Networks*. Springer, 2007, pp. 1–15.

[200] J. Lester, B. Hannaford, and G. Borriello, ““Are you with me?”–using accelerometers to determine if two devices are carried by the same person,” in *International Conference on Pervasive Computing*. Springer, 2004, pp. 33–50.

[201] Y. Ma, Z. Zhang, S. Chen, Y. Yu, and K. Tang, “A comparative study of aggressive driving behavior recognition algorithms based on vehicle motion data,” *IEEE Access*, vol. 7, pp. 8028–8038, 2018.

[202] N. Ravi, N. Dandekar, P. Mysore, and M. L. Littman, “Activity recognition from accelerometer data,” in *Aaai*, vol. 5, no. 2005, 2005, pp. 1541–1546.

[203] J.-G. Krieg, G. Jakllari, H. Toma, and A.-L. Beylot, "Unlocking the smartphone's sensors for smart city parking," *Pervasive and Mobile Computing*, vol. 43, pp. 78–95, 2018.

[204] Y. Qin, H. Luo, F. Zhao, C. Wang, J. Wang, and Y. Zhang, "Toward transportation mode recognition using deep convolutional and long short-term memory recurrent neural networks," *IEEE Access*, vol. 7, pp. 142 353–142 367, 2019.

[205] L. Wang, H. Gjoreski, M. Ciliberto, S. Mekki, S. Valentin, and D. Roggen, "Enabling reproducible research in sensor-based transportation mode recognition with the sussex-huawei dataset," *IEEE Access*, vol. 7, pp. 10 870–10 891, 2019.

[206] Z. He, J. Cao, X. Liu, and S. Tang, "Who sits where? Infrastructure-free in-vehicle cooperative positioning via smartphones," *Sensors*, vol. 14, no. 7, pp. 11 605–11 628, 2014.

[207] D. Dolev and A. C. chih Yao, "On the security of public key protocols," *IEEE Transactions on Information Theory*, vol. 29, pp. 198–208, 1983.

[208] V. Shoup, *A computational introduction to number theory and algebra*. Cambridge university press, 2009.

[209] K. Cohn-Gordon, C. Cremers, B. Dowling, L. Garratt, and D. Stebila, "A Formal Security Analysis of the Signal Messaging Protocol," Cryptology ePrint Archive, Report 2016/1013, 2016, https://eprint.iacr.org/2016/1013.

[210] F. Hao and S. F. Shahandashti, "The SPEKE protocol revisited," in *International Conference on Research in Security Standardisation*. Springer, 2014, pp. 26–38.

[211] C. Busold, A. Taha, C. Wachsmann, A. Dmitrienko, H. Seudié, M. Sobhani, and A.-R. Sadeghi, "Smart keys for cyber-cars: secure smartphone-based nfc-enabled car immobilizer," in *Proceedings of the third ACM conference on Data and application security and privacy*. ACM, 2013, pp. 233–242.

[212] A. Dmitrienko, A.-R. Sadeghi, S. Tamrakar, and C. Wachsmann, "Smarttokens: Delegable access control with nfc-enabled smartphones," in *International Conference on Trust and Trustworthy Computing*. Springer, 2012, pp. 219–238.

[213] I. Symeonidis, A. Aly, M. A. Mustafa, B. Mennink, S. Dhooghe, and B. Preneel, "Sepcar: A secure and privacy-enhancing protocol for car access provision," in *European Symposium on Research in Computer Security*. Springer, 2017, pp. 475–493.

[214] M. Asim, J. Guajardo, S. S. Kumar, and P. Tuyls, "Physical unclonable functions and their applications to vehicle system security," in *VTC Spring 2009-IEEE 69th Vehicular Technology Conference*. IEEE, 2009, pp. 1–5.

[215] S. Doleva, Ł. Krzywieckib, N. Panwara, and M. Segalc, "Optical puf for non-forwardable vehicle authentication."

[216] J. Yang, Z. Duan, M. Wang, J. Mahmood, Y. Xiao, and Y. Yang, "An authentication mechanism for autonomous vehicle ecu utilizing a novel slice-based puf design," *Journal of New Media*, vol. 2, no. 4, p. 157, 2020.

[217] M. Fomichev, J. Hesse, L. Almon, T. Lippert, J. Han, and M. Hollick, "Fastzip: Faster and more secure zero-interaction pairing," *arXiv preprint arXiv:2106.04907*, 2021.

[218] G. Baldini and I. Amerini, "Smartphones identification through the built-in microphones with convolutional neural network," *IEEE Access*, vol. 7, pp. 158 685–158 696, 2019.

[219] J. Garofolo, "Getting started with the darpa timit cd-rom: An acoustic phonetic continuous speech database, national institute of standards and technology (nist), gaithersburg," *Gaithersburgh, MD, USA*, 1988.

[220] NIST, "Recommendation for the entropy sources used for random bit generation," https://nvlpubs.nist.gov/nistpubs/SpecialPublications/NIST.SP.800-90B.pdf, [Online; accessed 1-July-2022].

[221] H. Han and X. Jiang, "Overcome support vector machine diagnosis overfitting," *Cancer informatics*, vol. 13, pp. CIN–S13 875, 2014.

[222] I. Ahmad, M. Basheri, M. J. Iqbal, and A. Rahim, "Performance comparison of support vector machine, random forest, and extreme learning machine for intrusion detection," *IEEE access*, vol. 6, pp. 33 789–33 795, 2018.

[223] L. Popa, B. Groza, C. Jichici, and P.-S. Murvay, "Ecuprint—physical fingerprinting electronic control units on can buses inside cars and sae j1939 compliant vehicles," *IEEE Transactions on Information Forensics and Security*, vol. 17, pp. 1185–1200, 2022.

[224] T. Kohno, A. Broido, and K. C. Claffy, "Remote physical device fingerprinting," *IEEE Transactions on Dependable and Secure Computing*, vol. 2, no. 2, pp. 93–108, 2005.

[225] R. M. Gerdes, M. Mina, S. F. Russell, and T. E. Daniels, "Physical-layer identification of wired ethernet devices," *IEEE Transactions on Information Forensics and Security*, vol. 7, no. 4, pp. 1339–1353, 2012.

[226] P.-S. Murvay and B. Groza, "Source identification using signal characteristics in controller area networks," *IEEE Signal Processing Letters*, vol. 21, no. 4, pp. 395–399, 2014.

[227] K.-T. Cho and K. G. Shin, "Viden: Attacker identification on in-vehicle networks," in *Proceedings of the 2017 ACM SIGSAC Conference on Computer and Communications Security*, 2017, pp. 1109–1123.

[228] W. Choi, K. Joo, H. J. Jo, M. C. Park, and D. H. Lee, "Voltageids: Low-level communication characteristics for automotive intrusion detection system," *IEEE Transactions on Information Forensics and Security*, vol. 13, no. 8, pp. 2114–2129, 2018.

[229] A. Hafeez, J. Mohan, M. Girdhar, and S. S. Awad, "Machine learning based ecu detection for automotive security," in *2021 17th International Computer Engineering Conference (ICENCO)*, 2021, pp. 73–81.

[230] A. Buscemi, I. Turcanu, G. Castignani, and T. Engel, "On frame fingerprinting and controller area networks security in connected vehicles," in *2022 IEEE 19th Annual Consumer Communications & Networking Conference (CCNC)*. IEEE, 2022, pp. 821–826.

[231] M. Tian, R. Jiang, C. Xing, H. Qu, Q. Lu, and X. Zhou, "Exploiting temperature-varied ecu fingerprints for source identification in in-vehicle network intrusion detection," in *2019 IEEE 38th International Performance Computing and Communications Conference (IPCCC)*. IEEE, 2019, pp. 1–8.

[232] K.-T. Cho and K. G. Shin, "Fingerprinting electronic control units for vehicle intrusion detection." in *USENIX Security Symposium*, vol. 40, 2016, pp. 911–27.

[233] W. Choi, H. J. Jo, S. Woo, J. Y. Chun, J. Park, and D. H. Lee, "Identifying ecus using inimitable characteristics of signals in controller area networks," *IEEE Transactions on Vehicular Technology*, vol. 67, no. 6, pp. 4757–4770, 2018.

[234] M. Kneib and C. Huth, "Scission: Signal characteristic-based sender identification and intrusion detection in automotive networks," in *Proceedings of the 2018 ACM SIGSAC Conference on Computer and Communications Security*, 2018, pp. 787–800.

[235] M. Yu, D. Zhang, Y. Cheng, and M. Wang, "An rfid electronic tag based automatic vehicle identification system for traffic iot applications," in *2011 Chinese Control and Decision Conference (CCDC)*. IEEE, 2011, pp. 4192–4197.

[236] D. R. Crow, S. R. Graham, and B. J. Borghetti, "Fingerprinting vehicles with can bus data samples," in *ICCWS 2020 15th International Conference on Cyber Warfare and Security*. Academic Conferences and publishing limited, 2020, p. 110.

[237] A. Hafeez, K. Rehman, and H. Malik, "State of the art survey on comparison of physical fingerprinting-based intrusion detection techniques for in-vehicle security," SAE Technical Paper, Tech. Rep., 2020.

[238] S. Bellaire, M. Bayer, A. Hafeez, R. U. D. Refat, and H. Malik, "Fingerprinting ecus to implement vehicular security for passenger safety using machine learning techniques," in *Intelligent Systems and Applications: Proceedings of the 2022 Intelligent Systems Conference (IntelliSys) Volume 3*. Springer, 2022, pp. 16–32.

[239] Mathworks, "Choose classifier options," https://www.mathworks.com/help/stats/choose-a-classifier.html, [Online; accessed 1-April-2022].