Developing Test Bed for Screen Gleaning Attack on Smartphones for OTP Retrieval

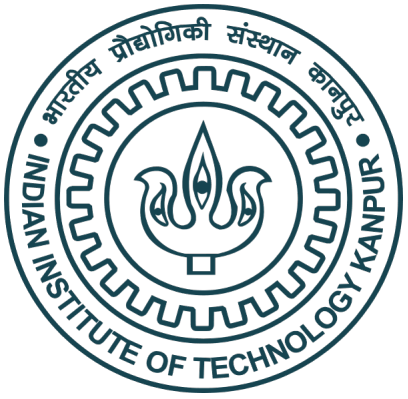
A thesis submitted in fulfilment of the requirements

for the degree of Master of Technology

by

Kakumanu Vamsee Krishna

22111065



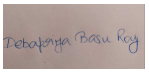
DEPARTMENT OF COMPUTER SCIENCE AND ENGINEERING

INDIAN INSTITUTE OF TECHNOLOGY KANPUR

July 2024

Certificate

It is certified that the work contained in this thesis entitled **"Developing Test Bed for Screen Gleaning Attack on Smartphones for OTP Retrieval"** by **Kakumanu Vamsee Krishna** has been carried out under my supervision and that it has not been submitted elsewhere for a degree. This thesis was submitted in July 2024.



Asst. Prof. Urbi Chatterjee
Assistant Professor
CSE Department
Indian Institute of Technology Kanpur

Asst. Prof. Debapriya Basu Roy
Assistant Professor
CSE Department
Indian Institute of Technology Kanpur

Declaration

This is to certify that the thesis titled **"Developing Test Bed for Screen Gleaning Attack on Smartphones for OTP Retrieval"** has been authored by me. It presents the research conducted by me under the supervision of **Asst. Prof. Urbi Chatterjee** and co-supervison of **Asst. Prof. Debapriya Basu Roy**.

To the best of my knowledge, it is an original work, both in terms of research content and narrative, and has not been submitted elsewhere, in part or in full, for a degree. Further, due credit has been attributed to the relevant state-of-the-art and collaborations with appropriate citations and acknowledgments, in line with established norms and practices.

Kakumanu Vamsee Krishna
Roll No. 22111065 (M.Tech)
CSE Department
Indian Institute of Technology Kanpur

Abstract

Name of the student: **Kakumanu Vamsee Krishna** Roll No: **22111065**

Degree for which submitted: **MTech** Department: **CSE Department**

Thesis title: **Developing Test Bed for Screen Gleaning Attack on Smartphones for OTP Retrieval**

Thesis supervisors: **Asst. Prof. Urbi Chatterjee** and **Asst. Prof. Debapriya Basu Roy**

Month and year of thesis submission: **July 2024**

Mobile display screens emit unintentional electromagnetic (EM) leakages that can be exploited to reconstruct visual information, such as One-Time Passwords (OTPs) and security codes. This research introduces the concept of "screen gleaning," a TEMPEST attack that leverages these EM emissions to reveal sensitive information without a direct line of sight to the device. By using an antenna and software-defined radio (SDR), the emitted signals are captured and transformed into gray-scale images, or "emages," which can be interpreted using machine learning techniques.

The core challenge of screen gleaning lies in the typically poor quality of these emages, which are often indecipherable to the human eye. To address this, deep learning classifiers are employed to decode the screen content. This thesis details the setup and execution of the screen gleaning attack, successfully generating emages, albeit of low quality, and explores the existing literature to propose measures for enhancing the attack's effectiveness.

The study involves the collection and analysis of EM signals, evaluating metrics such as Signal-to-Noise Ratio (SNR), Normalized Inter-Class Variance (NICV), and Test Vector

Leakage Assessment (TVLA) to better understand the signal characteristics. Additionally, supervised machine learning classification is performed on the Fast Fourier Transform (FFT) versions of these signals, demonstrating the capability of machine learning algorithms to predict screen content based on the EM traces. The potential use of Recurrent Neural Networks (RNNs) for classifying time-domain signals is also proposed, suggesting a future direction for real-time screen content prediction.

This thesis not only highlights the feasibility of screen gleaning attacks but also provides a comprehensive analysis of EM signal characteristics and the application of machine learning to enhance attack accuracy. The findings underscore the need for robust countermeasures to protect against such vulnerabilities, ensuring the security and privacy of sensitive information displayed on mobile devices.

Acknowledgements

I would like to express my sincere gratitude to my esteemed supervisors, Dr. Urbi Chaterjee and Dr. Debapriya Basu Roy, for their invaluable guidance and unwavering support throughout my thesis journey. Their expertise, encouragement, and insightful feedback have been instrumental in shaping and refining the enthusiastic and intricate scope of this research, which delves into side-channel attacks, video-signal processing, and machine learning.

I am deeply appreciative of the provision of state-of-the-art resources and equipment essential for conducting rigorous research. These resources have significantly enhanced the quality and depth of my investigations, allowing for comprehensive exploration and experimentation within the complex domains addressed in this thesis.

Furthermore, I extend my heartfelt thanks for the continuous mentorship and encouragement I received during the tenure of my thesis. Their mentorship not only facilitated academic growth but also nurtured my development as a researcher, enabling me to navigate and contribute meaningfully to the broader research areas encompassed by this work.

I am also grateful to Indian Institute of Technology Kanpur for fostering an environment conducive to intellectual inquiry and scholarly pursuits, which has been instrumental in my academic journey.

Contents

List of Figures

List of Tables

Abbreviations

AMOLED	Active Matrix Organic Light Emitting Diode
EM	Electro-Magnetic
FFT	Fast Fourier Transform
FPGA	Field Programmable Gate Array
NICV	Normalized Inter-Class Variance
OTP	One-Time Passwords
RNN	Recurrent Neural Networks
SDR	Software Defined Radio
SNR	Signal-to-Noise Ratio
SVM	Support Vector Machine
TVLA	Test Vector Leakage Assessment
USRP	Universal Software Radio Peripheral
UHD	USRP Hardware Driver

For the Lord gives wisdom; from his mouth come knowledge and understanding. - Proverbs 2:6(NIV)

Chapter 1

Introduction

In April 1820, Danish physicist Hans Christian Ørsted observed that a compass needle deflected when placed near a wire carrying an electric current, demonstrating that changing electric currents create magnetic fields and vice versa. This discovery eventually led to the development of the electrical telegraph for long-distance communication. However, the unintended effects of this phenomenon were largely ignored for decades. The British army noted crosstalk between telephone wires during the Nile and Suakin expedition in 1884-85. The first exploitation of this effect was reported in 1914, involving earth leakage from telephone wires causing significant crosstalk, leading to the establishment of listening posts for intercepting enemy messages. In 1915, the use of valve amplifiers extended the range of these listening activities.

In 1972, the US National Security Agency (NSA) conducted classified research under the codename TEMPEST [1], which was partially declassified in 2007. The research revealed that unwanted emanations from a Bell-telephone mixing device used for encryption allowed attackers to reconstruct the original plaintext from a distance of about 80 feet, with a success rate of approximately 75%.

1.1 Related works

The first unclassified technical analysis of electromagnetic emanations from computer monitors was published by Wim van Eck in 1985. His work, titled "Electromagnetic Radiation from Video Display Units: An Eavesdropping Risk?[2]" demonstrated that contemporary monitors emitted signals similar to black and white TV sets, which could be exploited for

eavesdropping using a modified TV receiver. Although his attack was viable, it required specialized knowledge to modify the TV receiver and constant manual adjustments, making it impractical. Additionally, the rapid evolution of technology rendered his methods obsolete for modern monitors.

In 2003, Markus Kuhn improved upon van Eck's work with his report "Compromising Emanations: Eavesdropping Risks of Computer Displays."[3] Kuhn analyzed the properties of emitted waves and the emitting circuitry, constructing a real-time monitoring system using an FPGA board, a specialized wideband AM radio receiver, and a VGA video monitor. Despite advancements, his system required expensive, export-restricted equipment and manual synchronization. Kuhn also explored the modulation of hidden messages into signals and methods to defend against such attacks.

In 2013, Elibol, Sarac, and Erer [4] demonstrated a more affordable and mobile system capable of eavesdropping from a distance of 50 meters in real-time. Their implementation used an algorithm to determine the horizontal synchronization frequency of the target display without relying on synchronization pulses. However, their system still required prior knowledge of the target system and remained expensive, costing tens of thousands of pounds. Subsequent research by Ghani et al [5]. focused on emissions from LCD touch screen displays of handheld devices.

1.2 Motivation

TEMPEST attacks have the potential to target any communication device, whether mechanical or electrical, provided that a third party can intercept the communication signal through unconventional means. Identifying these means and the root cause of the communication leakage can be complex, as the leaks are often unintentional and arise from the inherent properties of the communication signal.

This research focuses on electronic personal mobile devices, which inadvertently leak analog video signals as electromagnetic emanations. This leakage occurs through the ribbon cable that connects the graphical processing unit (GPU) to the display screen. The ribbon cable, which is responsible for transmitting electrical information, unintentionally acts as an antenna, emitting the video signal into the surrounding electromagnetic spectrum. This leakage can be exacerbated by an impedance mismatch between the cable and the motherboard or display socket. Such impedance differences may result from dimensional mismatches between the socket and the ribbon cable. To mitigate potential interference

between adjacent connectors, the connecting cable is often designed to be smaller than the socket. Consequently, phones from different manufacturers emit signals with varying strengths due to these differing offsets.

Screen Gleaning[6] a low-cost Side Channel Analysis attack that captures the electromagnetic emanation leaked at the mobile screen's ribbon cable. The captured EM signal (emanation) can be visualized as something called an "emage"- a gray-scale image generated by a software called "TempestSDR"[7], but it is often not interpretable by the naked eye. To address this challenge, we propose a machine learning-based approach to process the emage and recover secret/target information, such as a security code, text, etc. We aim to demonstrate and assess the security threat posed by such emanations in different scenarios and provide an implementation of our attack for further testing.

1.3 Organization of the Thesis

The remainder of this thesis is organized as follows: Chapter 2 provides an extensive review of the literature on Tempest attacks targeting smartphones, with a particular focus on screen gleaning techniques. This chapter covers the evolution of Tempest attacks, their execution, and their implications for security and privacy, and identifies critical areas for future research and development in the field. Chapter 3 presents an in-depth analysis of electromagnetic leakages from mobile screen displays, utilizing an oscilloscope to measure various metrics including SNR, NICV, TVLA, and applying machine learning techniques for classification of leakage traces based on screen content. Chapter 4 details the practical implementation of the screen gleaning attack, including the setup installation, image acquisition process, challenges encountered, and the solutions devised to overcome these obstacles. Finally, Chapter 5 concludes the thesis by summarizing the key findings and results from the research, and proposing potential enhancements and future directions for continued exploration and experimentation in the domain of smartphone security.

Chapter 2

TEMPEST attack on Smartphones

This detailed chapter provides a comprehensive exploration of screen gleaning attack [6] as a TEMPEST attack, covering its evolution, execution, implications for security and privacy, and outlining critical areas for future research and development in cybersecurity.

2.1 Introduction

The field of cybersecurity has witnessed a significant evolution with the emergence of novel attack vectors that exploit electromagnetic emanations from electronic devices. Among these, screen gleaning has emerged as a sophisticated form of TEMPEST attack, capable of capturing and reconstructing visual content displayed on mobile device screens through the interception of electromagnetic signals. This chapter delves into the intricacies of screen gleaning, exploring its execution, implications for security and privacy, and outlining future research directions in cybersecurity.

TEMPEST attacks traditionally involve the covert interception and analysis of unintended electromagnetic emissions from electronic devices. These emissions, which can include signals from video displays and other electronic components inadvertently leak information that can be exploited by attackers. Screen gleaning specifically targets the electromagnetic emanations emitted by mobile device screens, aiming to reconstruct visual content using specialized equipment and techniques.

The evolution of screen gleaning as a cyber threat can be attributed to advancements in antenna technology and software-defined radios (SDRs). These technological advancements have significantly enhanced the sensitivity and accuracy with which electromagnetic

signals can be intercepted and processed. Initially demonstrated on devices such as the Apple iPhone 6s and Android Honor 6X, screen gleaning highlights the vulnerability of modern mobile displays to electromagnetic eavesdropping techniques.

2.2 Execution of Screen Gleaning

The execution of a screen gleaning attack involves several intricate steps, each crucial for capturing and interpreting electromagnetic signals emitted by mobile device screens:

- Attackers deploy a passive magnetic probe antenna (Langer RF-R 400) in close proximity to the target mobile device, typically within a few centimeters, to intercept electromagnetic emissions. These emissions include signals generated by the display when rendering text messages, security codes, or other visual content.
- Intercepted signals are amplified using low-noise amplifiers (Minicircuits ZKL-2) to enhance weak signals for further processing. The amplified signals are then digitized using an Software Defined Radio (USRP X310) equipped with a high-resolution sampling daughter-board (UBX-160) as in 2.1, ensuring accurate capture of the electromagnetic spectrum.
- Specialized software tools, such as TempestSDR, are employed to reconstruct the digitized signals into coherent visual representations known as "emages." These emages resemble the original visual content displayed on the mobile device screen, enabling attackers to decipher sensitive information.
- Emages containing sensitive information, such as security codes or textual messages, are classified using machine learning models as in 2.2 because the content in emages reconstructed from mobile device emissions is visually uninterpretable as in 2.3. These models are trained on datasets of known visual patterns (e.g., digits 0-9), allowing attackers to automatically identify and interpret intercepted content.

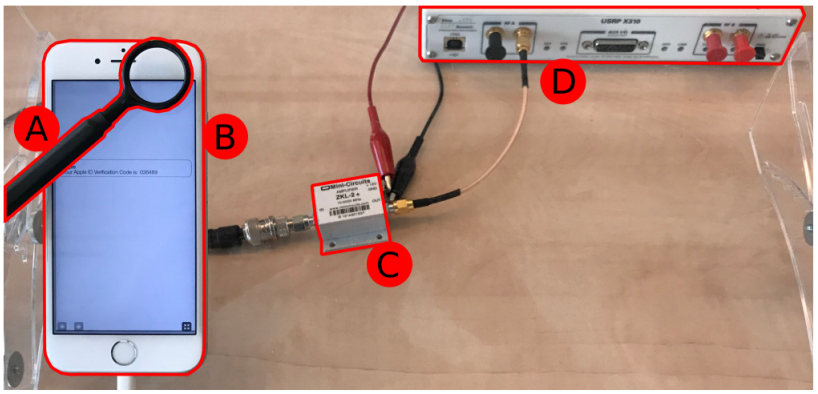


FIGURE 2.1: Screen Gleaning Attack Setup.

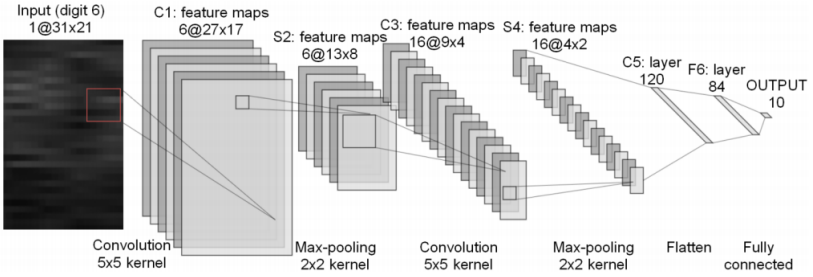


FIGURE 2.2: Modified LeNet5 Architecture.

FIGURE 2.3: Emage of a 6-digit OTP

2.3 Challenges and Consequences for Security and Privacy

The implications of screen gleaning for security and privacy are multifaceted and profound:

- Variability in Signal Strength: Different device models and environmental conditions can influence the strength and clarity of electromagnetic emissions, impacting the effectiveness of screen gleaning attacks.
- Privacy Breach: Screen gleaning enables attackers to access and decode sensitive information, such as security codes or private messages, displayed on mobile device

screens. This constitutes a severe breach of user privacy, compromising confidential data.

- Legal and Ethical Concerns: The practice of screen gleaning raises ethical dilemmas regarding privacy invasion and regulatory challenges in addressing sophisticated cyber threats. Regulatory frameworks may lag behind technological advancements, posing challenges in enforcing adequate protection measures.

2.4 Conclusion and Future Directions

In conclusion, screen gleaning represents a significant advancement in TEMPEST attacks, leveraging technological advancements in antenna technology and signal processing to breach the privacy of mobile device users. Addressing the challenges posed by screen gleaning requires collaborative efforts from academia, industry, and regulatory bodies to develop robust defense strategies and uphold privacy standards in an increasingly digital world.

While significant progress has been made in demonstrating the feasibility of screen gleaning attacks, several challenges and avenues for future exploration remain:

- Generalization and Reconstruction Scenarios: Advancing from discrimination scenarios (e.g., recognizing security codes) to more complex scenarios such as content prediction poses new technical and ethical challenges. These scenarios require enhanced signal processing capabilities and robust machine learning models.
- Countermeasures and Mitigation Strategies: Developing effective countermeasures against screen gleaning is crucial to mitigate its impact. Potential strategies include implementing electromagnetic shielding technologies, obfuscating signal emissions, or enhancing device-level security to prevent unauthorized access to visual content.

Looking ahead, future research into screen gleaning will focus on refining attack methodologies, exploring new device types susceptible to electromagnetic emissions, and developing comprehensive defense strategies. By advancing our understanding of screen gleaning and its implications, the cybersecurity community can proactively mitigate risks and enhance the resilience of mobile devices against sophisticated TEMPEST attacks.

Chapter 3

TEMPEST Attack on Smartphones using Oscilloscope

This chapter presents our analysis of electromagnetic leakages emitted from mobile screen displays, focusing on metrics such as SNR, NICV, TVLA, and the Machine Learning classification of leakage traces based on the screen's content.

3.1 Foundation of Attack model

The attack setup is depicted in Figure 3.1. The setup comprises several components that are labeled as follows:

- The antenna utilized is a passive Langer LF-R 400 magnetic probe.
- The target device being examined is a Realme XT smartphone.
- The signal captured by the probe is amplified using Langer PA303 pre-amplifier.
- Tektronix-MSO64B Mixed Signal Oscilloscope, is employed to receive and store the amplified signal as waveforms.

In the setup depicted in Figure 3.1, the EM signals are initially tapped using a Langer probe. Thereafter, the signal undergoes amplification before being sent to Oscilloscope. we have built the setup to obtain foundational results using the oscilloscope. Our Preliminary setup (as shown in Figure 3.1) consists of the target mobile device (Realme XT housing

AMOLED display) being probed with an EM probe (Langer LF-R 400). Also, these probes are connected to a power amplifier, which is connected to the oscilloscope (Tektronix-MSO64B) to sample EM traces.

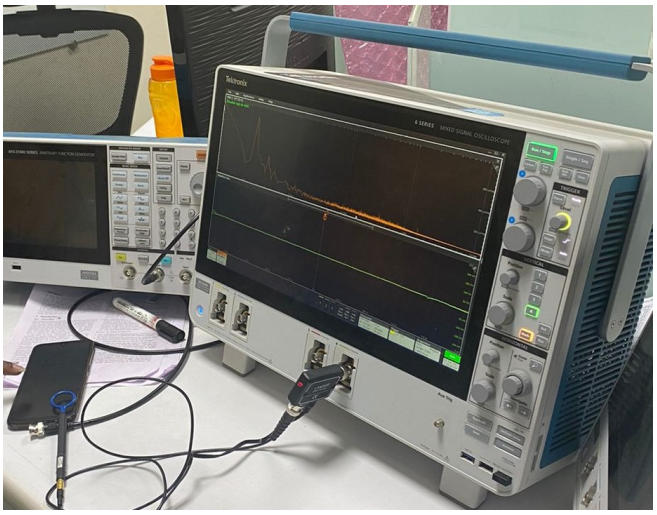


FIGURE 3.1: Oscilloscope Attack Setup.

This setup aims to sample electromagnetic (EM) traces and then identify significant patterns/correlations. For this purpose, we rely on the calculation of statistical parameters such as SNR (Signal to Noise Ratio), NICV (Normalized Inter Class Variance), and TVLA (Test Vector Leakage Assessment). Later, these parameters are utilized to assist us in the evaluation of signal quality, determination of the leakage spectrum, and assessment of differentiability factors.

3.1.1 Attack motivation: Android app to display Security Code

Our ultimate objective is to retrieve confidential information (such as security codes, passwords, or messages) that are displayed on the screen of the target mobile phone, even without a direct line of sight in the visible spectrum. To achieve this, we collect electromagnetic traces emitted by the target device, which represent the visual content displayed on the screen. In addition to the preliminary setup, we have developed an Android application.

In this mobile application, any number—or in our case, a random 6-digit OTP—will be displayed on the mobile phone screen upon manual input. By performing a long tap, the input number can be placed in one of the nine designated positions on the screen, as shown in Figure 3.2.

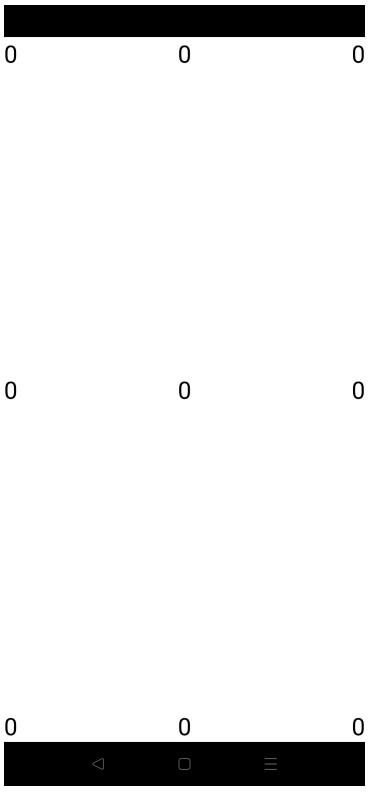


FIGURE 3.2: Android app displaying 6-digit code.

3.2 Detection & Analysis of Leakage of Electromagnetic Radiation

In order to identify the specific location of the leakage in the mobile device, it was necessary to determine the point where the display cable of the screen is connected to the phone's board, as indicated in the research paper by Liu et al [6], specifically in section IV.A.1. To accomplish this, we referred to various disassembly resources available online, allowing us to ascertain the location of the display cord. Figure 3.3 highlights the enclosed (red) area, which encompasses the connection between the display cord and the board of the mobile phone. By conducting probing tests with the screen both on and off, and observing significant waveform disparities between the probing results, we can confidently confirm that our identified location indeed corresponds to the area where the mobile phone is experiencing leakage.

FIGURE 3.3: Architecture of Realme XT.

Having identified the appropriate probing location, we proceeded to capture samples (traces) utilizing our initial setup. These samples were acquired using trigger activation to ensure synchronization among the traces in both the Time Domain and Frequency Domain. Through meticulous analysis of these electromagnetic (EM) traces, we are able to extract crucial parameters including Signal-to-Noise Ratio (SNR), Normalized Inter-Class Variance (NICV), Test Vector Leakage Assessment (TVLA), and other pertinent metrics.

3.2.1 Signal to Noise Ratio - SNR

To assess the quality of the leakage signal, we have calculated the signal-to-noise Ratio (SNR) on the traces sampled by our oscilloscope. These traces were taken when mobile was displaying our Android app's content.

The signal-to-noise Ratio is computed as follows:

- Use the FFT trace or perform the FFT on the windowed time-domain signal trace to obtain the frequency-domain representation of the signal.
- Calculate the power spectrum density (PSD) by taking the square of the absolute value of the FFT result and then normalizing it by recording length and sampling frequency.
- Calculate the average power of the signal by averaging the power spectrum values within the frequency bins.

- Calculate the average power of the noise by summing the power spectrum values within the noise frequency bins of the trace taken for noise power.
- Calculate the SNR using the formula: SNR $= 10 * log_{10}(\frac{\text{signal power}}{\text{noise power}})$.

The set of samples used to calculate SNR are taken in two different probe positions, so we accordingly have two sets. The computed SNR values can be found in Table 3.1.

TABLE 3.1: SNR

Set. No.	SNR-Max	SNR-Mean
1	83.6366 dB	5.4404 dB
2	86.3205 dB	5.5581 dB
Noise Power used for both cases	-139.6745 dB	-

The noise power is determined by analyzing the traces obtained when the phone display is inactive or in a locked/sleep state. This ensures consistency in the noise power calculation, which is subsequently utilized to evaluate the Signal-to-Noise Ratio (SNR) in both scenarios. It is noteworthy that our obtained SNR values exhibit improvement compared to the values reported in the study by Liu et al. (2020).

Moreover, our anticipated SNR values surpass those reported in [6]. Specifically, for a mobile phone model sold in India, the SNR values are recorded as 83.6 dB and 86.32 dB, as depicted in Table 3.1.

3.2.2 Normalized Inter-Class Variance - NICV

Normalized Inter-Class Variance is calculated to know the region where the Inter-Class Variance is high, i.e., the region (frequency) with more leakage.

Instead of using the complex approach of creating classes of samples based on the 6-digit code displayed on our Android application, we have decided to sample traces in specific classes. These classes include instances where the mobile phone displays Red, Violet, and Black colors individually. Additionally, we have sampled traces when the screen is locked, i.e., displays a blank screen. This approach simplifies the classification process by categorizing the data based on the displayed colors and screen status, with 50 samples per class. Clearly,

$$NICV = \frac{Var(E(\frac{Y}{X}))}{Var(Y)}$$

where X is the given Colour & Y is the value of trace at a timestamp or frequency.

In our case, the numerator in $NICV$ is the Variance of the mean of traces for a given Color, and the denominator is the variance of all traces at all timestamps and frequencies.

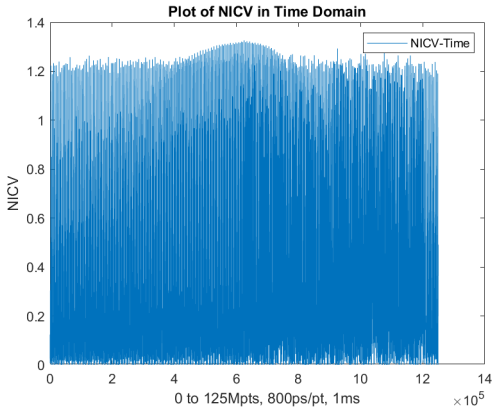


FIGURE 3.4: Plots of NICV in both Time Domain.

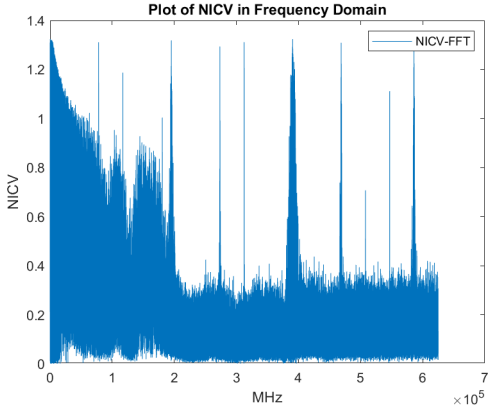


FIGURE 3.5: Plots of NICV in both Frequency Domain.

TABLE 3.2: NICV

NICV	NICV Time-Domain	NICV Frequency-Domain
NICV-Max	1.3263	1.3245
NICV-Min	1.9564e-05	1.6282e-04

As NICV values should be normalized i.e., the values should be restricted to lie between 0 & 1. The Max values in Table 3.2 are anomalies, if we calculate NICV over a large number of samples instead of 50 per type, this kind of anomalous values can be resolved.

3.2.3 Test Vector Leakage Assessment - TVLA

The Test Vector Leakage Assessment (TVLA) is computed by comparing samples from two distinct sets to determine their distinguishability. If the calculated TVLA value is equal to or greater than 4.5, it indicates that the two sets of samples can be considered distinguishable.

The first set corresponds to traces where the intermediate variable is equal to some predefined fixed value (f) and the second set corresponds to traces where the intermediate variable is not equal to some predefined fixed value (r). The TVLA can be computed as follows:

$\text{TVLA} = \frac{\mu_r - \mu_f}{\sqrt{\frac{\sigma_r^2}{n_r} + \frac{\sigma_f^2}{n_f}}}$, where,

- μ_r is the mean of the values of a set per timestamp and frequency, where the intermediate variable is not equal to some predefined fixed value.
- μ_f is the mean of the values of a set per timestamp and frequency, where the intermediate variable is equal to some predefined fixed value.
- σ_r is the variance of the values of a set per timestamp and frequency, where the intermediate variable is not equal to some predefined fixed value.
- σ_f is the variance of the values of a set per timestamp and frequency, where the intermediate variable is equal to some predefined fixed value.
- n_r is no of samples in a set where the intermediate variable is not equal to some predefined fixed value.
- n_f is no of samples in a set where the intermediate variable is equal to some predefined fixed value.

In our case both n_r & n_f equals to 50.

So on the color traces we took earlier in the case of NICV, we calculated TVLA with the first set being of color (f) and the second set being of Blank(screen lock).

Results are as in Table 3.3.

TABLE 3.3: TVLA

TVLA-Max	Red	Violet	Black
Blank-Time	210.8755	161.5112	239.0945
Blank-Frequency	19.4900	19.2562	20.4981

As the Values of TVLA in all cases are ≥ 4.5, we conclude that our color classes (Red, Violet, and Black) traces are very much distinguishable from Blank traces.

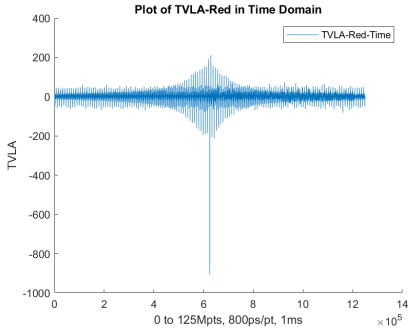


FIGURE 3.6: Plot of TVLA-Red & Blank in Time Domain.

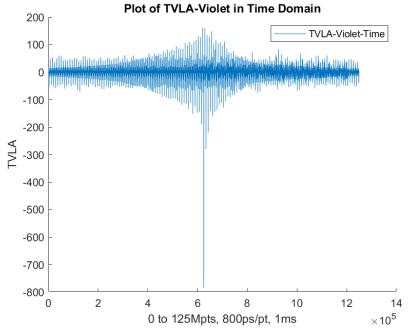


FIGURE 3.7: Plot of TVLA-Violet & Blank in Time Domain.

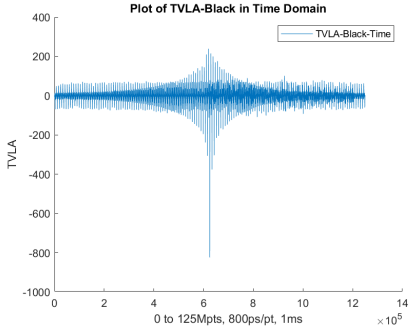


FIGURE 3.8: Plot of TVLA-Black & Blank in Time Domain.

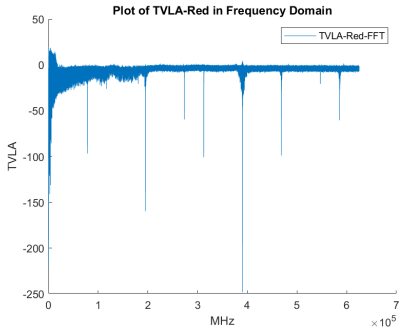


FIGURE 3.9: Plot of TVLA-Red & Blank in Frequency Domain.

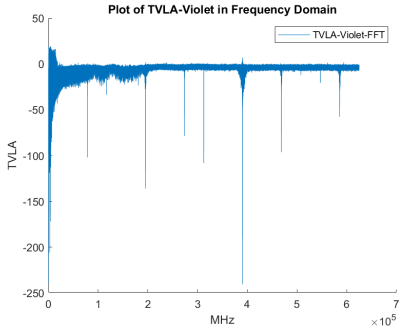


FIGURE 3.10: Plot of TVLA-Violet & Blank in Frequency Domain.

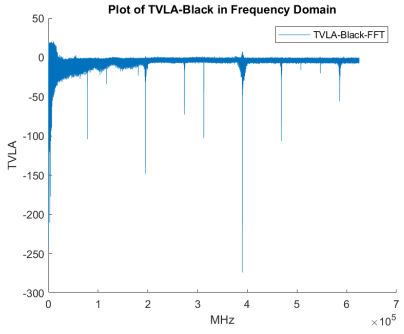


FIGURE 3.11: Plot of TVLA-Black & Blank in Frequency Domain.

Significant fluctuations observed in the plots of TVLA (Test-Vector Leakage Assessment) can be effectively mitigated by employing a larger sample size as opposed to the current 50 samples.

Based on the analysis of Figure 3.5 and Figures 3.9, 3.10, and 3.11, it is evident that leakage occurs predominantly at the 4th harmonic, which is approximately centered around the frequency of 390.626 MHz. It is important to note that this frequency may exhibit slight variations when analyzed using an extensive dataset in the future. However, we anticipate

that any such variations would remain in close proximity to the 390.626 MHz center frequency.

3.3 Classification of samples based on screen's content

To comprehend the characteristics of the acquired frequency domain samples (FFT traces), our focus centered on classifying traces based on the content displayed on the mobile phone screen. We opted to sample traces in specific classes, including instances where the screen exhibited Red, Violet, and Black colors individually. Moreover, we sampled traces when the screen was locked, presenting a blank screen, thereby establishing four classes for classification.

In an effort to investigate the display of Secret Key (OTP), we developed an Android app that sequentially displays digits. Subsequently, we expanded our sampling to include traces of four additional classes, featuring instances where the mobile phone displayed the digits 0, 1, 5, and 7 individually at the center within the app.

As our exploration progressed, we enhanced the Android app to showcase digits at multiple positions (9 in total) on the screen, encompassing the four corners, center, and midpoints of all four corners of the app. Traces of a specific digit were sampled with equal weights for each position it occupied. These meticulous sampling efforts allow for a comprehensive analysis of the observations, enabling us to draw meaningful inferences from the collected data. Now, let's examine these categories one by one.

3.3.1 Classification of samples according to the color displayed on the entire screen

The traces we gathered comprise 625,001 features, spanning the frequency range from 0 to 625 MHz, adhering to the Nyquist range. Each data point corresponds to a 1 KHz bandwidth. At the outset of our classification process, we employed SVM, a supervised learning algorithm. This choice was driven by the sampling, labeling, training, and testing workflow inherent in supervised learning. SVM was particularly chosen due to the substantial feature length of 625001, as it excels in scenarios where the number of features significantly surpasses the number of samples, demonstrating efficient performance in such scenarios. We applied SVM classifier (Multi-class Classification) to 500 samples from each of the four color classes, totaling 2000 samples for training and testing. The data was split

with a 70:30 train-test ratio, and with 625,001 features, we achieved an overall accuracy of **98%**. As the classification process became time-consuming and computationally expensive, especially when making predictions on a dataset comprising 2000 rows and 625,001 columns, we aimed to investigate the impact of feature reduction on classification accuracy. To achieve this, we generated smaller datasets with initial feature counts of 5000, 10000, 15000, 20000, 25000, 50000, and 100,000 columns and 500 rows for each class of color, extracted from the larger dataset of 625,001 columns. We then trained the same SVM classifier (Multi-class Classification) using the respective feature lengths and recorded the achieved accuracies, as outlined in Table 3.4 and Figure 3.12.

TABLE 3.4: Color - Accuracy outcomes corresponding to different feature lengths.

Dataset Size per class	5000 columns	10000 columns	15000 columns	20000 columns	25000 columns	50000 columns	100000 columns
500 rows	99.83%	**100%**	99.67%	99.67%	99.67%	99.33%	99.17%

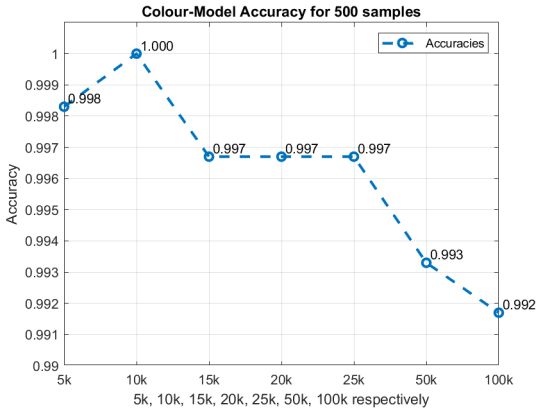


FIGURE 3.12: Plot of Color - Accuracy outcomes corresponding to different feature lengths.

Based on these observations, we can deduce that the initial 10,000 columns of the acquired data suffice for successful sample classification, achieving **100%** accuracy in discerning the displayed color on the entire screen at the time of acquisition. The observed 100% accuracy may be attributed to factors such as probe positioning and noise, contributing to variance in class data and thereby improving classification. To further validate this result and compare it with accuracies obtained at other feature counts, we conducted another round of sampling. This time, we collected 2000 samples per class, totaling 8000 samples, but restricted the features to the initial 10,000 of the acquired dataset of 625001 features. We utilized three distinct supervised learning algorithms—SVM, Decision Tree, and Random Forest—on a dataset comprising 10,000 features and 8,000 samples (rows). The resulting accuracies are presented in the accompanying Table 3.5.

TABLE 3.5: Color - Accuracy results from various classifiers using a feature length of 10,000.

Dataset Size per class	SVM	Decision Tree	Random Forest
2000 rows	**99.67%**	98.45%	99.20%

The three classifiers provided predictions with accuracies closely aligned, and the SVM model exhibited a slightly higher accuracy of **99.67%**. This result substantiates that the previously achieved 100% accuracy was not a chance occurrence. Consequently, it can be asserted that the initial 10,000 features are adequate for classifying samples based on the content of the screen's color.

3.3.2 Classification of samples according to the single-digit displayed at the center of the screen

Initially, we collected 500 traces corresponding to the display of digits 0, 1, 5, and 7 (four classes) individually in the Android app we developed. Subsequently, we employed SVM for multi-class classification, training the model on a dataset of 2000 samples and 625,001 features, resulting in an accuracy of **67.83%**. To further investigate the impact of varying feature lengths, we generated smaller datasets with initial feature counts of 5000, 10000, 15000, 20000, 25000, 50000, and 100,000 columns. Each dataset consisted of 500 rows for each color class, extracted from the larger dataset of 625,001 columns. The same SVM classifier (Multi-class Classification) was then trained on these datasets using the corresponding feature lengths, and the achieved accuracies are as in Table 3.6 and Figure 3.13.

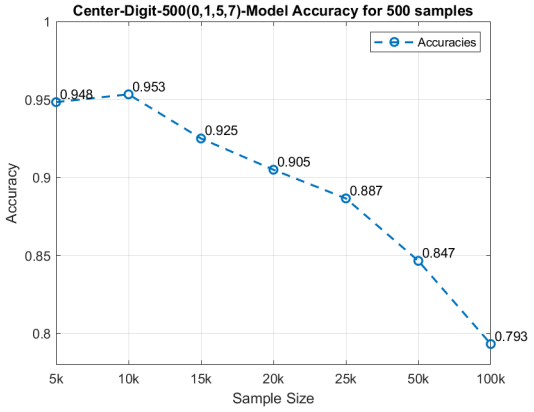


FIGURE 3.13: Plot of Single Digit - Accuracy outcomes corresponding to different feature lengths and 500 rows per class.

TABLE 3.6: Single Digit - Accuracy outcomes corresponding to different feature lengths.

Dataset Size per class	5000 columns	10000 columns	15000 columns	20000 columns	25000 columns	50000 columns	100000 columns
500 rows	94.83%	**95.33%**	92.5%	90.5%	88.66%	84.66%	79.33%

Once more, the model trained on 10,000 columns exhibited the highest accuracy. In this iteration, we sampled traces with 2000 samples per class, generating datasets of 625,001 features and another with the initial 10,000 features. We conducted both Binary Classification (0 or not 0) and Multi-class Classification using three distinct supervised learning algorithms—SVM, Decision Tree, and Random Forest—on a dataset containing 10,000 features and 8,000 samples (rows). The results for Multi-class Classification and Binary Classification are presented in Table 3.7 and Table 3.8, respectively.
According to Table 3.8, the Random Forest classifier demonstrated higher accuracies in predicting 0, 1, and 7 while the Decision Tree classifier in predicting 5.

TABLE 3.7: Single Digit (0, 1, 5, 7) - Multi class - Accuracy results from various classifiers using a feature length of 10,000.

Dataset Size per class (0,1,5,7)	SVM	Decision Tree	Random Forest
2000 rows	71.13%	79.08%	**79.45%**

TABLE 3.8: Single Digit (0, 1, 5, 7) - Binary - Accuracy results from various classifiers using a feature length of 10,000.

2000 rows per class (0,1,5,7)	0	1	5	7
SVM	92.83%	76.91%	73.66%	83.75%
Decision Tree	91.91%	82.33%	**83.50%**	88.00%
Random Forest	**93.83%**	**83.25%**	77.91%	**91.50%**

We conducted additional sampling, creating two sets of 2000 samples per class (6000 in total) for digits 2, 3, and 4 in one set, and digits 6, 8, and 9 in the other. Similar to the previous approach, we generated datasets with a feature length of 10,000 for both the 2,3,4 and 6,8,9 sets. Subsequently, we trained both Binary Classification (0 or not 0) and Multi-class Classification models using three distinct supervised learning algorithms—SVM, Decision Tree, and Random Forest—on datasets comprising 10,000 features and 6,000 samples (rows). The results for Multi-class Classification and Binary Classification for sets of 2,3,4 and 6,8,9 are presented in Table 3.9 & Table 3.10 and Table 3.11 & Table 3.12 respectively.

TABLE 3.9: Single Digit (2, 3, 4) - Multi class - Accuracy results from various classifiers using a feature length of 10,000.

Dataset Size per class (2,3,4)	SVM	Decision Tree	Random Forest
2000 rows	88.45%	**93.56%**	90.33%

TABLE 3.10: Single Digit (2, 3, 4) - Binary - Accuracy results from various classifiers using a feature length of 10,000.

2000 rows per class (2,3,4)	2	3	4
SVM	94.34%	87.83%	93.91%
Decision Tree	**96.58%**	**93.33%**	**97.83**
Random Forest	95.17%	88.42%	97.34%

TABLE 3.11: Single Digit (6, 8, 9) - Multi class - Accuracy results from various classifiers using a feature length of 10,000.

Dataset Size per class (6,8,9)	SVM	Decision Tree	Random Forest
2000 rows	87.89%	**92.00%**	90.11%

TABLE 3.12: Single Digit (6, 8, 9) - Binary - Accuracy results from various classifiers using a feature length of 10,000.

2000 rows per class (6,8,9)	6	8	9
SVM	93.84%	86.92%	**93.75%**
Decision Tree	**97.25%**	**91.33%**	93.5%
Random Forest	96.17%	87.5%	95.42%

As per the information in Table 3.10, the Random Forest classifier exhibited superior accuracies in predicting 2, 3, and 4. Additionally, based on the data in Table 3.12, the Decision Tree classifier showed higher accuracies in predicting 6 and 8, whereas the SVM classifier in predicting 9. The multi-class classification accuracies of these sets are higher than those of the 0, 1, 5, 7 set, primarily because they are trained with three classes each, whereas the latter involves four classes. Having obtained individual samples for digits 0 to 9, we conducted both multi-class and binary classifications on a dataset featuring 10,000 features and 2000 samples per class, totaling 20,000 samples. The outcomes of the multi-class and binary classifications are presented in Table 3.13 and Table 3.14, respectively.

TABLE 3.13: Single Digit (0 to 9) - Multi class - Accuracy results from various classifiers using a feature length of 10,000.

Dataset Size per class (0 to 9)	SVM	Decision Tree	Random Forest
2000 rows	82.00%	86.14%	**86.43%**

TABLE 3.14: Single Digit (0 to 9) - Binary - Accuracy results from various classifiers using a feature length of 10,000.

2000 rows per class (0 to 9)	SVM	Decision Tree	Random Forest
0	**97.67%**	95.17%	97.25%
1	91.25%	90.33%	**92.17%**
2	96.42%	**97.34%**	96.17%
3	93.67%	**95.00%**	93.25%
4	97.17%	**98.25%**	96.67%
5	89.08%	88.83%	**90.25%**
6	96.75%	**98.08%**	96.83%
7	91.75%	**93.75%**	90.67%
8	**94.84%**	94.50%	94.17%
9	97.50%	97.67%	**97.92%**

As indicated in Table 3.13, both Random Forest and Decision Tree classifiers exhibited strong performance, with Random Forest achieving a slightly higher overall accuracy of **86.43%** in multi-label classification on the dataset encompassing samples of digits 0 to 9. Based on the findings from Table 3.14, it can be concluded that we can predict whether the screen is displaying a specific digit or not, with higher accuracies of greater than 88%.

3.3.3 Classification of samples according to the single-digit displayed at multiple positions of the screen

In this category, we collected 2000 traces for each digit, with 224 traces featuring the digit at the center of the screen and 222 traces each in all corners and midpoints of corners. This distribution, totaling 224 + 1776 = 2000 traces, ensures equal weight distribution for different positions.Similar to our previous approach for the case of a single digit at the center of the screen, we conducted trace sampling for sets (0, 1, 5, 7), (2, 3, 4), and (6, 8, 9). We performed feature length classification analysis for feature lengths ranging from 5000 to 50000 for each set. Additionally, we carried out both multi-class and binary classification for each set individually. Finally, we conducted multi-class and binary classification on the entire set of digits from 0 to 9. The outcomes of the feature length analysis for the sets (0, 1, 5, 7), (2, 3, 4), and (6, 8, 9) can be found in Table 3.15.

TABLE 3.15: Digit at Positions - Accuracy outcomes corresponding to different feature lengths and sample sets.

Dataset Size per class - 500 rows	5000 columns	10000 columns	15000 columns	20000 columns	25000 columns	50000 columns
(0, 1, 5, 7)	89.71%	90.08%	**90.58%**	89.38%	89.88%	87.05%
(2, 3, 4)	88.72%	88.00%	88.22%	**88.28%**	87.61%	86.17%
(6, 8, 9)	81.67%	**82.72%**	82.72%	82.00%	81.39%	78.67%

From Table 3.15, it is evident that the best accuracies were achieved at feature lengths of 10,000 and 15,000. Building on this insight and considering our objective of minimizing feature length while maintaining good accuracies, we opted to proceed with datasets featuring 10,000 features. Subsequently, we conducted both multi-label and binary classifications on the sets (0, 1, 5, 7), (2, 3, 4), and (6, 8, 9). The results of the multi-class classification are presented in Table 3.16, and the outcomes of binary classification on the sets (0, 1, 5, 7), (2, 3, 4), and (6, 8, 9) are outlined in Table 3.17, Table 3.18, and Table 3.19, respectively. The highest accuracies in their respective categories are highlighted in bold.

TABLE 3.16: Single Digit at Positions - Multi class - Accuracy results from various classifiers using a feature length of 10,000.

Dataset Size per class (2000)	SVM	Decision Tree	Random Forest
(0,1,5,7)	**90.08%**	85.54%	87.96%
(2,3,4)	88.00%	**90.38%**	90.28%
(6,8,9)	82.72%	83.72%	**85.27%**

TABLE 3.17: Single Digit at Positions (0, 1, 5, 7) - Binary - Accuracy results from various classifiers using a feature length of 10,000.

2000 rows per class (0,1,5,7)	0	1	5	7
SVM	**99.08%**	93.17%	**87.92%**	**91.25%**
Decision Tree	98.58%	90.92%	84.42%	91.00%
Random Forest	97.17%	**95.17%**	86.33%	90.33%

TABLE 3.18: Single Digit at Positions (2, 3, 4) - Binary - Accuracy results from various classifiers using a feature length of 10,000.

2000 rows per class (2,3,4)	2	3	4
SVM	95.91%	86.50%	92.00%
Decision Tree	95.91%	**89.50%**	**92.58%**
Random Forest	**96.83%**	87.75%	90.92%

TABLE 3.19: Single Digit at Positions (6, 8, 9) - Binary - Accuracy results from various classifiers using a feature length of 10,000.

2000 rows per class (6,8,9)	6	8	9
SVM	**94.42%**	80.92%	87.50%
Decision Tree	91.92%	82.25%	**89.92%**
Random Forest	93.58%	**86.25%**	89.00%

With individual samples collected for digits 0 to 9 at various positions, we performed both multi-class and binary classifications on a dataset comprising 10,000 features and 2000 samples per class. The outcomes of the multi-class and binary classifications are presented in Table 3.20 and Table 3.21, respectively.

TABLE 3.20: Single Digit at Positions (0 to 9) - Multi class - Accuracy results from various classifiers using a feature length of 10,000.

Dataset Size per class (0 to 9)	SVM	Decision Tree	Random Forest
2000 rows	**87.52%**	84.04%	85.38%

TABLE 3.21: Single Digit at Positions (0 to 9) - Binary - Accuracy results from various classifiers using a feature length of 10,000.

2000 rows per class (0 to 9)	SVM	Decision Tree	Random Forest
0	**99.50%**	98.42%	98.58%
1	**97.17%**	95.42%	94.83%
2	**98.00%**	97.34%	97.25%
3	93.00%	**94.17%**	92.00%
4	96.50%	**96.58%**	96.50%
5	**93.67%**	91.83%	92.17%
6	96.50%	94.17%	**95.84%**
7	**95.08%**	93.17%	93.08%
8	92.00%	91.08%	**92.17%**
9	**94.75%**	94.25%	**94.75%**

As highlighted in Table 3.20, the SVM classifier demonstrated robust performance, achieving an overall accuracy of **87.52%** in multi-class classification. The information in Table 3.21 pertaining to single digits at various positions allows us to conclude that we can effectively predict whether the screen is showcasing a particular digit or not. This is achieved with a feature length of 10,000, attaining accuracies that surpass 91%.

3.3.4 Classification of samples according to the double digit numbers at the center of the screen

In our pursuit to assess the capability of a machine learning model in classifying double-digit samples, we curated a set of four numbers (13, 31, 38, 83). These numbers were chosen due to their shared, flipped, cross-matched, and visually similar digits, designed to create a challenging classification scenario. Given the impracticality of sampling all 100 double-digit numbers (ranging from 00 to 99) due to storage constraints and the computational burden of training a model for numerous classes, this evaluation serves to investigate whether visual resemblances manifest in the frequency domain, as reflected in the model's performance.

In our study, we collected 2000 traces for each class and conducted feature length analysis, multi-class classification, binary (1 vs rest) classification, and binary (1 vs 1) classification. We repeated the experiment with 500 and 2000 traces per class to assess the impact of

sample size on model performance. The results of these analyses are summarized in the respective tables below.

Table 3.22: Double digit at center - Accuracy outcomes corresponding to different feature lengths.

Dataset Size per class **(SVM)**	5000 columns	10000 columns	15000 columns	20000 columns	25000 columns	50000 columns
500 rows	98.83%	**99.00%**	99.00%	98.83%	99.17%	99.00%
2000 rows	96.95%	**97.08%**	97.17%	97.08%	96.92%	96.29%

Table 3.23: Double digit at center - Multi class - Accuracy results from various classifiers using a feature length of 10,000.

Dataset Size per class	SVM Linear	Decision Tree	Random Forest
500 rows	**99.00%**	97.50%	98.83%
2000 rows	**97.08%**	96.04%	97.08%

Table 3.24: Double Digits at center - Binary (1 vs rest) - Accuracy results for a feature length of 10,000 and 500 samples per class.

Dataset Size per class (500)	SVM Linear	Decision Tree	Random Forest
13	**99.00%**	98.67%	97.67%
31	**98.00%**	95.33%	97.00%
38	**99.67%**	98.33%	98.67%
83	**99.33%**	99.00%	98.67%

Table 3.25: Double Digits at center - Binary (1 vs rest) - Accuracy results for a feature length of 10,000 and 2000 samples per class.

Dataset Size per class (2000)	SVM Linear	Decision Tree	Random Forest
13	**97.50%**	94.33%	97.33%
31	**95.58%**	94.33%	94.92%
38	**99.83%**	99.08%	99.58%
83	**99.83%**	99.42%	99.83%

Table 3.26: Double Digits at center - Binary (1 vs 1) - SVM Linear - Accuracy results for a feature length of 10,000 and 500 samples per class.

SVM	13	31	38
31	99.00%	—	—
38	100.00%	99.67%	—
83	99.67%	99.67%	99.33%

Table 3.27: Double Digits at center - Binary (1 vs 1) - Decision Tree - Accuracy results for a feature length of 10,000 and 500 samples per class.

DT	13	31	38
31	98.33%	—	—
38	100.00%	100.00%	—
83	99.67%	99.33%	98.33%

TABLE 3.28: Double Digits at center - Binary (1 vs 1) - Random Forest - Accuracy results for a feature length of 10,000 and 500 samples per class.

RF	13	31	38
31	100.00%	—	—
38	100.00%	99.67%	—
83	100.00%	99.33%	99.00%

TABLE 3.29: Double Digits at center - Binary (1 vs 1) - SVM Linear - Accuracy results for a feature length of 10,000 and 2000 samples per class.

SVM	13	31	38
31	94.83%	—	—
38	99.92%	100.00%	—
83	100.00%	99.92%	99.83%

TABLE 3.30: Double Digits at center - Binary (1 vs 1) - Decision Tree - Accuracy results for a feature length of 10,000 and 2000 samples per class.

DT	13	31	38
31	92.25%	—	—
38	99.83%	99.91%	—
83	99.83%	99.67%	99.33%

TABLE 3.31: Double Digits at center - Binary (1 vs 1) - Random Forest - Accuracy results for a feature length of 10,000 and 2000 samples per class.

RF	13	31	38
31	95.83%	—	—
38	99.83%	99.92%	—
83	99.83%	99.83%	99.50%

In the context of double digit number at center, in Table 3.22 and Table 3.23, models trained on a smaller dataset exhibited higher test accuracy compared to models trained on larger datasets in almost every case, even in Tables 3.26 to 3.31, One plausible explanation is over-fitting, whereby the model trained on more data memorizes the idiosyncrasies of the training examples rather than discerning general patterns. Conversely, the model trained on less data is compelled to focus on these overarching patterns, resulting in superior performance on unseen data. Additionally, random fluctuations inherent in the training process, particularly with limited datasets, could favor the model with less data, leading to marginally better performance on the test set. Furthermore, if the dataset contains inherent noise or randomness, a simpler model trained on a smaller dataset may excel in capturing the underlying signal, contrasting with a more complex model susceptible to fitting the noise in the larger dataset.

From the Binary 1 vs 1 classification accuracies presented in Tables 3.26 to 3.31, a discernible pattern emerges suggesting a potential influence of visual resemblance on model

accuracies. Notably, models trained on cross-matched pairs such as (13,31) and (38,83) consistently exhibit lower accuracies across the tables, except for Table 3.28. Conversely, models trained on pairs with one shared digit, such as (13,38), (13,83), (31,38), and (31,83), consistently display higher accuracies. However, caution is warranted in drawing definitive conclusions, as even the lowest accuracy among these observations surpasses 92%. It is plausible that despite visual resemblance, samples from distinct classes may exhibit minimal correlation in their FFT traces. Factors such as the random state parameter, data quality, and noise levels could also contribute to the observed variations in model accuracies. Therefore, while the presence of visual resemblance appears to influence model performance, further analysis is necessary to elucidate the underlying mechanisms driving these outcomes.

3.3.5 Classification of samples according to the double digit numbers at different positions of the screen

We collected 2000 traces for each digit, with 224 traces featuring the digit at the center of the screen and 222 traces each in all corners and midpoints of corners. This distribution, totaling $224 + 1776 = 2000$ traces, ensures equal weight distribution for different positions. Similar to our previous approach for the case of a double digit at the center of the screen, we conducted sampling on the same numbers 13, 31, 38, and 83 at different positions. conducted feature length analysis, multi-class classification, binary (1 vs rest) classification, and binary (1 vs 1) classification. We repeated the experiment with 500 and 2000 traces per class to assess the impact of sample size on model performance. The results of these analyses are summarized in the respective tables below.

TABLE 3.32: Double digit at positions - Accuracy outcomes corresponding to different feature lengths.

Dataset Size per class **(SVM)**	5000 columns	10000 columns	15000 columns	20000 columns	25000 columns	50000 columns
500 rows	**81.33%**	80.17%	79.67%	78.83%	78.83%	72.33%
2000 rows	78.79%	**80.00%**	79.79%	79.54%	79.04%	77.58%

TABLE 3.33: Double digit at positions - Multi class - Accuracy results from various classifiers using a feature length of 10,000.

Dataset Size per class	SVM Linear	Decision Tree	Random Forest
500 rows	80.17%	76.33%	**81.67%**
2000 rows	80.00%	73.58%	**82.63%**

TABLE 3.34: Double Digits at positions - Binary (1 vs rest) - Accuracy results for a feature length of 10,000 and 500 samples per class.

Dataset Size per class (500)	SVM Linear	Decision Tree	Random Forest
13	93.33%	91.33%	**93.67%**
31	84.67%	80.67%	**85.67%**
38	85.00%	86.33%	**87.00%**
83	**89.67%**	85.33%	89.33%

TABLE 3.35: Double Digits at positions - Binary (1 vs rest) - Accuracy results for a feature length of 10,000 and 2000 samples per class.

Dataset Size per class (2000)	SVM Linear	Decision Tree	Random Forest
13	**96.17%**	92.08%	96.08%
31	86.00%	82.83%	**86.67%**
38	79.17%	78.92%	**83.42%**
83	89.25%	87.75%	**91.08%**

TABLE 3.36: Double Digits at positions - Binary (1 vs 1) - SVM Linear - Accuracy results for a feature length of 10,000 and 500 samples per class.

SVM	13	31	38
31	93.67%	—	—
38	98.67%	88.00%	—
83	97.33%	97.33%	87.33%

TABLE 3.37: Double Digits at positions - Binary (1 vs 1) - Decision Tree - Accuracy results for a feature length of 10,000 and 500 samples per class.

DT	13	31	38
31	85.67%	—	—
38	98.33%	82.67%	—
83	97.00%	89.67%	81.00%

TABLE 3.38: Double Digits at positions - Binary (1 vs 1) - Random Forest - Accuracy results for a feature length of 10,000 and 500 samples per class.

RF	13	31	38
31	91%	—	—
38	98.67%	87.33%	—
83	98.33%	94.00%	87.00%

TABLE 3.39: Double Digits at positions - Binary (1 vs 1) - SVM Linear - Accuracy results for a feature length of 10,000 and 2000 samples per class.

SVM	13	31	38
31	90.33%	—	—
38	98.75%	83.75%	—
83	98.67%	97.17%	80.75%

TABLE 3.40: Double Digits at positions - Binary (1 vs 1) - Decision Tree - Accuracy results for a feature length of 10,000 and 2000 samples per class.

DT	13	31	38
31	88.58%	—	—
38	97.33%	82.33%	—
83	97.67%	94.75%	78.83%

TABLE 3.41: Double Digits at positions - Binary (1 vs 1) - Random Forest - Accuracy results for a feature length of 10,000 and 2000 samples per class.

RF	13	31	38
31	92.25%	—	—
38	99.00%	86.42%	—
83	99.17%	96.58%	84.42%

In the context of double digits at positions, unlike the clear delineation observed with double digits at the center, the discernment of over-fitting and its underlying causes remains elusive based on the pattern of computed accuracies spanning from Tables 3.32 to 3.35. Moreover, the outcomes from the binary 1 vs 1 classification tables (Tables 3.36 to 3.41) fail to provide conclusive evidence regarding the impact of visual resemblance on model accuracies.

In this scenario, potential factors contributing to the variance between data points within a class become evident. Specifically, the inherent characteristics of the data, where traces from different positions within a class exhibit notable differences despite being labeled identically during training and testing, warrant consideration. Unlike the focused dataset of double digits at the center, the absence of data showcasing lower variance or higher similarity within a class poses a challenge. However, despite these complexities, the observed accuracies demonstrate satisfactory performance in terms of classification ability.

3.3.6 Classification of samples according to the six digit numbers at the center of the screen

Now, we aim to evaluate the performance of a machine learning model for 6-digit numbers. Given the impracticality of sampling all one million possible 6-digit numbers (i.e., from 000000 to 999999), we have selected three randomly generated numbers (375194, 412031, 656018) and added 656019 to that set to examine the influence of shared digits on the model's accuracy. We conducted a similar sequence of analyses as previously done for double-digit numbers at the center i.e. Multi-class, Binary (1 vs rest) and Binary (1 vs 1). The results of these analyses are summarized in the Tables 3.42 to 3.51.

Table 3.42: Six digit at center - Accuracy outcomes corresponding to different feature lengths.

Dataset Size per class **(SVM)**	5000 columns	10000 columns	15000 columns	20000 columns	25000 columns	50000 columns
500 rows	**98.00%**	97.83%	97.33%	97.17%	96.83%	96.00%
2000 rows	95.38%	**95.88%**	95.79%	95.17%	95.38%	94.54%

Table 3.43: Six digit at center - Multi class - Accuracy results from various classifiers using a feature length of 10,000.

Dataset Size per class	SVM Linear	Decision Tree	Random Forest
500 rows	**97.83%**	96.50%	94.67%
2000 rows	**95.88%**	93.29%	95.00%

Table 3.44: Six Digits at center - Binary (1 vs rest) - Accuracy results for a feature length of 10,000 and 500 samples per class.

Dataset Size per class (500)	SVM Linear	Decision Tree	Random Forest
375194	**97.67 %**	92.67%	96.67%
412031	**96.33%**	94.67%	92.67%
656018	98.67%	**100.00%**	97.33%
656019	**98.00%**	95.34%	96.33%

Table 3.45: Six Digits at center - Binary (1 vs rest) - Accuracy results for a feature length of 10,000 and 2000 samples per class.

Dataset Size per class (2000)	SVM Linear	Decision Tree	Random Forest
375194	**95.75%**	92.75%	95.58%
412031	95.75%	92.58%	**96.17%**
656018	**99.00%**	98.33%	98.17%
656019	**99.08%**	97.42%	98.00%

Table 3.46: Six Digits at center - Binary (1 vs 1) - SVM Linear - Accuracy results for a feature length of 10,000 and 500 samples per class.

SVM	375194	412031	656018
412031	97.67%	—	—
656018	100.00%	100.00%	—
656019	100.00%	100.00%	97.67%

Table 3.47: Six Digits at center - Binary (1 vs 1) - Decision Tree - Accuracy results for a feature length of 10,000 and 500 samples per class.

DT	375194	412031	656018
412031	94.00%	—	—
656018	100.00%	100.00%	—
656019	100.00%	99.67%	100.00%

Table 3.48: Six Digits at center - Binary (1 vs 1) - Random Forest - Accuracy results for a feature length of 10,000 and 500 samples per class.

RF	375194	412031	656018
412031	96.00%	—	—
656018	100.00%	99.67%	—
656019	100.00%	100.00%	96.67%

Table 3.49: Six Digits at center - Binary (1 vs 1) - SVM Linear - Accuracy results for a feature length of 10,000 and 2000 samples per class.

SVM	375194	412031	656018
412031	92.58%	—	—
656018	100.00%	99.92%	—
656019	100.00%	99.75%	98.92%

Table 3.50: Six Digits at center - Binary (1 vs 1) - Decision Tree - Accuracy results for a feature length of 10,000 and 2000 samples per class.

DT	375194	412031	656018
412031	89.75%	—	—
656018	99.92%	99.58%	—
656019	99.83%	99.25%	98.25%

Table 3.51: Six Digits at center - Binary (1 vs 1) - Random Forest - Accuracy results for a feature length of 10,000 and 2000 samples per class.

RF	375194	412031	656018
412031	92.33%	—	—
656018	99.83%	99.50%	—
656019	100.00%	99.67%	97.67%

In the context of six-digit numbers at the center, the accuracies obtained across all types of classifications are notably high, surpassing 93% in Multi class classification and exceeding 92% in Binary classification (both 1 vs rest and 1 vs 1). Notably, the pattern of accuracies derived from Binary 1 vs 1 classification indicates that the presence of shared digits did not lead to significantly lower accuracies, as observed in the case of double digits at the center. This suggests that the model effectively distinguishes between nearest neighbors despite the existence of shared digits. From this observation, it can be inferred that the FFT traces of nearest neighbor digits exhibit sufficient variance for robust classification, notwithstanding the presence of shared digits and background noise.

Chapter 4

Screen Gleaning Attack on Smartphones

This chapter provides detailed information about the screen gleaning setup installation, emage acquisition, the challenges encountered, and the solutions implemented to address these challenges.

4.1 Configuration and Installation of Attack Setup

The configuration for our implementation of the screen gleaning attack involves several critical components, including the Langer EMV LF-R near field probe, Langer PA-303 preamplifier, BNC female-to-female connector, Ettus USRP X310, and a workstation PC, as illustrated in Figure 4.1. To successfully generate 'emages' (electromagnetic images), it is necessary to meet several prerequisites on the workstation PC. These prerequisites include the installation of UHD (USRP Hardware Driver), PyBombs (Python Build Overlay Managed Bundle System), and JDK (Java Development Kit).

Additionally, it is crucial to ensure that the UHD image of the corresponding version is installed or flashed onto the FPGA within the USRP X310 SDR (Software Defined Radio) device. This step is essential for the proper functioning and compatibility of the system.

The installation process involves several detailed steps, which must be followed meticulously to ensure the correct setup of the hardware and software components. Below is the step-by-step procedure for the required installations.

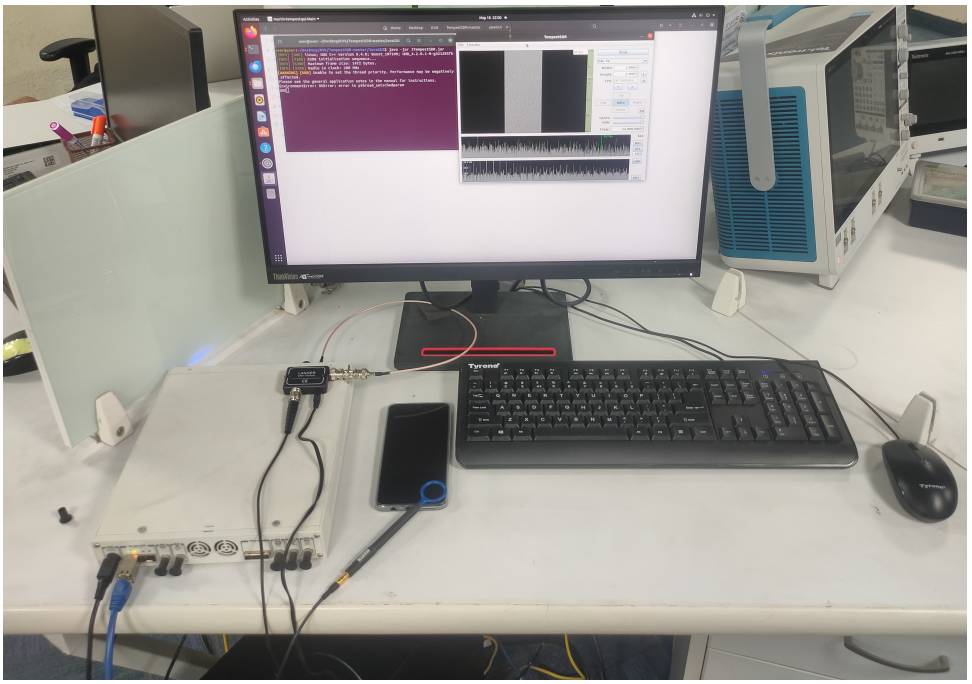


FIGURE 4.1: Our Configuration for Screen Gleaning Attack Setup.

4.1.1 Installation of USRP Hardware Driver (UHD) and FPGA Flashing

UHD is the driver that facilitates low-level communication between the SDR and the Workstation PC. The steps for installing UHD on both the PC and the SDR are as follows: To facilitate the seamless installation of UHD, begin by installing PyBombs. Follow the sequence of commands provided in Figure 4.2. This process will install a recent, stable version of UHD on your PC, though it may not be the latest release.

Once UHD is installed, the next step is to flash the corresponding UHD FPGA image onto the memory of the SDR. First, download the necessary image files to your PC using the commands listed in Figure 4.3. With UHD installed and the images downloaded, you can proceed to connect the SDR to the PC via Ethernet. Configure the PC's IP address to 192.168.10.1; the default IP address for the SDR is 192.168.10.2. Ensure that any automatic IP configuration services, such as Network Manager, are either deactivated or configured to avoid altering the network device settings. Failure to do so could potentially result in bricking the USRP. Additionally, never attempt to hot plug the SDR while its power switch is on.

After establishing the physical connection and configuring the network, turn on the SDR. To complete the process, follow the instructions in Figure 4.4 to flash the FPGA image onto the SDR. The following instruction snippets are referenced from [8] and [9]. You may refer to these sources for additional guidance.

Caution: Whenever possible, install in the default directories to ensure proper configuration and avoid potential issues.

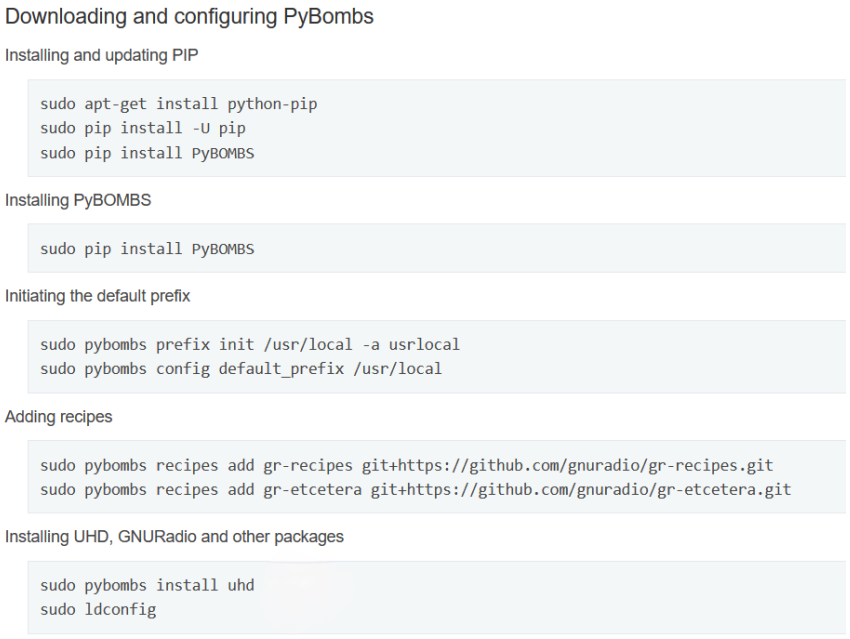


FIGURE 4.2: PyBombs and UHD installation commands

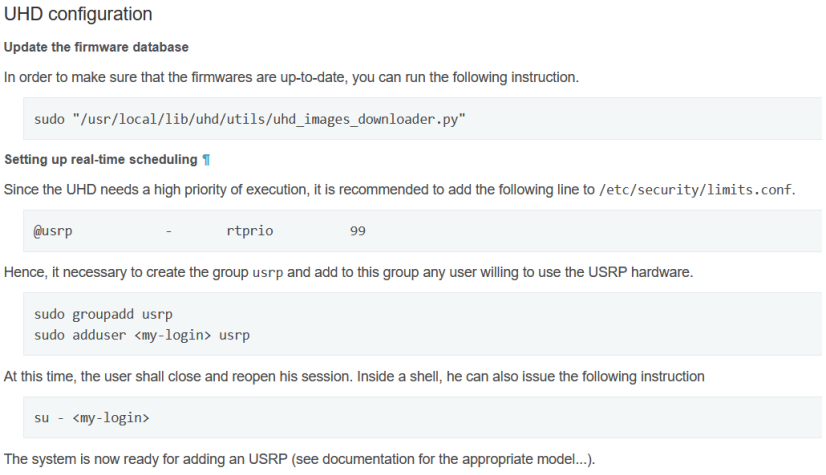
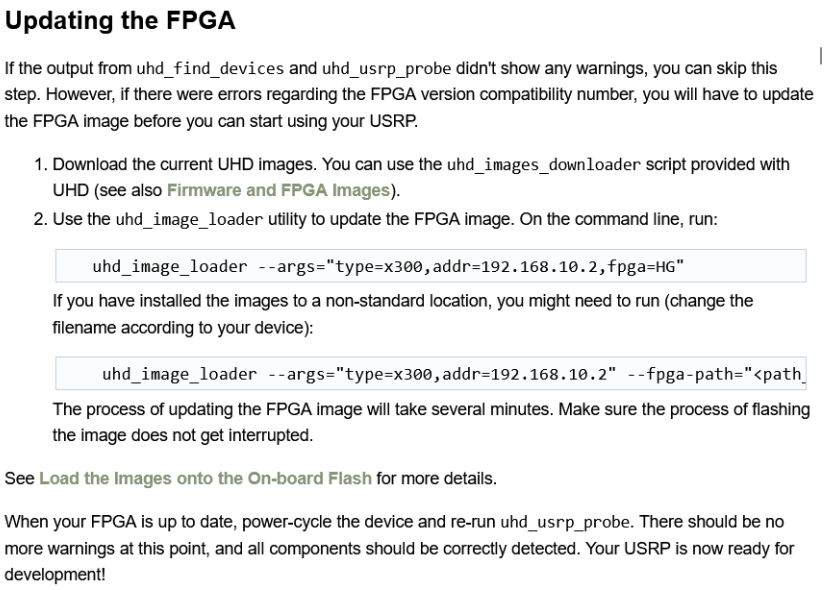


FIGURE 4.3: Commands for Downloading UHD Images



Updating the FPGA

If the output from `uhd_find_devices` and `uhd_usrp_probe` didn't show any warnings, you can skip this step. However, if there were errors regarding the FPGA version compatibility number, you will have to update the FPGA image before you can start using your USRP.

1. Download the current UHD images. You can use the `uhd_images_downloader` script provided with UHD (see also **Firmware and FPGA Images**).
2. Use the `uhd_image_loader` utility to update the FPGA image. On the command line, run:

```
uhd_image_loader --args="type=x300,addr=192.168.10.2,fpga=HG"
```

If you have installed the images to a non-standard location, you might need to run (change the filename according to your device):

```
uhd_image_loader --args="type=x300,addr=192.168.10.2" --fpga-path="<path_
```

The process of updating the FPGA image will take several minutes. Make sure the process of flashing the image does not get interrupted.

See **Load the Images onto the On-board Flash** for more details.

When your FPGA is up to date, power-cycle the device and re-run `uhd_usrp_probe`. There should be no more warnings at this point, and all components should be correctly detected. Your USRP is now ready for development!

FIGURE 4.4: Commands for Flashing UHD Images

If the network configuration is as it should be then executing the command '**uhd_find_devices**' will detect your USRP and provide pertinent information, as illustrated in Figure 4.5. Additionally, you should be able to successfully **ping the USRP at 192.168.10.2**, as demonstrated in Figure 4.6. Furthermore, when the FPGA image flash corresponds to the correct version and type (HG - 1 Gigabit, XG - 10 Gigabit), executing '**uhd_usrp_probe**' will yield relevant information, as depicted in Figure 4.7. With these steps successfully executed, we can confirm that the UHD installation process is complete and the software is functioning correctly.

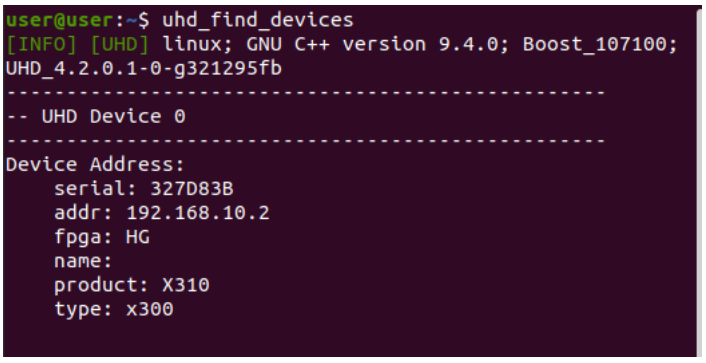


FIGURE 4.5: Snippet of Terminal Output for the 'uhd_find_devices' Command

```
user@user:~$ ping 192.168.10.2
PING 192.168.10.2 (192.168.10.2) 56(84) bytes of data.
64 bytes from 192.168.10.2: icmp_seq=1 ttl=32 time=0.781
ms
64 bytes from 192.168.10.2: icmp_seq=2 ttl=32 time=0.802
ms
64 bytes from 192.168.10.2: icmp_seq=3 ttl=32 time=0.803
ms
CD^C
--- 192.168.10.2 ping statistics ---
3 packets transmitted, 3 received, 0% packet loss, time 2
036ms
rtt min/avg/max/mdev = 0.781/0.795/0.803/0.010 ms
```

FIGURE 4.6: Snippet of Terminal Output for the 'ping@192.168.10.2' Command

```
user@user:~$ uhd_usrp_probe
[INFO] [UHD] linux; GNU C++ version 9.4.0; Boost_107100; UHD_4.2.0.1-0-g321295fb
[INFO] [X300] X300 initialization sequence...
[INFO] [X300] Maximum frame size: 1472 bytes.
[INFO] [X300] Radio 1x clock: 200 MHz
 _____________________________________________________
/
|       Device: X-Series Device
|     _____________________________________________________
|    /
|    |       Mboard: X310
|    |   revision: 13
|    |   revision_compat: 7
|    |   product: 30818
|    |   mac-addr0: 00:80:2f:37:07:b0
|    |   mac-addr1: 00:80:2f:37:07:b1
|    |   gateway: 192.168.10.1
|    |   ip-addr0: 192.168.10.2
|    |   subnet0: 255.255.255.0
|    |   ip-addr1: 192.168.20.2
|    |   subnet1: 255.255.255.0
|    |   ip-addr2: 192.168.30.2
|    |   subnet2: 255.255.255.0
|    |   ip-addr3: 192.168.40.2
|    |   subnet3: 255.255.255.0
|    |   serial: 327D83B
|    |   FW Version: 6.0
|    |   FPGA Version: 38.0
|    |   FPGA git hash: 8daa80c
|    |   RFNoC capable: Yes
|    |
|    |   Time sources:  internal, external, gpsdo
|    |   Clock sources: internal, external, gpsdo
|    |   Sensors: ref_locked
|    _____________________________________________________
|    /
|    |       RFNoC blocks on this device:
|    |
|    |   * 0/DDC#0
|    |   * 0/DDC#1
|    |   * 0/DUC#0
|    |   * 0/DUC#1
|    |   * 0/Radio#0
|    |   * 0/Radio#1
|    |   * 0/Replay#0
|    _____________________________________________________
|    /
|    |       Static connections on this device:
|    |
```

FIGURE 4.7: Snippet of Terminal Output for the 'uhd_usrp_probe' Command

4.1.2 Installation of TempestSDR

To install TempestSDR, the Java Development Kit (JDK) is required as this is a Java application. It is recommended to install the JDK using the Debian package for Ubuntu, as the default installation path aligns with the guidelines provided in the TempestSDR README file [7]. Once the JDK installation is complete, download the repository from Martin Marinov's GitHub [7] as a zip file and extract it to the desired directory. For systems running Ubuntu OS, follow the steps outlined in Figure 4.8 to compile the TempestSDR jar executable.

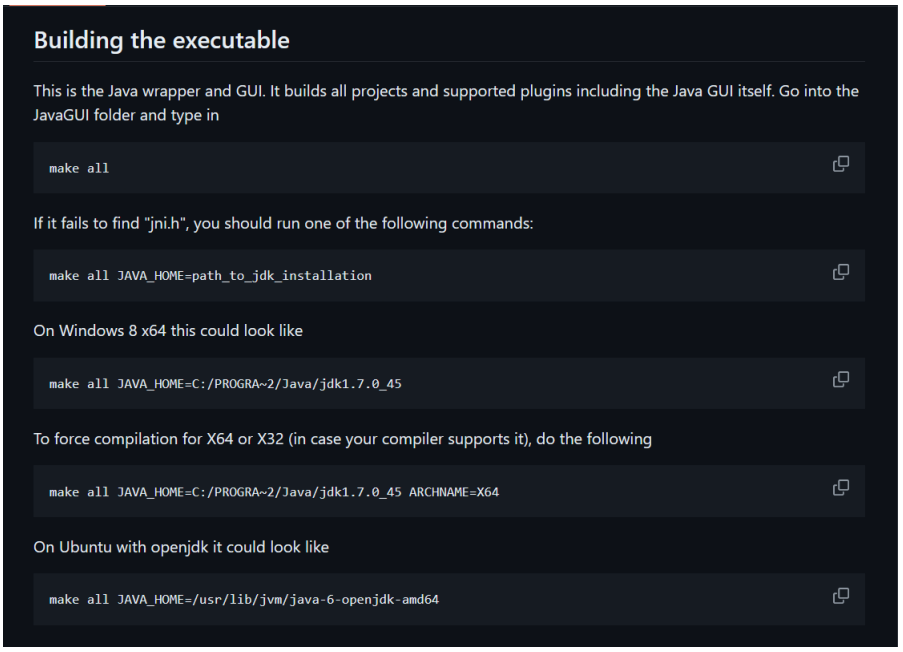


FIGURE 4.8: Compilation Command Snippet for TempestSDR's Jar Executable

Finally, execute the '**libuhd-dev**' command in the terminal as the last required step. The 'libuhd-dev' package provides the essential headers and libraries for developing applications that utilize the UHD (USRP Hardware Driver) library. On Linux, compiling the GUI will also compile the UHD driver, necessitating the installation of UHD and the corresponding Boost libraries (UHD will install them automatically). If you prefer not to compile the UHD drivers, you can skip this step by removing line 91 for Linux from the Makefile in the JavaGUI directory.

After executing these commands, you will have a '**JTempestSDR.jar**' executable in the JavaGUI directory, compiled based on the JDK version installed on your PC. Do not confuse this newly created jar file with the one originally present in the Release/JavaGUI directory upon download, as the latter may not be ready for use. Therefore, we compiled it as described above. To launch the executable created from our compilation, navigate to the JavaGUI directory and use the command '**java -jar JTempestSDR.jar**' and it should appear as shown in Figure 4.9.

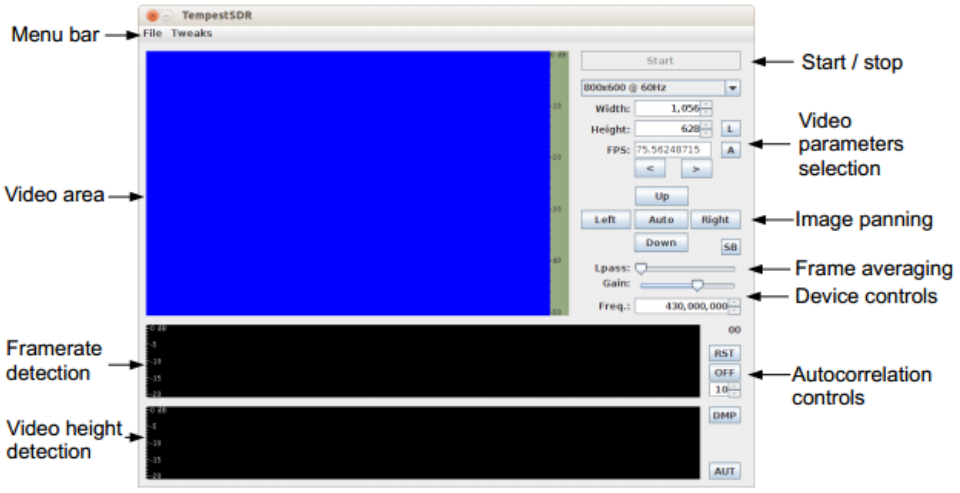


FIGURE 4.9: Snippet of TempestSDR Immediately After Opening with Callouts.

4.2 TempestSDR - Emage Acquisition and Challenges

This application was designed with a focus on monitor displays. First, let's review the features that TempestSDR offers and how to utilize them in the context of monitor displays.

- First, establish the attack setup as described in the earlier sections and run the application.
- In order to start the USRP-UHD, we need to go to File and choose Load USRP (via UHD).
- We can enter the command "*-rate* $= x$", $x \in \mathbb{W}$, which will set the sampling rate to x Hz. Where $\mathbb{W}$ is the set of whole numbers.
- If no errors occur, we will see the "Start" button becoming active. Once we press on the button, the system will start processing the data sampled by the USRP.
- The "A" button will enable the frame-rate tracking feature, a.k.a refresh rate.
- The "Auto" button will attempt to detect synchronisation regions.
- The "AUT" button will try to automatically detect the resolution and line rate (video height).
- The scale on the right of the video shows in real-time the grey-scale values that map to a particular sample intensity.

- If we turn on the "Lpass" averaging and "Gain", we will get rid of most of the noise and improve the signal-to-noise ratio bringing up the weak video signal.
- First, we need to estimate the screen refresh frequency from the target peak, Note that we can try the "AUT" button which will simply choose the highest values of the two plots.
- To do that we need to keep an eye on the autocorrelation plot and keep changing the frequency until a promising signal appears.
- However, most video displays run at known refresh rates, If we spot a signal near these values, then we know we are on to something.
- In order to set the video parameters to the frequencies in those spikes, we just need to click with the mouse on top of them.
- Once we select the best candidate for the refresh rate, we can now choose the best candidate for the line rate (height) as well.

However, when the signal-to-noise ratio is low, other repeating signals on the same frequency may be more prominent so manual adjustment would be required. The autocorrelation plots are interactive and the user can zoom with the mouse wheel and pan around for choosing small details in the plots. Digital signals will also show activity in regions that have constant intensities. That's why digital signals could be eavesdropped easier - they emit more energy because of the sharp edges of the individual bits. We may see another autocorrelation peak. This is aliasing due to the two strong synchronisation lines that appear in the video. We can increase the sampling rate of the USRP to improve the resolution of the obtained image. If we obtain a good enough signal, we can reconstruct text from the screen and save them by going to 'Tweaks' and click save screenshot.

Now, when it comes to reconstructing text from a mobile screen, we must consider the unique characteristics of mobile phone displays. Unlike monitor screens, mobile screens are not simply rotated 90 degrees in terms of pixel layout and raster format. Although the height of a monitor corresponds to the width of a mobile phone, and vice versa when a monitor and a 90-degree rotated phone display are compared, the raster display format remains a horizontal left-to-right, top-to-bottom line-by-line arrangement when viewed in their respective orientations. Typically, mobile phones are viewed in portrait mode, while monitor displays are in landscape mode i.e. when a monitor and a 90-degree rotated phone display are compared monitor has horizontal raster layout and mobile phone has vertical raster layout in perspective of comparision.

However, since TempestSDR software was developed for monitor displays, its automated features for refresh-rate (frame-rate) and line-rate (horizontal frequency) estimation do not function correctly due to the differences in monitor and mobile screen technologies mentioned earlier. Consequently, manual entry of video parameters is required. Enter the mobile screen's height, width, and refresh-rate (FPS) unchanged with respect to monitor. Additionally, input the leakage center frequency observed manually into the 'Freq' text-input field. Once these parameters are set, press 'Start'. The application should appear as shown in Figure 4.10.

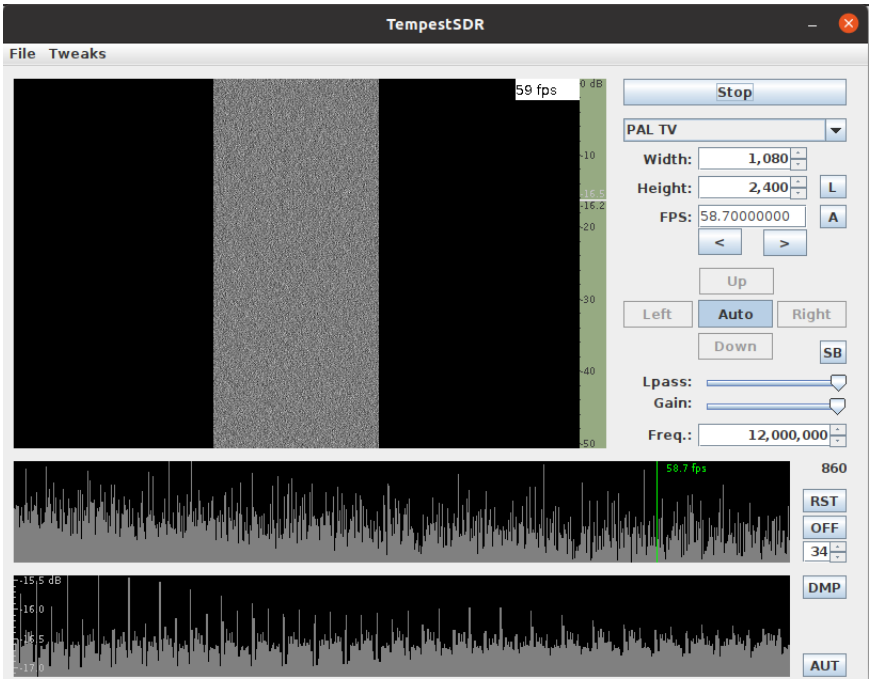


FIGURE 4.10: Snippet of Operational TempestSDR.

At a sampling rate of 25 MSamples/s (–rate=250000000 MHz), we entered the respective display parameters. Following a stepwise manual observation of the leakage center frequency, we identified a weak video signal at 115 MHz for the test image. This signal could potentially be a weaker harmonic of a stronger signal at even higher center frequencies, such as 1.51 GHz, as described in **Section 3.2.4 of [10]**. Additionally, according to the methodology outlined in **Section 3.2.3 of [10]**, the lower harmonic starts from 151.6 MHz. This observation was made for the Realme XT, which has a resolution of 2340x1080, a refresh rate of 60Hz, and a color depth of 10 bits per color (AMOLED) . The resulting emage, depicted in Figure 4.11, is suboptimal for text reconstruction from the screen. It exhibits significant horizontal drift, attributed to pixel interpolation where unknown pixel data is estimated based on neighboring pixels in a horizontal pixel line. This drift arises

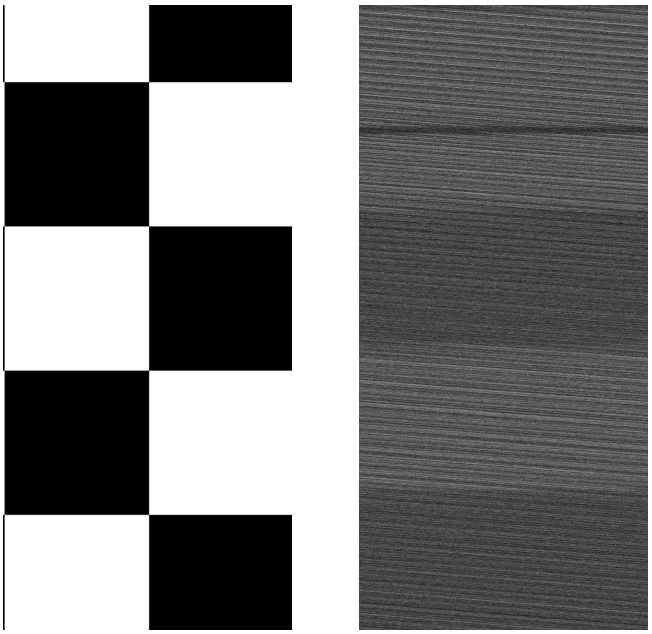


FIGURE 4.11: Comparison of Screen Content and its Emage Snippet

due to lower sampling rates than required for accurate retrieval as stated **Section 3.4 in** [10].The suboptimal emage quality may also be attributed to the LF-R 400 probe-antenna used, which operates accurately within the 0 to 50 MHz range. Additionally, the thin dark line visible represents the frame boundary, with the upper part depicting the previous frame and the lower part representing the new frame corresponding to a certain time-stamp. This discrepancy occurred as the refresh rate was not synchronized properly.

Chapter 5

Conclusions and future work

This chapter summarizes the key findings and results from the research on screen gleaning and TEMPEST attacks on smartphones. It also outlines potential enhancements and future directions for further research and experimentation.

5.1 Screen Gleaning Attack on Smartphones

As outlined in **Table 3 of** [11], the standard connection of our Ettus X-310 to the PC utilizes a 1-Gigabit interface, which supports a maximum throughput or sampling rate of 25 MS/s. However, as detailed in **Section 3.4 of** [10], upgrading to a 10-Gigabit interface can significantly enhance the quality of the acquired emages. With a 10-Gigabit interface, the maximum sampling rate can increase to 200 MS/s, a substantial improvement.

To facilitate this upgrade, we propose installing a 10-Gigabit Ethernet card, specifically the Intel® Ethernet Converged Network Adapter X710-DA2, into the workstation PC. This upgrade will not only boost the data transfer rates but also enhance the overall performance of the signal acquisition system. Additionally, employing a near-field probe capable of receiving signals in the 30 MHz to 3 GHz range, such as the Langer EMV RF-R 400, will further enrich the signal acquisition process. This probe's capability to capture a wider range of frequencies will provide a more detailed and comprehensive dataset.

These technological enhancements will enable us to sample at higher frequencies, thereby facilitating the acquisition of higher quality emages. This, in turn, will significantly improve our ability to conduct and replicate screen gleaning attacks aimed at cracking 6-digit

OTPs. Furthermore, these advancements will support more accurate text content reading and enable image classification-based screen content prediction.

By integrating these upgrades, we will be better equipped to capture and analyze the subtle electromagnetic signals that emanate from mobile devices. This will enhance our understanding of the underlying processes and improve our ability to develop countermeasures against such attacks. Additionally, this will allow us to push the boundaries of what is possible in the realm of screen gleaning and electromagnetic signal analysis, paving the way for new discoveries and innovations in this field.

5.2 TEMPEST Attack on Smartphones using Oscilloscope

The FFT traces of leakage EM signals are highly distinguishable through machine learning classification, demonstrating that the FFT representation of the traces exhibits significant variance among different classes. This variance indicates that the traces contain information about the screen content specific to each class, leading to improved classification results as observed previously. In essence, a supervised machine learning model can effectively classify one trace from another based on the class or screen content.

For future experimentation, we propose sending real-time signals from an oscilloscope to a PC and utilizing an RNN (Recurrent Neural Network) to predict the screen content based on the time-series signal rather than the FFT-transformed signal. While this approach leverages the sequential nature of time-series data and the capabilities of RNNs to model temporal dependencies, the expected accuracies from such a procedure are anticipated to be lower than those obtained through FFT analysis. This is due to the FFT's ability to highlight frequency-domain characteristics that are more distinguishable and relevant for classification purposes. Nonetheless, exploring the use of RNNs for time-series signal analysis could provide valuable insights and potentially uncover new avenues for improving screen content prediction based on EM signal leakages.

Bibliography

[1] U. N. S. Agency., "Tempest: A signal problem.," 1972.

[2] W. V. Eck, "Electromagnetic radiation from video display units: an eavesdropping risk?," *Computers Security, 4(4):269-286*, 1985.

[3] M. G. Kuhn, "Compromising emanations: eavesdropping risks of computer displays.," *University of Cambridge Computer Laboratory, Technical Report, UCAM-CL-TR-577*, 2003.

[4] U. S. Furkan Elibol and I. Erer, "Realistic eavesdropping attacks on computer displays with low-cost and mobile receiver system.," *Signal Processing Conference (EUSIPCO), 2012 Proceedings of the 20th European, pages 1767-1771. IEEE*, 2012.

[5] K. I. Kamaruddin Abdul Ghani, Kaharudin Dimyati and L. S. Supian, "Radiated emission from handheld devices with touch-screen lcds.," *Intelligence and Security Informatics Conference (EISIC), 2013 European, pages 219-219. IEEE*, 2013.

[6] Z. Liu, N. Samwel, L. Weissbart, Z. Zhao, D. Lauret, L. Batina, and M. Larson, "Screen gleaning: A screen reading tempest attack on mobile devices exploiting an electromagnetic side channel.," *arXiv preprint arXiv:2011.09877*, 2020.

[7] M. Marinov, "Tempestsdr."

[8] Ralf, "Uhd and gnuradio installation from sources with pybombs."

[9] E. Research, "Usrp hardware driver and usrp manual-x310."

[10] M. Marinov, "Tempestsdr-documentation."

[11] E. Research, "Usrp hardware driver and usrp manual-x310-choosing-a-host-interface."